Tightly-Secure Authenticated Key Exchange[1]

Christoph Bader[*], Dennis Hofheinz[†], Tibor Jager[*], Eike Kiltz[*], and Yong Li[*]

[*]Horst Görtz Institute, Ruhr-University Bochum,
{christoph.bader,tibor.jager,eike.kiltz,yong.li}@rub.de
[†]Karlsruhe Institute of Technology, dennis.hofheinz@kit.edu

Abstract

We construct the first Authenticated Key Exchange (AKE) protocol whose security does not degrade with an increasing number of users or sessions. We describe a three-message protocol and prove security in an enhanced version of the classical Bellare-Rogaway security model.

Our construction is modular, and can be instantiated efficiently from standard assumptions (such as the SXDH or DLIN assumptions in pairing-friendly groups). For instance, we provide an SXDH-based protocol whose communication complexity is only 14 group elements and 4 exponents (plus some bookkeeping information).

Along the way we develop new, stronger security definitions for digital signatures and key encapsulation mechanisms. For instance, we introduce a security model for digital signatures that provides existential unforgeability under chosen-message attacks in a *multi-user setting* with *adaptive corruptions of secret keys*. We show how to construct efficient schemes that satisfy the new definitions with *tight* security proofs under standard assumptions.

1 Introduction

Authenticated Key Exchange (AKE) protocols allow two parties to establish a cryptographic key over an insecure channel. Secure AKE protects against strong active attackers that may for instance read, alter, drop, replay, or inject messages, and adaptively corrupt parties to reveal their long-term or session keys. This makes such protocols much stronger (and thus harder to construct) than simpler passively secure key exchange protocols like e.g. [DH76].

The most prominent example of an AKE protocol is the TLS Handshake [DA99, DR06, DR08], which is widely used for key establishment and authentication on the Internet. The widespread use of TLS makes AKE protocols one of the most widely-used cryptographic primitives. For example, the social network Facebook.com reports 802 million *daily* active users on average in September 2013. This makes more than 2^{29} executions of the TLS Handshake protocol *per day* only on this single web site.[2] The wide application of AKE protocols makes it necessary and interesting to study their security in large-scale settings with many millions of users.

Provably-secure AKE and tight reductions. A reduction-based security proof describes an algorithm, the *reduction*, which turns an efficient attacker on the protocol into an efficient algorithm solving an assumed-to-be-hard computational problem. The quality of such a reduction can be measured by its efficiency: the running time and success probability of the reduction running the

[1]A public version of this paper has been posted to the ePrint Archive at `http://eprint.iacr.org/2014/`.

[2]Figure obtained from `http://newsroom.fb.com/Key-Facts` on May 26, 2014. We assume that each active user logs-in once per day.

attacker as a subroutine, relative to the running time and success probability of the attacker alone. Ideally the reduction adds only a minor amount of computation and has about the same success probability as the attacker. In this case the reduction is said to be *tight*.

The existence of tight security proofs has been studied for many cryptographic primitives, like digital signatures [Ber08, Sch11, KK12], public-key encryption [BBM00, HJ12a, LJYP14], or identity-based encryption [CW13, BKP14]. However, there is no example of an authenticated key exchange protocol that comes with tight security proof under a standard assumption, not even in the Random Oracle Model [BR93b].

Known provably secure AKE protocols come with a reduction which loses a factor that depends on the number μ of users and the number ℓ of sessions per user. The loss of the reduction ranges typically between $1/(\mu \cdot \ell)$ (if the reduction has to guess only one party participating in a particular session) and $1/(\mu \cdot \ell)^2$ (if the reduction has to guess both parties participating in a particular session). This may become significant in large-scale applications. We also consider tight reductions as theoretically interesting in their own right, because it is challenging to develop new proof strategies that avoid guessing. The difficulty of constructing tightly secure AKE is discussed in more detail in Appendix A.

Our contribution. We construct the first AKE protocols whose security does not degrade in the number of users and instances. Following [BR93a] we consider a very strong security model, which allows adaptive corruptions of long-term secrets, adaptive reveals of session keys, and multiple adaptive Test queries.

Our model provides *perfect forward secrecy* [BWJM97, Kra05]: the corruption of a long-term secret does not foil the security of previously established session keys. Furthermore, we prevent *key-compromise impersonation* attacks [BWM98, GBN09]: in our security model, an attacker may introduce maliciously-generated keys. On the other hand, we do not allow reveals of internal states or intermediate results of computations, as considered in the (extended) Canetti-Krawczyk model [CK01, LLM07]. The existence of a tightly secure construction in such a model is an interesting open problem.

While our approach is generic and modular, we give efficient instantiations from standard assumptions (such as the SXDH or DLIN assumptions in pairing-friendly groups). Specifically, we propose an SXDH-based AKE protocol with a communication complexity of only 14 group elements and 4 exponents (plus some bookkeeping information). The security reduction to SXDH loses a factor of κ (the security parameter), but does not depend on the number of users or instances. (Using different building blocks, this reduction loss can even be made constant, however at a significant expense of communication complexity.)

Our approach. At a very high level, our AKE protocol follows a well-known paradigm: we use a public-key encryption scheme to transport shared keys, and a digital signature scheme to authenticate exchanged messages. Besides, we use one-time signature scheme to provide a session-specific authentication, and thus to guarantee a technical "matching conversations" property.[3] The

[3]Intuitively, the matching conversations property, introduced by Bellare and Rogaway [BR93a] establishes the notion of a "session" between two communication partners (essentially as the transcript of exchanged messages itself). Such a notion is essential in a model without explicit session identifiers (such as the one of Canetti and Krawczyk [CK01, CK02]) that separate different protocol instances.

combination of these building blocks in itself is fairly standard; the difficulty in our case is to construct suitable buildings blocks that are *tightly and adaptively* secure.

More specifically, we require, e.g., a signature scheme that is tightly secure in face of adaptive corruptions. Specifically, it should be hard for an adversary $\mathcal{A}$ to forge a new signature in the name of *any* so far uncorrupted party in the system, even though $\mathcal{A}$ may corrupt arbitrary other parties adaptively. While regular signature security applies adaptive security in this sense, this involves a (non-tight) guessing argument. In fact, currently, no adaptively tightly secure signature scheme is known: while, e.g., [HJ12a] describe a tightly secure signature scheme, their analysis does not consider adaptive corruptions, and in particular no release whatsoever of signing keys. (The situation is similar for the encryption scheme used for key transport.)

How we construct adaptively secure signatures. Hence, while we cannot directly use existing building blocks, we can use the (non-adaptively) tightly secure signature scheme of [HJ12a] as a basis to construct adaptively and tightly secure components. In a nutshell, our first (less efficient but easier-to-describe) scheme adapts the "double encryption" technique of Naor and Yung [NY90] to the signature setting. A little more concretely, our scheme uses two copies of an underlying signature scheme SIG (that has to be tightly secure, but not necessarily against adaptive corruptions). A public key in our scheme consists of two public keys $\mathsf{pk}_1, \mathsf{sk}_2$ of SIG; however, our secret key consists only of one (randomly chosen) secret key sk_b of SIG. Signatures are (non-interactive, witness-indistinguishable) proofs of knowledge of *one* signature σ_i under one sk_i.

During the security proof, the simulation will know one valid secret key sk_b for each scheme instance.[4] This allows to plausibly reveal secret keys upon corruptions. However, the witness-indistinguishability of the employed proof system will hide *which* of the two possible keys sk_i are known for each user until that user is corrupted. Hence, an adversary $\mathcal{A}$ who forges a signature for an uncorrupted user will (with probability about 1/2) forge a signature under a secret key which is unknown to the simulation. Hence, the simulation will lose only about a factor of 2 relative to the success probability of $\mathcal{A}$.

Of course, this requires using a suitable underlying signature scheme and proof system. For instance, the tightly secure (without corruptions) signature scheme from [HJ12a, ADK$^+$13] and the Groth-Sahai non-interactive proof system [GS08] will be suitable DLIN-based building blocks.

Efficient adaptively secure signatures. The signature scheme arising from the generic approach above is not overly efficient. Hence, we also construct a very optimized scheme that is not as modularly structured as the scheme above, but has extremely compact ciphertexts (of only 3 group elements). In a nutshell, this compact scheme uses the signature scheme that arises out of the recent almost-tightly secure MAC of [BKP14] as a basis. Instead of Groth-Sahai proofs, we use a more implicit consistency proof reminiscent of hash proof systems. Security can be based on a number of computational assumptions (including SXDH and DLIN), and the security reduction loses a factor of κ (the security parameter), independently of the number of users or generated signatures. We believe that this signature scheme can be of independent interest.

[4]This can be seen as a variation of the approach of "two-signatures" approach of [GJKW07]. Concretely, [GJKW07] construct a signature scheme in which the simulation – by cleverly programming a random oracle – knows one out of two possible signatures for each message.

Adaptively secure PKE and AKE schemes. A similar (generic) proof strategy allows to construct adaptively (chosen-plaintext) secure public-key encryption schemes using a variation of the Naor-Yung double encryption strategy [NY90]. (In this case, the simulation will know one out of two possible decryption keys. Furthermore, because we only require chosen-plaintext security, no consistency proof will be necessary.) Combining these tightly and adaptively secure building blocks with the tightly secure one-time signature scheme from [HJ12a] finally enables the construction of a tightly secure AKE protocol. As already sketched, our signature scheme ensures authenticated channels, while our encryption scheme is used to exchange session keys. (However, to achieve perfect forward secrecy – i.e., the secrecy of finished sessions upon corruption –, we generate PKE instances freshly for each new session.)

Notation. The symbol $\emptyset$ denotes the empty set. Let $[n] := \{1, 2, \ldots, n\} \subset \mathbb{N}$ and let $[n]^0 := [n] \cup \{0\}$. If A is a set, then $a \stackrel{\$}{\leftarrow} A$ denotes the action of sampling a uniformly random element from A. If A is a probabilistic algorithm, then we denote by $a \stackrel{\$}{\leftarrow} A$ that a is output by A using fresh random coins. If an algorithm A has black-box access to an algorithm $\mathcal{O}$, we will write $A^{\mathcal{O}}$.

2 Digital Signatures in the Multi-User Setting with Corruptions

In this section we define digital signature schemes and their security in the multi-user setting. Our strongest definition will be *existential unforgeability under adaptive chosen-message attacks in the multi-user setting with adaptive corruptions*. We show how to construct a signature scheme with tight security proof, based on a combination of a non-interactive witness indistinguishable proof of knowledge with a signature scheme with weaker security properties.

2.1 Basic Definitions

Definition 1. A (one-time) signature scheme SIG consists of four probabilistic algorithms:

- $\Pi \stackrel{\$}{\leftarrow} \mathsf{SIG.Setup}(1^\kappa)$: The parameter generation algorithm on input a security parameter 1^κ returns public parameters Π, defining the message space $\mathcal{M}$, signature space $\mathcal{S}$, and key space $\mathcal{VK} \times \mathcal{SK}$.
- $\mathsf{SIG.Gen}(\Pi)$: On input Π the key generation algorithm ouputs a pair $(vk, sk) \in \mathcal{VK} \times \mathcal{SK}$.
- $\mathsf{SIG.Sign}(sk, m)$: On input a private key sk and a message $m \in \mathcal{M}$, the signing algorithm outputs a signature σ.
- $\mathsf{SIG.Vfy}(vk, m, \sigma)$: On input a verification key vk, a message m, and a purported signature σ, the verification algorithm returns $b \in \{0, 1\}$.

Throughout the paper, we will assume signature schemes with message space $\{0, 1\}^*$ for simplicity. It is well-known that such a scheme can be constructed from a signature scheme with arbitrary message space $\mathcal{M}$ by applying a collision-resistant hash function $H : \{0, 1\}^* \to \mathcal{M}$ to the message before signing.

Security Definitions. The standard security notion for signature schemes in the single user setting is *existential unforgeability under chosen-message attacks*, as proposed by Goldwasser, Micali and Rivest [GMR88]. We consider natural extensions of this notion to the multi-user setting with or without adaptive corruptions.

Consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which is parametrized by the number of public keys μ.

1. For each $i \in [\mu]$, $\mathcal{C}$ runs $(vk^{(i)}, sk^{(i)}) \leftarrow \mathsf{SIG.Gen}(\Pi)$, where Π are public parameters. Furthermore, the challenger initializes a set $\mathcal{S}^{\mathsf{corr}}$ to keep track of corrupted keys, and μ sets $\mathcal{S}_1, \ldots, \mathcal{S}_\mu$, to keep track of chosen-message queries. All sets are initially empty. Then it outputs $(vk^{(1)}, \ldots, vk^{(\mu)})$ to $\mathcal{A}$.
2. $\mathcal{A}$ may now issue two different types of queries. When $\mathcal{A}$ outputs an index $i \in [\mu]$, then $\mathcal{C}$ updates $\mathcal{S}^{\mathsf{corr}} := \mathcal{S}^{\mathsf{corr}} \cup \{i\}$ and returns sk_i. When $\mathcal{A}$ outputs a tuple (m, i), then $\mathcal{C}$ computes $\sigma := \mathsf{SIG.Sign}(sk_i, m)$, adds (m, σ) to $\mathcal{S}_i$, and responds with σ.
3. Eventually $\mathcal{A}$ outputs a triple (i^*, m^*, σ^*).

Now we can derive various security definitions from this generic experiment. We start with existential unforgeability under chosen-message attacks in the multi-user setting with corruptions.

Definition 2. Let $\mathcal{A}$ be an algorithm that runs in time t. We say that $\mathcal{A}$ (t, ϵ, μ)-breaks the MU-EUF-CMA$^{\mathsf{Corr}}$-security of SIG, if in the above game it holds that

$$\Pr\left[\mathcal{A}^{\mathcal{C}} = (m^*, i^*, \sigma^*) : i^* \notin \mathcal{S}^{\mathsf{corr}} \wedge (m^*, \cdot) \notin \mathcal{S}_{i^*} \wedge \mathsf{SIG.Vfy}(vk^{(i^*)}, m^*, \sigma^*) = 1\right] \geq \epsilon$$

In order to construct an MU-EUF-CMA$^{\mathsf{Corr}}$-secure signature scheme, we will also need the following weaker definition of EUF-CMA security in the multi-user setting *without* corruptions. We note that this definition was also considered in [MS04].

Definition 3. Let $\mathcal{A}$ be an algorithm that runs in time t. We say that $\mathcal{A}$ (t, ϵ, μ)-breaks the MU-EUF-CMA-security of SIG, if in the above game it holds that

$$\Pr\left[\mathcal{A}^{\mathcal{C}} = (m^*, i^*, \sigma^*) : \mathcal{S}^{\mathsf{corr}} = \emptyset \wedge (m^*, \cdot) \notin \mathcal{S}_{i^*} \wedge \mathsf{SIG.Vfy}(vk^{(i^*)}, m^*, \sigma^*) = 1\right] \geq \epsilon$$

Note that both MU-EUF-CMA$^{\mathsf{Corr}}$ and MU-EUF-CMA security notions are polynomially equivalent to the standard (single user) EUF-CMA security notion for digital signatures. However, the reduction is not tight.

Finally, we need *strong* existential unforgeability in the multi-user setting without corruptions for *one-time signatures.*

Definition 4. Let $\mathcal{A}$ be an algorithm that runs in time t. We say that $\mathcal{A}$ (t, ϵ, μ)-breaks the MU-sEUF-1-CMA-security of SIG, if in the above it game holds that

$$\Pr\left[\mathcal{A}^{\mathcal{C}} = (m^*, i^*, \sigma^*) : \mathcal{S}^{\mathsf{corr}} = \emptyset \wedge |\mathcal{S}_i| \leq 1, \forall i \wedge (m^*, \sigma^*) \notin \mathcal{S}_{i^*} \wedge \mathsf{SIG.Vfy}(vk^{(i^*)}, m^*, \sigma^*) = 1\right] \geq \epsilon$$

2.2 MU-EUF-CMA$^{\mathsf{Corr}}$-Secure Signatures from General Assumptions

In this section we give a generic construction of a MU-EUF-CMA$^{\mathsf{Corr}}$-secure signature scheme, based on a MU-EUF-CMA-signature scheme and a non-interactive witness-indistinguishable proof of knowledge that allows a tight security proof. The main purpose of this construction is to resolve the "paradox" explained in the introduction.

2.2.1 NIWI Proofs of Knowledge

Let R be a binary relation. If $(x, w) \in R$, then we call x the *statement* and w the *witness.* R defines a language $\mathcal{L}_R := \{x : \exists w : (x, w) \in R\}$. A *non-interactive proof system* $\mathsf{NIPS} = (\mathsf{NIPS.Gen}, \mathsf{NIPS.Prove}, \mathsf{NIPS.Vfy})$ for R consists of the following algorithms.

- Algorithm $\mathsf{NIPS.Gen}$ takes as input the security parameter and ouputs a *common reference string* $\mathsf{CRS} \overset{\$}{\leftarrow} \mathsf{NIPS.Gen}(1^\kappa)$.
- Algorithm $\mathsf{NIPS.Prove}$ takes as input the CRS, a statement x and a witness w, and outputs a proof $\pi \overset{\$}{\leftarrow} \mathsf{NIPS.Prove}(\mathsf{CRS}, x, w)$.
- The verification algorithm $\mathsf{NIPS.Vfy}(\mathsf{CRS}, x, \pi) \in \{0, 1\}$ takes as input the CRS, a statement x, and a purported proof π. It outputs 1 if the proof is accepted, and 0 otherwise.

Definition 5. We call NIPS a *witness indistinguishable proof of knowledge* (NIWI-PoK) for R, if the following conditions are satisfied:

Perfect completeness. For all $(x, w) \in R$, $\kappa \in \mathbb{N}$, $\mathsf{CRS} \overset{\$}{\leftarrow} \mathsf{NIPS.Gen}(1^\kappa)$, and all proofs π computed as $\pi \overset{\$}{\leftarrow} \mathsf{NIPS.Prove}(\mathsf{CRS}, x, w)$ holds that

$$\Pr[\mathsf{NIPS.Vfy}(\mathsf{CRS}, x, \pi) = 1] = 1$$

Perfect Witness Indistinguishability. For all $\mathsf{CRS} \overset{\$}{\leftarrow} \mathsf{NIPS.Gen}(1^\kappa)$, for all (x, w_0, w_1) such that $(x, w_0) \in R$ and $(x, w_1) \in R$, and all algorithms $\mathcal{A}$ it holds that

$$\Pr[\mathcal{A}(\pi_0) = 1] = \Pr[\mathcal{A}(\pi_1) = 1] \tag{1}$$

where $\pi_0 \overset{\$}{\leftarrow} \mathsf{NIPS.Prove}(\mathsf{CRS}, x, w_0)$ and $\pi_1 \overset{\$}{\leftarrow} \mathsf{NIPS.Prove}(\mathsf{CRS}, x, w_1)$.

Simulated CRS. There exists an algorithm $\mathcal{E}_0$, which takes as input κ and outputs a simulated common reference string $\mathsf{CRS_{sim}}$ and a trapdoor τ.

Perfect Knowledge Extraction on Simulated CRS. There exists an algorithms $\mathcal{E}_1$ such that for all $(\mathsf{CRS_{sim}}, \tau) \overset{\$}{\leftarrow} \mathcal{E}_0(1^\kappa)$ and all $(\pi, x) \leftarrow \mathcal{A}$ such that $\mathsf{NIPS.Vfy}(\mathsf{CRS_{sim}}, x, \pi) = 1$

$$\Pr\Big[w \overset{\$}{\leftarrow} \mathcal{E}_1(\mathsf{CRS_{sim}}, \pi, x, \tau) : (x, w) \in R\Big] = 1$$

Security Definition for NIWI-PoK. An algorithm $(t, \epsilon_{\mathsf{CRS}})$-breaks the security of a NIWI-PoK if it runs in time t and for all $\kappa \in \mathbb{N}$, $\mathsf{CRS_{real}} \overset{\$}{\leftarrow} \mathsf{NIPS.Gen}(1^\kappa)$, all $(\mathsf{CRS_{sim}}, \tau) \overset{\$}{\leftarrow} \mathcal{E}_0(1^\kappa)$, it holds that

$$\Pr\Big[\mathcal{A}(\mathsf{CRS_{real}}) = 1)\Big] - \Pr\Big[\mathcal{A}(\mathsf{CRS_{sim}}) = 1\Big] \geq \epsilon_{\mathsf{CRS}}$$

We note that *perfect* witness indistinguishability is preserved if the algorithm $\mathcal{A}$ sees more than one proof. That is, let $\mathcal{O}_b^q(x, w_0, w_1)$ denote an oracle which takes as input (x, w_0, w_1) with $(x, w_0) \in R$ and $(x, w_1) \in R$, and outputs $\mathsf{NIPS.Prove}(\mathsf{CRS}, x, w_b)$ for random $b \in \{0, 1\}$. Consider an algorithm $\mathcal{A}$ which asks $\mathcal{O}_b^q$ at most q times. We observe the following (see Appendix B for a proof):

Lemma 1. *equation 1 implies for all $q \in \mathbb{N}$:*

$$\Pr\left[\mathcal{A}^{\mathcal{O}_1^q} = 1 : \mathsf{CRS} \overset{\$}{\leftarrow} \mathsf{NIPS.Gen}(1^\kappa)\right] = \Pr\left[\mathcal{A}^{\mathcal{O}_0^q} = 1 : \mathsf{CRS} \overset{\$}{\leftarrow} \mathsf{NIPS.Gen}(1^\kappa)\right] \tag{2}$$

2.2.2 Generic Construction

In this section we show how to construct a MU-EUF-CMA$^{\mathsf{Corr}}$-secure (Definition 2) signature scheme $\mathsf{SIG}_{\mathsf{MU}}$ from an MU-EUF-CMA-secure (Definition 3) scheme SIG and a NIWI-PoK.

In the sequel let $\mathsf{NIPS} = (\mathsf{NIPS.Gen}, \mathsf{NIPS.Prove}, \mathsf{NIPS.Vfy})$ denote a NIWI-PoK for relation

$$R := \{((\mathsf{vk}_0, \mathsf{vk}_1, m), (\sigma_0, \sigma_1)) : \mathsf{SIG.Vfy}(\mathsf{vk}_0, m, \sigma_0) = 1 \vee \mathsf{SIG.Vfy}(\mathsf{vk}_1, m, \sigma_1) = 1\}.$$

That is, R consists of statements of the form $(\mathsf{vk}_0, \mathsf{vk}_1, m)$, where $(\mathsf{vk}_0, \mathsf{vk}_1)$ are verification keys for signature scheme SIG, and m is a message. Witnesses are tuples (σ_0, σ_1) such that either σ_0 is a valid signature for m under vk_0, or σ_1 is a valid signature for m under vk_1, or both.

The new signature scheme $\mathsf{SIG}_{\mathsf{MU}} = (\mathsf{SIG.Setup}_{\mathsf{MU}}, \mathsf{SIG.Gen}_{\mathsf{MU}}, \mathsf{SIG.Sign}_{\mathsf{MU}}, \mathsf{SIG.Vfy}_{\mathsf{MU}})$ works as follows:

- $\Pi_{\mathsf{SIG}_{\mathsf{MU}}} \overset{\$}{\leftarrow} \mathsf{SIG.Setup}_{\mathsf{MU}}(1^\kappa)$: The parameter generation algorithm $\mathsf{SIG.Setup}_{\mathsf{MU}}$ runs $\Pi_{\mathsf{SIG}} \overset{\$}{\leftarrow} \mathsf{SIG.Setup}(1^\kappa)$ and $\mathsf{CRS} \overset{\$}{\leftarrow} \mathsf{NIPS.Gen}(1^\kappa)$. It outputs $\Pi_{\mathsf{SIG}_{\mathsf{MU}}} := (\Pi_{\mathsf{SIG}}, \mathsf{CRS})$.
- $\mathsf{SIG.Gen}_{\mathsf{MU}}(\Pi_{\mathsf{SIG}_{\mathsf{MU}}})$: The key generation algorithm generates two key pairs by running the key generation algorithm of SIG twice: $(\mathsf{vk}_i, \mathsf{sk}_i) \overset{\$}{\leftarrow} \mathsf{SIG.Gen}(\Pi_{\mathsf{SIG}})$, for $i \in \{0,1\}$. Then it flips a random coin $\delta \overset{\$}{\leftarrow} \{0,1\}$ and returns $(vk, sk) = \big((\mathsf{vk}_0, \mathsf{vk}_1), (\mathsf{sk}_\delta, \delta)\big)$. Observe that $\mathsf{sk}_{1-\delta}$ is discarded.
- $\mathsf{SIG.Sign}_{\mathsf{MU}}(sk, m)$: The signing algorithm generates a SIG-signature $\sigma_\delta \overset{\$}{\leftarrow} \mathsf{SIG.Sign}(\mathsf{sk}_\delta, m)$. Then it defines a witness w as

$$w := \begin{cases} (\sigma_\delta, \perp), & \text{if } \delta = 0, \\ (\perp, \sigma_\delta), & \text{if } \delta = 1, \end{cases}$$

  where $\perp$ is an arbitrary constant (e.g., a fixed element from the signature space). Note that $((\mathsf{vk}_0, \mathsf{vk}_1, m), w) \in R$. Finally it runs $\pi \overset{\$}{\leftarrow} \mathsf{NIPS.Prove}\big(\mathsf{CRS}, (\mathsf{vk}_0, \mathsf{vk}_1, m), w\big)$. The signature for message m is $\sigma := \pi$.
- $\mathsf{SIG.Vfy}_{\mathsf{MU}}(vk, m, \sigma)$: The verification algorithm parses vk as $(\mathsf{vk}_0, \mathsf{vk}_1)$ and returns whatever $\mathsf{NIPS.Vfy}\big(\mathsf{CRS}, (\mathsf{vk}_0, \mathsf{vk}_1, m), \sigma\big)$ returns.

Theorem 1. *From any attacker $\mathcal{A}_{\mathsf{SIG}_{\mathsf{MU}}}$ that (t, ϵ, μ)-breaks the* MU-EUF-CMA$^{\mathsf{Corr}}$*-security (with corruptions) of* $\mathsf{SIG}_{\mathsf{MU}}$ *there exists an algorithm $\mathcal{B} = (\mathcal{B}_{\mathsf{NIPS}}, \mathcal{B}_{\mathsf{SIG}})$ such that either $\mathcal{B}_{\mathsf{NIPS}}$ $(t_{\mathsf{CRS}}, \epsilon_{\mathsf{CRS}})$-breaks the security of NIWI-PoK or $\mathcal{B}_{\mathsf{SIG}}$ $(t_{\mathsf{SIG}}, \epsilon_{\mathsf{SIG}}, \mu)$-breaks the* MU-EUF-CMA*-security (without corruptions) of* SIG *with $t \approx t_{\mathsf{CRS}} \approx t_{\mathsf{SIG}}$ and*

$$\epsilon < 2 \cdot \epsilon_{\mathsf{SIG}} + \epsilon_{\mathsf{CRS}}$$

Proof. We proceed in a sequence of games. The first game is the real game that is played between an attacker $\mathcal{A}$ and a challenger $\mathcal{C}$, as described in Section 2.1. We say that $\mathcal{A}$ wins in Game i if it $(t, \Pr[\chi_i], \mu)$-breaks the MU-EUF-CMA$^{\mathsf{Corr}}$-security of $\mathsf{SIG}_{\mathsf{MU}}$ where the values $\Pr[\chi_i]$ are described in the respective games.

Game 0. This is the real game that is played between $\mathcal{A}$ and $\mathcal{C}$. We set

$$\Pr[\chi_0] = \epsilon.$$

Game 1. In this game we change the way keys are generated and chosen-message queries are answered by the challenger.

When generating a key pair by running $\mathsf{SIG.Gen_{MU}}$, the challenger does not discard $\mathsf{sk}_{1-\delta}$ but keeps it. However, corruption queries by the attacker are still answered by responding only with sk_δ. Therefore this change is completely oblivious to $\mathcal{A}$.

To explain the second change, recall that a $\mathsf{SIG_{MU}}$-signature in Game 0 consists of a proof $\pi \xleftarrow{\$} \mathsf{NIPS.Prove}\big(\mathsf{CRS}, (\mathsf{vk}_0, \mathsf{vk}_1, m), w\big)$, where either $w = (\sigma_\delta, \perp)$ or $w = (\perp, \sigma_\delta)$ for $\sigma_\delta \xleftarrow{\$} \mathsf{SIG.Sign}(\mathsf{sk}_\delta, m)$. In Game 1 the challenger now defines w as follows. It first computes two signatures $\sigma_0 \xleftarrow{\$} \mathsf{SIG.Sign}(\mathsf{sk}_0, m)$ and $\sigma_1 \xleftarrow{\$} \mathsf{SIG.Sign}(\mathsf{sk}_1, m)$, and sets $w := (\sigma_0, \sigma_1)$. Then it proceeds as before, by computing π as $\pi \xleftarrow{\$} \mathsf{NIPS.Prove}\big(\mathsf{CRS}, (\mathsf{vk}_0, \mathsf{vk}_1, m), w\big)$. Thus, in Game 1 *two* valid signatures are used as witnesses. Due to the *perfect* witness indistinguishability property of NIPS we have:

$$\Pr[\chi_0] = \Pr[\chi_1]$$

Game 2. This game is very similar to the previous game, except that we change the way the CRS is generated. Now, we run $(\mathsf{CRS_{sim}}, \tau) \xleftarrow{\$} \mathcal{E}_0$ and all proofs are generated with respect to $\mathsf{CRS_{sim}}$. Since the contrary would allow $\mathcal{B}_{\mathsf{NIPS}}$ to break the $(t, \epsilon_{\mathsf{CRS}})$-security of NIPS we have

$$\big|\Pr[\chi_1] - \Pr[\chi_2]\big| < \epsilon_{\mathsf{CRS}}$$

Game 3. This game is similar to Game 2 except for the following. We abort the game (and $\mathcal{A}$ loses) if the forgery (i^*, m^*, σ^*) returned by $\mathcal{A}$ satisfies $\mathsf{SIG.Vfy_{MU}}\left(vk^{(i^*)}, m^*, \sigma^*\right) = 1$, but the extractor $\mathcal{E}_1$ is not able to extract a witness (s_0, s_1) from σ^*. Due to the *perfect* knowledge extraction property of NIPS on a simulated CRS we have:

$$\Pr[\chi_2] = \Pr[\chi_3]$$

Game 4. In this game we raise event $\mathsf{abort}_{\delta^{(i^*)}}$ and abort (and $\mathcal{A}$ loses) if $\mathcal{A}$ outputs a forgery (i^*, m^*, σ^*) such that the following holds.

Given (i^*, m^*, σ^*), the challenger first runs the extractor $(s_0, s_1) \xleftarrow{\$} \mathcal{E}_1(\tau, \sigma^*)$. Then it checks whether

$$\mathsf{SIG.Vfy}\left(\mathsf{vk}^{(i^*)}_{1-\delta^{(i^*)}}, m^*, s_{1-\delta^{(i^*)}}\right) = 0.$$

Recall here that $\delta^{(i^*)}$ denotes the random bit chosen by the challenger for the generation of the long-term secret of user i^*. If this condition is satisfied, then the game is aborted. Putting it differently, the challenger aborts, if the witness $s_{1-\delta^{(i^*)}}$ is *not* a valid signature for m^* under $\mathsf{vk}^{(i^*)}_{1-\delta^{(i^*)}}$.

Since $\mathcal{A}$ is not allowed to corrupt the secret key of user i^*, and the adversary sees only proofs which use *two* valid signatures (s_0, s_1) as witnesses (cf. Game 1), the random bit $\delta^{(i^*)}$ is information-theoretically perfectly hidden from $\mathcal{A}$. Therefore, we have $\Pr[\mathsf{abort}_{\delta^{(i^*)}}] \leq 1/2$ and

$$\Pr[\chi_3] \leq 2 \cdot \Pr[\chi_4]$$

Claim 1. *For any attacker* $\mathcal{A}_{\mathsf{SIG_{MU}}}$ *that breaks the* $(t, \Pr[\chi_4], \mu)$-MU-EUF-CMA$^{\mathsf{Corr}}$-*security of* $\mathsf{SIG_{MU}}$ *in Game 4 there exists an attacker* $\mathcal{B}_{\mathsf{SIG}}$ *that breaks the* $(t_{\mathsf{SIG}}, \epsilon_{\mathsf{SIG}}, \mu)$-MU-EUF-CMA-*security of* SIG *with* $t_{\mathsf{SIG}} \approx t$ *and* $\epsilon_{\mathsf{SIG}} \geq \Pr[\chi_4]$.

Given the above claim, we can conclude the proof of Theorem 1. In summary we have $\epsilon \leq \epsilon_{\mathsf{CRS}} + 2 \cdot \epsilon_{\mathsf{SIG}}$.. □

Proof of Claim 1. Attacker $\mathcal{B}_{\mathsf{SIG}}$ simulates the challenger for an adversary $\mathcal{A}_{\mathsf{SIG}_{\mathsf{MU}}}$ in Game 4. We show that any successful forgery that is output by $\mathcal{A}_{\mathsf{SIG}_{\mathsf{MU}}}$ can be used by $\mathcal{B}_{\mathsf{SIG}}$ to win the SIG security game.

$\mathcal{B}_{\mathsf{SIG}}$ receives μ public verification keys $\mathsf{vk}^{(i)}, i \in [\mu]$, and public parameters Π_{SIG} from the SIG challenger. Next, it samples μ key pairs $(\mathsf{vk}^{(i)}, \mathsf{sk}^{(i)}) \xleftarrow{\$} \mathsf{SIG.Gen}(\Pi_{\mathsf{SIG}}), i \in \{\mu+1, \ldots, 2\mu\}$. Moreover, it chooses a random vector $\delta = (\delta^{(1)}, \ldots, \delta^{(\mu)}) \in \{0,1\}^{\mu}$. It sets

$$(vk^{(i)}, sk^{(i)}) \leftarrow \left(\left(\mathsf{vk}^{(\delta^{(i)}\mu+i)}, \mathsf{vk}^{\left((1-\delta^{(i)})\mu+i\right)}\right), \mathsf{sk}^{\mu+i}\right).$$

Note that now each $\mathsf{SIG}_{\mathsf{MU}}$-verification key contains one SIG-verification key that $\mathcal{A}_{\mathsf{SIG}}$ has obtained from its challenger, and one that was generated by $\mathcal{B}_{\mathsf{SIG}}$. $\mathcal{B}_{\mathsf{SIG}}$ furthermore generates a "simulated" CRS for the NIWI-PoK along with a trapdoor by running $(\mathsf{CRS}_{\mathsf{sim}}, \tau) \xleftarrow{\$} \mathcal{E}_0$. $\mathcal{A}_{\mathsf{SIG}_{\mathsf{MU}}}$ receives as input $\{vk^{(i)} : i \in [\mu]\}$, Π_{SIG} and CRS.

Now, when asked to sign a message m under public key $vk^{(i)}$, $\mathcal{A}_{\mathsf{SIG}}$ proceeds as follows. Let $\delta^{(i)} = 0$ without loss of generality. Then it computes $\sigma_1 = \mathsf{SIG.Sign}(\mathsf{sk}^{(\mu+i)}, m)$. Moreover it requests a signature for public key $\mathsf{vk}^{(i)}$ and message m from its SIG-challenger. Let σ_0 be the response. $\mathcal{A}_{\mathsf{SIG}}$ computes the signature for m using both signatures $w = (\sigma_0, \sigma_1)$ as witnesses. Note that this is a perfect simulation of Game 4.

If Game 4 is not aborted, then any valid forgery of $\mathcal{A}_{\mathsf{SIG}_{\mathsf{MU}}}$ can be used by $\mathcal{B}_{\mathsf{SIG}}$ as a forgery in the SIG security game. The claim follows. $\square$

2.2.3 (Somewhat Inefficient) Instantiation From Existing Building Blocks

The generic construction $\mathsf{SIG}_{\mathsf{MU}}$ above can be instantiated conveniently from existing building blocks:

- Suitable tightly secure MU-EUF-CMA-secure signature schemes can be found in [HJ12b, ADK+13] (based on the DLIN assumption in pairing-friendly groups).
- Similarly, a suitable tightly MU-sEUF-1-CMA-secure one-time signature scheme is described in [HJ12b, Section 4.2]. Its security is based on the discrete logarithm assumption.
- Finally, a compatible NIWI-PoK is given by Groth-Sahai proofs [GS08]. (In a Groth-Sahai proof system, there exist "hiding" and "binding" CRSs. These correspond to our honestly generated, resp. simulated CRSs.) The security of Groth-Sahai proofs can be based on a number of assumptions, including the DLIN assumption in pairing-friendly groups.

When used in our generic construction, this yields a signature scheme whose MU-EUF-CMA$^{\mathsf{Corr}}$ security can be tightly (i.e., with a small constant loss) reduced to the DLIN assumption in pairing-friendly groups. However, we note that the resulting scheme is not overly efficient. In particular, the scheme suffers from public keys and signatures that contain a linear – in the security parameter – number of group elements.

Thus, in the next section, we offer an optimized, significantly more efficient MU-EUF-CMA$^{\mathsf{Corr}}$-secure signature scheme.

2.3 Efficient and Almost Tightly MU-EUF-CMA$^{\mathsf{Corr}}$-Secure Signatures

In this section, we present a very efficient signature scheme whose MU-EUF-CMA$^{\mathsf{Corr}}$ security can be almost tightly (i.e., with a reduction loss that is linear in the security parameter) reduced to a number of standard assumptions in cyclic groups. In fact, we prove security under any *matrix* *assumption* [EHK$^+$13], which encompasses, e.g., the SXDH, DLIN, and k-Linear assumptions. The following definitions are taken from [BKP14].

Pairing Groups and Matrix Diffie-Hellman Assumption. Let GGen be a probabilistic polynomial time (PPT) algorithm that on input 1^κ returns a description $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, g_1, g_2, e)$ of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic groups of order q for a κ-bit prime q, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2$ is an efficiently computable (non-degenerated) bilinear map. Define $g_T := e(g_1, g_2)$, which is a generator in $\mathbb{G}_T$.

We use implicit representation of group elements as introduced in [EHK$^+$13]. For $s \in \{1, 2, T\}$ and $a \in \mathbb{Z}_q$ define $[a]_s = g_s^a \in \mathbb{G}_s$ as the *implicit representation* of a in $\mathbb{G}_s$. More generally, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_q^{n\times m}$ we define $[\mathbf{A}]_s$ as the implicit representation of $\mathbf{A}$ in $\mathbb{G}_s$:

$$[\mathbf{A}]_s := \begin{pmatrix} g_s^{a_{11}} & \dots & g_s^{a_{1m}} \\ & & \\ g_s^{a_{n1}} & \dots & g_s^{a_{nm}} \end{pmatrix} \in \mathbb{G}_s^{n\times m}$$

We will always use this implicit notation of elements in $\mathbb{G}_s$, i.e., we let $[a]_s \in \mathbb{G}_s$ be an element in $\mathbb{G}_s$. Note that from $[a]_s \in \mathbb{G}_s$ it is generally hard to compute the value a (discrete logarithm problem in $\mathbb{G}_s$). Further, from $[b]_T \in \mathbb{G}_T$ it is hard to compute the value $[b]_1 \in \mathbb{G}_1$ and $[b]_2 \in \mathbb{G}_2$ (pairing inversion problem). Obviously, given $[a]_s \in \mathbb{G}_s$ and a scalar $x \in \mathbb{Z}_q$, one can efficiently compute $[ax]_s \in \mathbb{G}_s$. Further, given $[a]_1, [a]_2$ one can efficiently compute $[ab]_T$ using the pairing e. For $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^k$ define $e([\mathbf{a}]_1, [\mathbf{b}]_2) := [\mathbf{a}^\top \mathbf{b}]_T \in \mathbb{G}_T$.

We recall the definition of the Matrix Diffie-Hellman (MDDH) assumption [EHK$^+$13].

Definition 6 (Matrix Distribution)**.** Let $k \in \mathbb{N}$. We call $\mathcal{D}_k$ a matrix distribution if it outputs matrices in $\mathbb{Z}_q^{(k+1)\times k}$ of full rank k in polynomial time.

For $\mathbf{B} \in \mathbb{Z}_q^{(k+1)\times n}$, we define $\overline{\mathbf{B}} \in \mathbb{Z}_q^{k\times n}$ as the first k rows of $\mathbf{B}$ and $\underline{\mathbf{B}} \in Z_q^{1\times n}$ as the last row vector of $\mathbf{B}$. Without loss of generality, we assume the first k rows $\overline{\mathbf{A}}$ of $\mathbf{A} \overset{\$}{\leftarrow} \mathcal{D}_k$ form an invertible matrix.

The $\mathcal{D}_k$-Matrix Diffie-Hellman problem is to distinguish the two distributions $([\mathbf{A}], [\mathbf{Aw}])$ and $([\mathbf{A}], [\mathbf{u}])$ where $\mathbf{A} \overset{\$}{\leftarrow} \mathcal{D}_k$, $\mathbf{w} \overset{\$}{\leftarrow} \mathbb{Z}_q^k$ and $\mathbf{u} \overset{\$}{\leftarrow} \mathbb{Z}_q^{k+1}$.

Definition 7 ($\mathcal{D}_k$-Matrix Diffie-Hellman Assumption $\mathcal{D}_k$-MDDH)**.** Let $\mathcal{D}_k$ be a matrix distribution and $s \in \{1, 2, T\}$. We say that $\mathcal{D}$ (ϵ, t)-breaks the $\mathcal{D}_k$-Matrix Diffie-Hellman ($\mathcal{D}_k$-MDDH) Assumption relative to GGen in group $\mathbb{G}_s$ if for alladversaries $\mathcal{D}$ running in time at most t,

$$\Pr[\mathcal{D}(\mathcal{G}, [\mathbf{A}]_s, [\mathbf{Aw}]_s) = 1] - \Pr[\mathcal{D}(\mathcal{G}, [\mathbf{A}]_s, [\mathbf{u}]_s) = 1]| \leq \epsilon,$$

where the probability is taken over $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda)$, $\mathbf{A} \leftarrow \mathcal{D}_k, \mathbf{w} \overset{\$}{\leftarrow} \mathbb{Z}_q^k, \mathbf{u} \overset{\$}{\leftarrow} \mathbb{Z}_q^{k+1}$.

For each $k \geq 1$, [EHK$^+$13] specifies distributions $\mathcal{L}_k$, $\mathcal{C}_k$, $\mathcal{SC}_k$, $\mathcal{IL}_k$ such that the corresponding $\mathcal{D}_k$-MDDH assumption is the k-Linear assumption, the k-Cascade, the k-Symmetric Cascade, and the Incremental k-Linear Assumption, respectively. All assumptions a generically secure in bilinear groups and form a hierarchy of increasingly weaker assumptions. The distributions are exemplified for $k = 2$, where $a, a_1, a_2 \overset{\$}{\leftarrow} \mathbb{Z}_q$.

$$\mathcal{C}_2 : \mathbf{A} = \begin{pmatrix} a_1 & 0 \\ 1 & a_2 \\ 0 & 1 \end{pmatrix} \quad \mathcal{SC}_2 : \mathbf{A} = \begin{pmatrix} a & 0 \\ 1 & a \\ 0 & 1 \end{pmatrix} \quad \mathcal{L}_2 : \mathbf{A} = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{pmatrix} \quad \mathcal{IL}_2 : \mathbf{A} = \begin{pmatrix} a & 0 \\ 0 & a+1 \\ 1 & 1 \end{pmatrix},$$

It was also shown in [EHK$^+$13] that $\mathcal{U}_k$-MDDH (where $\mathcal{U}_k$ is the uniform distribution over $\mathbb{Z}_q^{(k+1)\times k}$) is implied by all other $\mathcal{D}_k$-MDDH assumptions. If $\mathbf{A}$ is chosen from $\mathcal{SC}_k$, then $[\mathbf{A}]_s$ can be represented with 1 group element; if $\mathbf{A}$ is chosen from $\mathcal{L}_k$ or $\mathcal{C}_k$, then $[\mathbf{A}]_s$ can be represented with k group elements; If $\mathbf{A}$ is chosen from $\mathcal{U}_k$, then $[\mathbf{A}]_s$ can be represented with $(k+1)k$ group elements. Hence, $\mathcal{SC}_k$-MDDH and $\mathcal{IL}_k$-MDDH offer the same security guarantees as k-Linear, while having the advantage of a more compact representation.

2.3.1 The Construction and its Security

Let GGen be a pairing group generator and let $\mathcal{D}_k$ be a matrix distribution. The new signature scheme $\mathsf{SIG_C} = (\mathsf{SIG.Setup_C}, \mathsf{SIG.Gen_C}, \mathsf{SIG.Sign_C}, \mathsf{SIG.Vfy_C})$ for message $m \in \{0,1\}^\ell$ works as follows:

- $\Pi \overset{\$}{\leftarrow} \mathsf{SIG.Setup_C}(1^\kappa)$: The parameter generation algorithm $\mathsf{SIG.Setup_C}$ runs $\mathcal{G} \overset{\$}{\leftarrow} \mathsf{GGen}$, $\mathbf{A}, \mathbf{A}' \overset{\$}{\leftarrow} \mathcal{D}_k$ and defines $\mathbf{B} := \overline{\mathbf{A}'} \in \mathbb{Z}_q^{k\times k}$, the $k \times k$ matrix consisting of the k top rows of $\mathbf{A}'$. For $0 \leq i \leq \ell, 0 \leq b \leq 1$ it picks $\mathbf{x}_{i,b} \overset{\$}{\leftarrow} \mathbb{Z}_q^k$, $\mathbf{Y}_{i,b} \overset{\$}{\leftarrow} \mathbb{Z}_q^{k\times k}$, and defines $\mathbf{Z}_{i,b} = (\mathbf{Y}_{i,b}^\top || \mathbf{x}_{i,b}) \cdot \mathbf{A} \in \mathbb{Z}_q^{k\times k}$. It outputs

$$\Pi := \big(\mathcal{G}, [\mathbf{A}]_1, [\mathbf{B}]_2, ([\mathbf{Z}_{i,b}]_1, [\mathbf{x}_{i,b}^\top \mathbf{B}]_2, [\mathbf{Y}_{i,b}\mathbf{B}]_2)_{1\leq i\leq \ell, 0\leq b\leq 1}\big).$$

  For a message $m = (m_1, \ldots, m_\ell) \in \{0,1\}^\ell$, define the following functions

$$\begin{aligned} \mathbf{x}(m) &:= \sum_{i=1}^{\ell} \mathbf{x}_{i,m_i}^\top \in \mathbb{Z}_q^{1\times k}, \quad \mathbf{Y}(m) := \sum_{i=1}^{\ell} \mathbf{Y}_{i,m_i} \in \mathbb{Z}_q^{k\times k}, \\ \mathbf{Z}(m) &:= \sum_{i=1}^{\ell} \mathbf{Z}_{i,m_i} = (\mathbf{Y}(m)^\top || \mathbf{x}(m)^\top) \cdot \mathbf{A} \in \mathbb{Z}_q^{k\times k}. \end{aligned} \tag{3}$$

- $\mathsf{SIG.Gen_C}(\Pi)$: The key generation algorithm picks $a \overset{\$}{\leftarrow} \mathbb{Z}_q$, $\mathbf{b} \overset{\$}{\leftarrow} \mathbb{Z}_q^k$, and defines $\mathbf{c} = (\mathbf{b}^\top || a) \cdot \mathbf{A} \in \mathbb{Z}_q^{1\times k}$. It returns $(\mathsf{vk}, \mathsf{sk}) = \big([\mathbf{c}]_1, ([a]_2, [\mathbf{b}]_2)\big) \in \mathbb{G}_1^k \times \mathbb{G}_2^{k+1}$.
- $\mathsf{SIG.Sign_C}(\Pi, \mathsf{sk}, m)$: The signing algorithm parses sk as $\mathsf{sk} = ([a]_2, [\mathbf{b}]_2)$. Next, it picks $\mathbf{r}' \overset{\$}{\leftarrow} \mathbb{Z}_q^k$ and defines

$$\mathbf{r} := \mathbf{B} \cdot \mathbf{r}' \in \mathbb{Z}_q^k, \quad u = a + \mathbf{x}(m) \cdot \mathbf{r} \in \mathbb{Z}_q, \quad \mathbf{v} = \mathbf{b} + \mathbf{Y}(m) \cdot \mathbf{r} \in \mathbb{Z}_q^k. \tag{4}$$

  The signature for message m is $\sigma := ([\mathbf{r}]_2, [u]_2, [\mathbf{v}]_2) \in \mathbb{G}_2^{2k+1}$. Note that $[u]_2, [\mathbf{v}]_2$ can be computed from $\mathbf{r}'$ and Π.
- $\mathsf{SIG.Vfy_C}(\Pi, \mathsf{vk} = [\mathbf{c}]_1, m, \sigma = ([\mathbf{r}]_2, [u]_2, [\mathbf{v}]_2))$: The verification algorithm picks $\mathbf{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^k$ and returns 1 iff the equation

$$e([\mathbf{c}\cdot\mathbf{s}]_1, [1]_2) = e([\mathbf{A}\cdot\mathbf{s}]_1, \begin{bmatrix}\mathbf{v}\\ \mathbf{u}\end{bmatrix}_2) \cdot e([\mathbf{Z}(m)\cdot\mathbf{s}]_1, [\mathbf{r}]_2)^{-1} \tag{5}$$

  holds, where $e([\mathbf{z}]_1, [\mathbf{z}']_2) := [\mathbf{z}^\top \cdot \mathbf{z}']_T$.

Instantiated under the SXDH assumption (i.e., $k = 1$ and DDH in $\mathbb{G}_1$ and $\mathbb{G}_2$) we obtain a signature scheme with $|\mathsf{vk}| = 1 \times \mathbb{G}_1$ and $|\sigma| = 3 \times \mathbb{G}_2$. Instantiated under the k-Lin assumption, we obtain a signature scheme with $\mathsf{vk}| = k \times \mathbb{G}_1$ and $|\sigma| = (2k+1) \times \mathbb{G}_2$. In both cases the public parameters contain ℓk^2 group elements.

Theorem 2. *For any attacker $\mathcal{A}$ that (t, ϵ, μ)-breaks the* MU-EUF-CMA$^{\mathsf{Corr}}$*-security of* $\mathsf{SIG_C}$*, there exists an algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ such that $\mathcal{B}_1$ (t_1, ϵ_1)-breaks the $\mathcal{D}_k$-*MDDH *assumption in $\mathbb{G}_1$, and $\mathcal{B}_2$ (t_2, ϵ_2)-breaks the $\mathcal{D}_k$-*MDDH *assumption in $\mathbb{G}_2$, where $t \approx t_1 \approx t_2$ and $\epsilon < \epsilon_1 + 2\ell\epsilon_2 + 2/q$.*

Due to space limitations, the security proof is deferred to Appendix C.

3 KEMs in the Multi-User Setting with Corruptions

In this section we will describe a generic construction of a KEM with tight MU-IND-CPA$^{\mathsf{Corr}}$-security proof, based on any public-key encryption scheme with tight security proof in the multi-user setting *without* corruptions. Encryption schemes with the latter property were described in [BBM00, HJ12a]. In particular, a tight security proof for the DLIN-based scheme from [BBS04] is given in [HJ12a]. Due to space limitations, we have to refer to Appendix D for the standard definitions for public-key encryption and KEMs.

3.1 Secure Public-Key Encryption in the Multi-User setting

The standard security notions for public key encryption in the multi-user setting (without corruptions) go back to Bellare, Boldyreva and Micali [BBM00]. Security is formalized by a game that is played between an attacker $\mathcal{A}$ and a challenger $\mathcal{C}$.

1. $\mathcal{C}$ runs $\Pi \xleftarrow{\$} \mathsf{PKE.Setup}(1^\kappa)$, generates $\mu \cdot \ell$ key pairs $(sk_i^s, pk_i^s) \xleftarrow{\$} \mathsf{PKE.KGen}(\Pi)$ for $(i, s) \in [\mu] \times [\ell]$, and chooses $b \xleftarrow{\$} \{0, 1\}$ uniformly at random.
2. $\mathcal{A}$ receives Π and $pk_1^1, \ldots, pk_\mu^\ell$, and may now adaptively query an oracle $\mathcal{O}_{\mathsf{Encrypt}}$, which takes as input (pk_i^s, m_0, m_1), computes $c \xleftarrow{\$} \mathsf{PKE.Enc}(pk_i^s, m_b)$, and responds with c.
3. Eventually $\mathcal{A}$ ouputs a bit b'.

Definition 8. We say that $\mathcal{A}$ (t, ϵ, μ, ℓ)-breaks the MU-IND-CPA security of PKE, if it runs in time t in the above security game and

$$\Pr[b' = b] \geq 1/2 + \epsilon$$

3.2 Secure KEM in the Multi-User Setting

We extend the standard indistinguishability under chosen-plaintext attacks (IND-CPA) security for KEMs to a multi-user setting with $\mu \geq 1$ public keys and adaptive corruptions of secret keys. We will refer to this new notion as MU-IND-CPA$^{\mathsf{Corr}}$-security.

Consider the following game played between a challenger $\mathcal{C}$ and an attacker $\mathcal{A}$.

1. At the beginning $\mathcal{C}$ generates parameters $\Pi \xleftarrow{\$} \mathsf{KEM.Setup}(1^\kappa)$. Then, for each $(i, s) \in [\mu] \times [\ell]$, it generates a key pair $(sk_i^s, pk_i^s) \xleftarrow{\$} \mathsf{KEM.Gen}(\Pi)$ and chooses an independently random bit $b_i^s \xleftarrow{\$} \{0, 1\}$. Finally, the challenger initializes a set $\mathcal{S}^{\mathsf{corr}} := \emptyset$ to keep track of corrupted keys. The attacker receives as input $(pk_1^1, \ldots, pk_\mu^\ell)$.
2. Now the attacker may adaptively query two oracles. $\mathcal{O}_{\mathsf{Corrupt}}$ takes as input a public key pk_i^s. It appends (i, s) to $\mathcal{S}^{\mathsf{corr}}$ and responds with sk_i^s. Oracle $\mathcal{O}_{\mathsf{Encap}}$ takes as input a public key pk_i^s. It generates a ciphertext-key-pair as $(C_i^s, K_{i,1}^s) \xleftarrow{\$} \mathsf{KEM.Encap}(pk_i^s)$ and chooses a random key $K_{i,0}^s$. It responds with $(C_i^s, K_{i,b_i^s}^s)$.
3. Finally, the attacker outputs a pair (i, s, b).

Definition 9 (MU-IND-CPA$^{\mathsf{Corr}}$-security)**.** Algorithm $\mathcal{A}$ (t, ϵ, μ, ℓ)-breaks the MU-IND-CPA$^{\mathsf{Corr}}$-security of the KEM, if it runs in time at most t and it holds that

$$\Pr\left[b_i^s = b \wedge (i,s) \notin \mathcal{S}^{\mathsf{corr}}\right] \geq 1/2 + \epsilon$$

Remark 1. It is easy to see that security in the sense of Definition 9 can efficiently be reduced to standard IND-CPA security. However, the reduction incurs a loss of $1/(\mu \cdot \ell)$. We will describe a KEM with tight security proof.

3.2.1 Generic KEM Construction

Our KEM $\mathsf{KEM}_{\mathsf{MU}}$ is based on a PKE-scheme $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.KGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$. It works as follows:

- $\Pi \xleftarrow{\$} \mathsf{KEM.Setup}_{\mathsf{MU}}(1^\kappa)$: The parameter generation algorithm $\mathsf{KEM.Setup}_{\mathsf{MU}}$ on input κ runs $\Pi_{\mathsf{PKE}} \xleftarrow{\$} \mathsf{PKE.Setup}(1^\kappa)$. The session key space $\mathcal{K}$ is set to $\mathcal{M}$, the message space of PKE that is determined by Π_{PKE}.
- $(sk, pk) \xleftarrow{\$} \mathsf{KEM.Setup}_{\mathsf{MU}}(\Pi)$: The key generation algorithm generates two keys of the PKE scheme by running $(\mathsf{sk}_i, \mathsf{pk}_i) \xleftarrow{\$} \mathsf{PKE.KGen}(\Pi)$ for $i \in \{0,1\}$. It furthermore flips a random coin $\delta \xleftarrow{\$} \{0,1\}$ and returns $(sk, pk) = \big((\mathsf{sk}_\delta, \delta), (\mathsf{pk}_0, \mathsf{pk}_1)\big)$.
- $(K, C) \xleftarrow{\$} \mathsf{KEM.Encap}_{\mathsf{MU}}(pk)$: On input $pk = (\mathsf{pk}_0, \mathsf{pk}_1)$ the encapsulation algorithm samples a random key $K \xleftarrow{\$} \mathcal{K}$, computes two ciphertexts (C_0, C_1) as $C_i \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}_i, K)$ for $i \in \{0,1\}$, sets $C := (C_0, C_1)$, and outputs (K, C).
- $K \leftarrow \mathsf{KEM.Decap}_{\mathsf{MU}}(sk, C)$: The decapsulation algorithm parses $sk = (sk_\delta, \delta)$ and $C = (C_0, C_1)$. It computes $K \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_\delta, C_\delta)$ and returns K.

Theorem 3. *For each attacker* $\mathcal{A}^{\mathsf{KEM}}$ *that* $(\epsilon_{\mathsf{kem}}, t_{\mathsf{kem}}, \mu, \ell)$*-breaks the* MU-IND-CPA$^{\mathsf{Corr}}$*-security of* $\mathsf{KEM}_{\mathsf{MU}}$ *there exists an attacker* $\mathcal{A}^{\mathsf{PKE}}$ *that* $(\epsilon_{\mathsf{pke}}, t_{\mathsf{pke}}, \mu, \ell)$*-breaks the* MU-IND-CPA*-security of* PKE *with* $t_{\mathsf{kem}} \approx t_{\mathsf{pke}}$ *and* $\epsilon_{\mathsf{kem}} \leq \epsilon_{\mathsf{pke}}$.

Due to space limitations, the proof can be found in Appendix E.

4 A Tightly-Secure AKE Protocol

In this section we construct an AKE-protocol AKE, which is based on three building blocks: a key encapsulation mechanism, a signature scheme, and a one-time signature scheme.

The protocol is a key transport protocol that needs three messages to authenticate both participants and to establish a shared session key between both parties. Informally, the key encapsulation mechanism guarantees that session keys are indistinguishable from random keys. The signature scheme is used to guarantee authentication: The long-term keys of all parties consist of verification keys of the signature scheme. Finally, the one-time signature scheme prevents oracles from accepting without having a (unique) partner oracle.

In the sequel let SIG and OTSIG be signature schemes (Definition 1) and let KEM be a key-encapsulation mechanism (Definition 10). We will assume common parameters $\Pi_{\mathsf{SIG}} \xleftarrow{\$} \mathsf{SIG.Setup}(1^\kappa)$, $\Pi_{\mathsf{OTSIG}} \xleftarrow{\$} \mathsf{OTSIG.Setup}(1^\kappa)$, and $\Pi_{\mathsf{KEM}} \xleftarrow{\$} \mathsf{KEM.Setup}(1^\kappa)$.

Long-term secrets. Each party is in possession of a key pair $(vk, sk) \xleftarrow{\$} \mathsf{SIG.Gen}(\Pi_{\mathsf{SIG}})$ for signature scheme SIG. In the sequel let $(vk^{(i)}, sk^{(i)})$ and $(vk^{(j)}, sk^{(j)})$ denote the key pairs of parties P_i, P_j, respectively.

Protocol execution. In order to establish a key, parties P_i, P_j execute the following protocol (see also Figure 1).

1. First, P_i runs $(sk^{(i)}_{\mathsf{KEM}}, pk^{(i)}_{\mathsf{KEM}}) \xleftarrow{\$} \mathsf{KEM.Gen}(\Pi_{\mathsf{KEM}})$ and $(vk^{(i)}_{\mathsf{OTS}}, sk^{(i)}_{\mathsf{OTS}}) \xleftarrow{\$} \mathsf{OTSIG.Gen}(\Pi_{\mathsf{OTSIG}})$ and computes a signature $\sigma^{(i)} := \mathsf{SIG.Sign}(sk^{(i)}, vk^{(i)}_{\mathsf{OTS}})$. It defines $\mathsf{Pid} = j$ and $m_1 := (vk^{(i)}_{\mathsf{OTS}}$, $\sigma^{(i)}$, $pk^{(i)}_{\mathsf{KEM}}$, Pid, $i)$ and transmits m_1 to P_j.
2. Upon receiving m_1, P_j parses m_1 as the tuple $(vk^{(i)}_{\mathsf{OTS}}$, $\sigma^{(i)}$, $pk^{(i)}_{\mathsf{KEM}}$, Pid, $i)$. Then it checks whether $\mathsf{Pid} = j$ and $\mathsf{SIG.Vfy}$ $(vk^{(i)}$, $vk^{(i)}_{\mathsf{OTS}}$, $\sigma^{(i)}) = 1$. If at least one of both check is not passed, then P_j outputs $\perp$ and rejects.
   Otherwise it runs $(vk^{(j)}_{\mathsf{OTS}}, sk^{(j)}_{\mathsf{OTS}}) \xleftarrow{\$} \mathsf{OTSIG.Gen}(\Pi_{\mathsf{OTSIG}})$ and $(K, C) \xleftarrow{\$} \mathsf{KEM.Encap}(pk^{(i)}_{\mathsf{KEM}})$ and computes $\sigma^{(j)} := \mathsf{SIG.Sign}(sk^{(j)}, vk^{(j)}_{\mathsf{OTS}})$. Then it sets $m_2 := (vk^{(j)}_{\mathsf{OTS}}, \sigma^{(j)}, C)$ and computes a one-time signature $\sigma^{(j)}_{\mathsf{OTS}} := \mathsf{OTSIG.Sign}(sk^{(j)}_{\mathsf{OTS}}, (m_1, m_2))$ and transmits the tuple $(m_2$, $\sigma^{(j)}_{\mathsf{OTS}})$ to P_i.
3. Upon receiving the message $(m_2$, $\sigma^{(j)}_{\mathsf{OTS}})$, P_i parses m_2 as $(vk^{(j)}_{\mathsf{OTS}}, \sigma^{(j)}, C)$ and checks whether $\mathsf{SIG.Vfy}$ $(vk^{(j)}$, $vk^{(j)}_{\mathsf{OTS}}$, $\sigma^{(j)}) = 1$ and $\mathsf{OTSIG.Vfy}$ $(vk^{(j)}_{\mathsf{OTS}}$, (m_1, m_2), $\sigma^{(j)}_{\mathsf{OTS}}) = 1$. If at least one of both check is not passed, then P_i outputs $\perp$ and rejects.
   Otherwise it computes $\sigma^{(i)}_{\mathsf{OTS}} := \mathsf{OTSIG.Sign}$ $(sk^{(i)}_{\mathsf{OTS}}$, $(m_1, m_2))$ and sends $\sigma^{(i)}_{\mathsf{OTS}}$ to P_j. Finally, P_i computes and outputs the session key $K_{i,j} := \mathsf{KEM.Decap}(sk^{(i)}_{\mathsf{KEM}}, C)$.
4. Upon receiving $\sigma^{(i)}_{\mathsf{OTS}}$, P_j checks whether $\mathsf{OTSIG.Vfy}(vk^{(i)}_{\mathsf{OTS}}$, (m_1, m_2), $\sigma^{(i)}_{\mathsf{OTS}}) = 1$. If this fails, then $\perp$ is returned. Otherwise P_j outputs the session key $K_{i,j} := K$.

Efficiency and security. See Appendix F for an efficiency analysis of this protocol. We prove security in an enhanced version of the Bellare-Rogaway [BR93a] security model, which is descibed in Appendix G. The proof of the following theorem is given in Appendix H.

Theorem 4. *If there is an attacker $\mathcal{A}_{\mathsf{AKE}}$ that $(t, \mu, \ell, \epsilon_{\mathsf{AKE}})$-breaks the security of* AKE *in the sense of Definition 14 then there is an algorithm $\mathcal{B} = (\mathcal{B}_{\mathsf{KEM}}, \mathcal{B}_{\mathsf{SIG}}, \mathcal{B}_{\mathsf{OTSIG}})$ such that either $\mathcal{B}_{\mathsf{KEM}}$ $(t', \mu \cdot \ell, \epsilon_{\mathsf{KEM}})$-breaks the* MU-IND-CPA$^{\mathsf{Corr}}$*-security of* KEM *(Definition 9), or $\mathcal{B}_{\mathsf{SIG}}$ $(t', \epsilon_{\mathsf{SIG}}, \mu)$-breaks the* MU-EUF-CMA*-security of* SIG *(Definition 2), or $\mathcal{B}_{\mathsf{OTSIG}}$ $(t', \epsilon_{\mathsf{OTSIG}}, \mu \cdot \ell)$-breaks the* MU-sEUF-1-CMA*-security of* OTSIG *(Definition 4) where $t \approx t'$ and*

$$\epsilon_{\mathsf{AKE}} \leq 4\epsilon_{\mathsf{OTSIG}} + 2\epsilon_{\mathsf{SIG}} + \epsilon_{\mathsf{KEM}}.$$

References

[ADK$^+$13] Masayuki Abe, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Tagged one-time signatures: Tight security and optimal tag size. In *PKC 2013: 16th International Workshop on Theory and Practice in Public Key Cryptography*, Lecture Notes in Computer Science, pages 312–331. Springer, 2013.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, May 2000.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, August 2004.

[Ber08] Daniel J. Bernstein. Proving tight security for Rabin-Williams signatures. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 70–87. Springer, April 2008.

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (hierarchical) identity-based encryption from affine message authentication. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 408–425, 2014.

[BR93a] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, August 1993.

[BR93b] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. pages 409–426, 2006.

[BWJM97] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In Michael Darnell, editor, *6th IMA International Conference on Cryptography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer, December 1997.

[BWM98] Simon Blake-Wilson and Alfred Menezes. Authenticated Diffie-Hellman key agreement protocols (invited talk). In Stafford E. Tavares and Henk Meijer, editors, *SAC 1998: 5th Annual International Workshop on Selected Areas in Cryptography*, volume 1556 of *Lecture Notes in Computer Science*, pages 339–361. Springer, August 1998.

[BWM99] Simon Blake-Wilson and Alfred Menezes. Unknown key-share attacks on the station-to-station (sts) protocol. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 154–170. Springer, 1999.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, May 2001.

[CK02] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351. Springer, April / May 2002.

[CW13] Jie Chen and Hoeteck Wee. Fully, (almost) tightly secure IBE and dual system groups. Lecture Notes in Computer Science, pages 435–460. Springer, August 2013.

[DA99] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999. Obsoleted by RFC 4346, updated by RFCs 3546, 5746.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DR06] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), April 2006. Obsoleted by RFC 5246, updated by RFCs 4366, 4680, 4681, 5746.

[DR08] T. Dierks and E. Rescorla. RFC 5246: The transport layer security (tls) protocol; version 1.2, August 2008.

[EHK$^+$13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. Lecture Notes in Computer Science, pages 129–147. Springer, August 2013.

[GBN09] M. Choudary Gorantla, Colin Boyd, and Juan Manuel González Nieto. Modeling key compromise impersonation attacks on group key exchange protocols. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 105–123. Springer, March 2009.

[GJKW07] Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the Diffie-Hellman problems. *Journal of Cryptology*, 20(4):493–514, October 2007.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, April 2008.

[HJ12a] Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 590–607. Springer, August 2012.

[HJ12b] Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. Cryptology ePrint Archive, Report 2012/311, 2012. `http://eprint.iacr.org/`.

[JKSS12] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 273–293. Springer, August 2012.

[JV96] Mike Just and Serge Vaudenay. Authenticated multi-party key agreement. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT'96*, volume 1163 of *Lecture Notes in Computer Science*, pages 36–49. Springer, November 1996.

[KK12] Saqib A. Kakvi and Eike Kiltz. Optimal security proofs for full domain hash, revisited. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 537–553. Springer, April 2012.

[Kra05] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, August 2005.

[LJYP14] Benoit Libert, Marc Joye, Moti Yung, and Thomas Peters. Concise multi-challenge cca-secure encryption and signatures with almost tight security. In *ASIACRYPT 2014*, 2014. `https://eprint.iacr.org/2014/743.pdf`.

[LLM07] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007: 1st International Conference on Provable Security*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16. Springer, November 2007.

[MS04] Alfred Menezes and Nigel P. Smart. Security of signature schemes in a multi-user setting. *Des. Codes Cryptography*, 33(3):261–274, 2004.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437. ACM Press, May 1990.

[Sch11] Sven Schäge. Tight proofs for signature schemes without random oracles. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 189–206. Springer, May 2011.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`.

A The Difficulty of Tightly-Secure AKE

There are two main difficulties with proving tight security of an AKE protocol, which we would like to explain with concrete examples.

To illustrate the first, let us think of an AKE protocol where the long-term key pair (pk_i, sk_i) is a key pair for a digital signature scheme. Clearly, at some point in the security proof the security of the signature scheme must be used as an argument for the security of the AKE protocol, by giving a reduction from forging a signature to breaking the AKE protocol. Note that the attacker may use the $\mathsf{Corrupt}$-query to learn the long-term secret of *all* parties, except for communication partner P_j of the Test-oracle. The index j might be chosen at random by the attacker.

A standard approach in security proofs for AKE protocols is to let the reduction, which implements the challenger in order to take advantage of the attacker, *guess* the index j of party P_j. The reduction generates all key pairs (pk_i, sk_i) with $i \neq j$ on its own, and thus is able to answer $\mathsf{Corrupt}$-queries to party P_i for all $i \neq j$. In order to use the security of the signature scheme as an argument, a challenge public-key pk^* from the security experiment of the signature scheme is embedded as $pk_j := pk^*$.

Note that this strategy works *only if* the reduction guesses the index $i \in [\ell]$ correctly, which leads to a loss factor of $1/\ell$ in the success probability of the reduction. It is not immediately clear how to avoid this guessing: a reduction that avoids it would be required to be able to reveal *all* long-term secret key at any time in the security experiment, while simultaneously it needs to use the security of the signature scheme as an argument for the security of the AKE protocol. It turns out that we can resolve this seeming paradox by combining two copies of a signature scheme with a non-interactive proof system in a way somewhat related to the Naor-Yung paradigm [NY90] for public-key encryption.

To explain the second main difficulty, let us consider signed-DH protocol as an example. Let us first sketch this protocol. We stress that we leave out many details for simplicity, to keep the discussion on an intuitive level. In the sequel let $\mathcal{G}$ be a cyclic group of order p with generator g. Two parties P_i, P_j exchange a key as follows.

1. $\mathcal{P}_i$ chooses $x \xleftarrow{\$} \mathbb{Z}_p$ at random. It computes g^x and a digital signature σ_i over g^x, and sends (g^x, σ_i) to P_j.
2. If $\mathcal{P}_j$ receives (g^x, σ_i). It verifies σ_i, chooses $y \xleftarrow{\$} \mathbb{Z}_p$ at random, computes g^y and a digital signature σ_j over g^y, and sends (g^y, σ_j) to P_i. Moreover, P_j computes the key as $K = (g^x)^y$.
3. If $\mathcal{P}_i$ receives (g^y, σ_j), and σ_j is a valid signature, then P_i computes the key as $K = (g^y)^x$.

The security of this protocol can be proved [CK01] based on the (assumed) security of the signature scheme and the hardness of the decisional Diffie-Hellman problem, which asks for given a vector $(g, g^x, g^y, g^w) \in \mathcal{G}$ to determine whether $w = xy$ or w is random. However, even though the DDH problem is randomly self-reducible [BBM00], it seems impossible to avoid guessing at least one oracle participating in the Test-session.

To explain this, consider an attacker in the AKE security model from Appendix G. Assume that the attacker asks $\mathsf{Send}(i, s, (\top, j))$ to an (uncorrupted) oracle π_i^s. According to the protocol specification, the oracle has to responds with (g^x, σ_i). At some point in the security proof the security of the protocol is reduced to the hardness of the DDH problem, thus, the challenger of the AKE security experiment has to decide whether it embeds (a part of) the given DDH-instance in g^x. Essentially, there are two options:

- The challenger decides that it embeds (a part of) the given DDH-instance in g^x. In this case, there exists an attacker which makes the simulation fail (with probability 1) if oracle π_i^s does not participate in the Test-session. This attacker proceeds as follows.
  1. It corrupts some unrelated party P_j to learn sk_j.
  2. It computes g^y for $y \xleftarrow{\$} \mathbb{Z}_p$ along with a signature σ_j under sk_j, and asks $\mathsf{Send}(i, s, (g^y, \sigma_j))$ to send (g^y, σ_j) to π_i^s.
  3. Finally it asks $\mathsf{Reveal}(i, s)$ to learn the session key k_i^s computed by π_i^s, and checks whether

$k_i^s = (g^x)^y$.

A challenger interacting with this attacker faces the problem that it needs to be able to compute $k_i^s = (g^y)^x$, *knowing neither x or y.* Note that the challenger can not answer with an incorrect k_i^s, because the attacker knows y and thus is able check whether k_i^s is computed correctly.

- The challenger decides that it does not embed (a part of) the given DDH-instance in g^x. If now the attacker asks $\mathsf{Test}(i, s)$, then the challenger is not able to take advantage of the attacker, because the DDH-challenge is not embedded in the Test-session.

The only way we see to circumvent this technical issue is to let the challenger guess in advance (at least) one oracle that participates in the Test-session, which however leads to a loss of $1/(\mu\ell)$ in the reduction.

The challenge with describing a tightly-secure AKE protocol is therefore to come up with a proof strategy that that avoids guessing. This requires to apply a strategy where essentially an instance of a hard computational problem is embedded into *any* protocol session, while at the same time the AKE-challenger is always able to compute the same keys as the attacker.

B Proof of Lemma 1

For sake of contradiction assume that there is an algorithm $\mathcal{A}$ such that

$$\Pr[\mathcal{A}_q^{\mathcal{O}_1^q} = 1] - \Pr[\mathcal{A}_q^{\mathcal{O}_0^q} = 1] = \epsilon_{\mathsf{q}} > 0.$$

We construct an algorithm $\mathcal{A}_1$ such that $\Pr[\mathcal{A}_1^{\mathcal{O}_1^1} = 1] - \Pr[\mathcal{A}_1^{\mathcal{O}_0^1} = 1] \geq \epsilon_1 = \frac{\epsilon_{\mathsf{q}}}{q+1} > 0$. This contradicts equation 1.

$\mathcal{A}_1$ acts as challenger for $\mathcal{A}_q$. We describe $q+2$ games and $\mathcal{A}_1$ will choose which of these games is to be played. We note that $\mathcal{A}_1$ may issue one proof query to a challenger $\mathcal{C}$. Let $\pi = \mathcal{C}(\cdot,\cdot,\cdot)$ denote the response and let b_1 denote the bit that was chosen by $\mathcal{C}$. In game $i, i \in [q+1]^0$, $\mathcal{A}_1$ responds to the proof queries $(x^{(j)}, w_0^{(j)}, w_1^{(j)}), j \in [q]$, of $\mathcal{A}_q$ as follows:

```
1: IF $j < i$ RETURN $\mathsf{NIPS.Prove}(\mathsf{CRS}, x^{(j)}, w_0^{(j)})$
2: IF $i = j$ RETURN $\pi = \mathcal{C}(x^{(j)}, w_0^{(j)}, w_1^{(j)})$
3: RETURN $\mathsf{NIPS.Prove}(\mathsf{CRS}, x^{(j)}, w_1^{(j)})$
```

Finally $\mathcal{A}_q$ will output a bit b_q as its guess for b. $\mathcal{A}_1$ will just forward this bit as its own guess for b_1. By $\Pr[\chi_i]$ we denote the probability that $\mathcal{A}_q$ outputs 1 in game i. Obviously, we have that game 0 is equal to $\mathcal{O}_1^q$ whereas game $q+1$ is equal to $\mathcal{O}_0^q$. Therefore we have $\Pr[\chi_0] - \Pr[\chi_{q+1}] \geq \epsilon_{\mathsf{q}}$. Now $\mathcal{A}_1$ will choose $i \in [q]^0$ uniformly at random. We note that $\mathcal{O}_1^1$ will return $\mathsf{NIPS.Prove}(\mathsf{CRS}, x, w_1)$ whereas $\mathcal{O}_0^1$ will return $\mathsf{NIPS.Prove}(\mathsf{CRS}, x, w_0)$. Therefore, we have $\Pr[\mathcal{A}_1^{\mathcal{O}_1^1}(\pi) = 1] = \frac{1}{q+1}\sum_{i=0}^{q} \Pr[\chi_i]$. We observe that game i and game $i+1$ are identical if $b_1 = 0$ in game i and $b_1 = 1$ in game $i+1$. It follows $\Pr[\mathcal{A}_1^{\mathcal{O}_0^1}(\pi) = 1] = \frac{1}{q+1}\sum_{i=0}^{q} \Pr[\chi_{i+1}]$. So:

$$\Pr[\mathcal{A}_1^{\mathcal{O}_1^1} = 1] - \Pr[\mathcal{A}_1^{\mathcal{O}_0^1} = 0] \geq \frac{1}{q+1}\epsilon_{\mathsf{q}}$$

C Proof of Theorem 2

We proceed in a sequence of games. The first game is the real MU-EUF-CMA$^{\mathsf{Corr}}$-security game that is played between an attacker $\mathcal{A}$ and a challenger $\mathcal{C}$, as described in Section 2.1. We say that $\mathcal{A}$ wins in Game i if it $(t, \Pr[\chi_i], \mu)$-breaks the MU-EUF-CMA$^{\mathsf{Corr}}$-security of $\mathsf{SIG_C}$ where the values $\Pr[\chi_i]$ are described in the respective games.

GAME 0. This is the real game that is played between $\mathcal{A}$ and $\mathcal{C}$. We use $(\mathsf{vk}_i, \mathsf{sk}_i) = \big([\mathbf{c}_i]_1, ([a_i]_2, [\mathbf{b}_i]_2)\big)$ to denote the verification/signing key of the i-th user. We have

$$\Pr[\chi_0] = \epsilon.$$

GAME 1. In this game we change the way the experiment treats forgery $\sigma^* = ([\mathbf{r}^*]_2, [u^*]_2, [\mathbf{v}^*]_2)$ for user i^* on message m^*. The experiment picks $\mathbf{s}^* \xleftarrow{\$} \mathbb{Z}_q^k$ and defines $\mathbf{t}^* = \mathbf{A} \cdot \mathbf{s}^*$. Next, it changes verification equation (5) and returns 1 iff equation

$$e([(\mathbf{b}_{i^*}^\top || a_{i^*}) \cdot \mathbf{t}^*]_1, [1]_2) = e([\mathbf{t}^*]_1, \begin{bmatrix} \mathbf{v}^* \\ u^* \end{bmatrix}_2) \cdot e([(\mathbf{Y}^\top(m^*) || \mathbf{x}(m^*)^\top) \cdot \mathbf{t}^*]_1, [\mathbf{r}^*]_2)^{-1} \tag{6}$$

holds. By equation (3) and by the definition of $\mathbf{c}_{i^*} = (\mathbf{b}_{i^*}^\top || a_{i^*}) \cdot \mathbf{A}$, equations (5) and (6) are equivalent. Hence,

$$\Pr[\chi_1] = \Pr[\chi_0].$$

GAME 2. In this game, instead of defining $\mathbf{t}^* = \mathbf{A} \cdot \mathbf{s}^*$, we pick $\mathbf{t}^* \xleftarrow{\$} \mathbb{Z}_q^{k+1}$. Clearly, there exists an adversary $\mathcal{B}_1$ such that $\mathcal{B}_1$ (t_1, ϵ_1)-breaks the $\mathcal{D}_k$-MDDH assumption in $\mathbb{G}_1$ with $t \approx t_1$ and

$$\Pr[\chi_2] - \Pr[\chi_1] = \epsilon_1.$$

GAME 3. In this game, we make a change of variables by substituting all $\mathbf{Y}_{i,b}$ and $\mathbf{b}_i$ using

$$\mathbf{Y}_{i,b}^\top = (\mathbf{Z}_{i,b} - \mathbf{x}_{i,b} \cdot \underline{\mathbf{A}})\overline{\mathbf{A}}^{-1}, \quad \mathbf{b}_i^\top = (\mathbf{c}_i - a_i \cdot \underline{\mathbf{A}})\overline{\mathbf{A}}^{-1}, \tag{7}$$

respectively. First, the public parameters Π are computed by picking $\mathbf{Z}_{i,b}$ and $\mathbf{x}_{i,b}$ at random and then defining $\mathbf{Y}_{i,b}$ using (7). Second, the verification keys vk_i for user i are computed by picking $\mathbf{c}_i$ and a_i at random and then defining $\mathbf{b}_i$ using (7).

Third, on a signing query (m, i), the values $\mathbf{r}$ and u are computed as before, but the value $\mathbf{v}$ is computed as

$$\mathbf{v}^\top = (\mathbf{r}^\top \mathbf{Z}(m) + \mathbf{c}_i - u \cdot \underline{\mathbf{A}}) \cdot \overline{\mathbf{A}}^{-1}. \tag{8}$$

Fourth, the verification query for message m^* and user i^* is answered by picking $h^* \xleftarrow{\$} \mathbb{Z}_q$ and $\overline{\mathbf{t}^*} \xleftarrow{\$} \mathbb{Z}_q^k$, defining $\underline{\mathbf{t}^*} = h^* + \underline{\mathbf{A}}\overline{\mathbf{A}}^{-1}\overline{\mathbf{t}^*}$ and changing equation (6) to

$$e([\mathbf{c}_{i^*} \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{t}^*} + a_{i^*} \cdot h^*]_1, [1]_2) = e([\mathbf{t}^*]_1, \begin{bmatrix} \mathbf{v}^* \\ \mathbf{u}^* \end{bmatrix}_2) \cdot e([\mathbf{Z}(m^*)\overline{\mathbf{A}}^{-1}\overline{\mathbf{t}^*} + \mathbf{x}(m^*)h^*]_1, [\mathbf{r}^*]_2)^{-1} \tag{9}$$

By the substitution formulas for $\mathbf{Y}_{i,b}$ and $\mathbf{b}_i$ and be the definition of h and $\overline{\mathbf{t}^*}$, equations (4) and (8) and equations (6) and (9) are equivalent. Hence,

$$\Pr[\chi_3] = \Pr[\chi_2].$$

Game 4. In this game, the answer $\sigma = ([\mathbf{r}]_2, [u]_2, [\mathbf{v}]_2)$ to a signing query (m, i) is computed differently. Concretely, the values $\mathbf{r}$ and $\mathbf{v}$ are computed as before, but the value u is chosen as $u \xleftarrow{\$} \mathbb{Z}_q$.

The remaining argument is purely information-theoretic. Note that in Game 4, the value a_{i^*} from sk_{i^*} only leaks through vk_{i^*} via $\mathbf{c}_{i^*} = (\mathbf{b}_{i^*}^\top || a_{i^*}) \cdot \mathbf{A}$. As the uniform $\mathbf{t}^* \notin \mathit{span}(\mathbf{A})$ (except with probability $1/q$) the value $(\mathbf{b}_{i^*}^\top || a_{i^*}) \cdot \mathbf{t}^*$ from (6) (which is equivalent to (9)) is uniform and independent from $\mathcal{A}$'s view. Hence,

$$\Pr[\chi_4] = 2/q.$$

The following lemma completes the proof of the Theorem. It follows [BKP14, CW13] and essentially proves that the underlying message authentication code is tightly secure in a multi-user setting with corruptions.

Lemma 2. *There exists an adversary $\mathcal{B}_2$ such that $\mathcal{B}_2$ (t_2, ϵ_2)-breaks the $\mathcal{D}_k$-MDDH assumption in $\mathbb{G}_2$ with $t \approx t_1$ and*

$$\Pr[\chi_4] - \Pr[\chi_3] \leq 2\ell\epsilon_2.$$

To prove the lemma, we define the following hybrid games H_j, $0 \leq j \leq \ell$. All variables are distributed as in Game 4. For $m \in \{0,1\}^*$, define $m_{|j}$ as the j-th prefix of m. (By definition, $m_{|0}$ is the empty string ε.) Let $\mathsf{RF}_{i,j} : \{0,1\}^j \to \mathbb{Z}_q$ be independent random functions. (For concreteness, one may think of $\mathsf{RF}_{i,0}(\varepsilon) := a_i$, the MAC secret key $\mathsf{sk}_{\mathsf{MAC}}$ of the i-th user. In each hybrid H_j, we will double the number of secret-keys used in answering the queries until each query uses an independent secret key.) In Hybrid H_j, an adversary $\mathcal{C}$ first obtains the values $[\mathbf{B}]_2$ and $([\mathbf{x}_{i,b}^\top \mathbf{B}]_2)_{i,b}$, which can be seen as the public MAC parameters Π_{MAC}. Next, adversary $\mathcal{C}$ can make an arbitrary number of tagging and corruption queries, plus one forgery query. On a tagging query called with (m, i), hybrid H_j picks a random $\mathbf{r} \in \mathbb{Z}_q^k$, computes $u = \mathsf{RF}_{i,j}(m_{|j}) + \mathbf{x}(m) \cdot \mathbf{r}$ and returns $([\mathbf{r}]_2, [u]_2)$, the MAC tag. Note that the value $\mathbf{v}$ is not provided by the oracle. On a Corrupt query called with i, the hybrid returns $[a_i]_2 = [\mathsf{RF}_{i,j}(m(i)_{|j})]_2$, where $m(i)$ is the first message for which the tagging oracle was called for with respect to user i. (We make one dummy query if $m(i)$ is undefined.) Further, user i is added to the list of corrupted users. The adversary is also allowed to make one single forgery query (i^*, m^*) for an uncorrupted user i^* which is answered with $([h^*]_1, [h^* \cdot \mathsf{RF}_{i^*,j}(m^*_{|j})]_1, [h^* \cdot \mathbf{x}(m^*)]_1)$, for $h^* \xleftarrow{\$} \mathbb{Z}_q$. Finally, hybrid H_j outputs whatever adversary $\mathcal{C}$ outputs.

Note that Game 3 can be perfectly simulated using the oracles provided by hybrid H_0. The reduction picks $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$, inputs $([\mathbf{x}_{i,b}\mathbf{B}]_2)_{i,b}$ from the hybrid game, picks $\mathbf{Z}_{i,b}$ at random, and computes $[\mathbf{Y}_{i,b}\mathbf{B}]_2$ via (7). The public verification keys $\mathsf{vk}_i = [\mathbf{c}_i]_1$ are picked at random, without knowing $\mathsf{sk}_i = ([a_i]_2, [\mathbf{b}_i]_2)$. To simulate a signing query on (m, i), the reduction queries the tagging oracle to obtain $([\mathbf{r}]_2, [u]_2)$ and computes the value $[\mathbf{v}]_2$ as in Game 3. Forgery and Corrupt queries can be simulated the same way by defining $\mathsf{RF}_{i,0}(\varepsilon) =: a_i$. Hence $\Pr[\chi_4] = \Pr[H_0 = 1]$. Similarly, $\Pr[\chi_3] = \Pr[H_\ell = 1]$ as in hybrid H_ℓ are values $\mathbf{u} = \mathsf{RF}_{i,\ell}(m) + \mathbf{x}(m) \cdot \mathbf{r}$ are uniform.

We make the following claim:

Claim 2. $|\Pr[H_{j-1} = 1] - \Pr[H_j = 1]| \leq 2\epsilon_2$*, for a suitable adversary $\mathcal{B}_2$.*

The proof of this claim essentially follows verbatim from Lemma B.3 of [BKP14]. The reduction uses the fact that the $\mathcal{D}_k$-MDDH assumption is random self-reducible. There is a multiplicative loss of 2 since the reduction has to guess m^*_j, the j-th bit of the forgery m^*.

Fix $0 \leq j \leq \ell - 1$. Let Q be the maximal number of tagging queries. Adversary $\mathcal{B}_2$ inputs a Q-fold $\mathcal{D}_k$-MDDH challenge $([\mathbf{A}']_2, [\mathbf{H}]_2) \in \mathbb{G}_2^{(k+1)\times k} \times \mathbb{G}_2^{(k+1)\times Q}$ and has to distinguish $\mathbf{H} = \mathbf{A}'\mathbf{W}$ for $\mathbf{W} \in \mathbb{Z}_q^{k\times Q}$ from $\mathbf{H} \xleftarrow{\$} \mathbb{Z}_q^{(k+1)\times Q}$. The Q-fold $\mathcal{D}_k$-MDDH assumption has been proved tightly equivalent to the $\mathcal{D}_k$-MDDH assumption in [EHK$^+$13].

Adversary $\mathcal{B}_2$ defines $\mathbf{B} := \overline{\mathbf{A}'}$ and picks a random bit α which is a guess for m_j^*, the j-th bit of m^*. We assume that this guess is correct, which happens with probability 1/2. For each user i, define the random function $\mathsf{RF}_{i,j}(\cdot)$ via

$$\mathsf{RF}_{i,j}(m_{|j}) := \begin{cases} \mathsf{RF}_{i,j-1}(m_{|j-1}) & m_j = \alpha \\ \mathsf{RF}_{i,j-1}(m_{|j-1}) + R_{i,m_{|j}} & m_j = 1-\alpha \end{cases}, \tag{10}$$

where $R_{i,m_{|j}} \xleftarrow{\$} \mathbb{Z}_q$. Let $\pi_{i,j} : \{0,1\}^j \to Q$ be arbitrary injective functions. Next, for $i = 1, \dots, \ell$, $b = 0, 1$ with $(i, b) \neq (j, 1-\alpha)$, $\mathcal{B}_2$ picks $\mathbf{x}_{i,b} \xleftarrow{\$} \mathbb{Z}_q^k$ and implicitly defines $\mathbf{x}_{j,1-\alpha}^\top \mathbf{B} := \mathbf{x}'^\top \mathbf{A}'$ for $\mathbf{x}' \xleftarrow{\$} \mathbb{Z}_q^{k+1}$. Note that $\mathbf{x}_{j,1-\alpha}$ is not known to $\mathcal{B}_2$, only $[\mathbf{x}_{j,1-\alpha}^\top \mathbf{B}]_2$. Adversary $\mathcal{B}_2$ returns the $\Pi_{\mathsf{MAC}} = ([\mathbf{B}]_2, ([\mathbf{x}_{i,b}^\top \mathbf{B}]_2)_{i,b})$.

A signing query on (i, m) is simulated as follows. We distinguish two cases. Case 1, if $m_j = \alpha$, then pick random $\mathbf{r} \in \mathbb{Z}_q^k$ and define $u = \mathsf{RF}_{i,j-1}(m_{|j-1}) + \mathbf{x}(m) \cdot \mathbf{r}$. By (10), the value u has the same distribution in H_{j_1} and H_j. Case 2, if $m_j \neq \alpha$ (i.e., only $[\mathbf{x}_{j,m_j}^\top \mathbf{B}]_2$ is known, $\mathbf{x}_{j,m_j}$ not), then pick a random $\mathbf{r}' \xleftarrow{\$} \mathbb{Z}_q^k$, define $\mathbf{r} := \mathbf{B}\mathbf{r}' + \overline{\mathbf{H}}_\beta$ and $u := \mathsf{RF}_{i,j-1}(m_{|j-1}) + \sum_{l \neq j} \mathbf{x}_{l,m_l}^\top \cdot \mathbf{r} + \mathbf{x}'^\top (\mathbf{A}'\mathbf{r}' + \mathbf{H}_\beta)$. Here $\mathbf{H}_\beta$ is the β-th column of $\mathbf{H}$ and $\beta = \pi_{i,j}(m_{|j})$. Let $\mathbf{H}_\beta = \mathbf{A}'\mathbf{W}_\beta + \mathbf{R}_\beta$, where $\mathbf{R}_\beta = 0$ or $\mathbf{R}_\beta$ is uniform. Then $\mathbf{r} = \overline{\mathbf{A}'}(\mathbf{r}' + \mathbf{W}_\beta) + \mathbf{R}_\beta$ and

$$\mathbf{x}'^\top (\mathbf{A}'\mathbf{r}' + \mathbf{H}_\beta) = \mathbf{x}'^\top \mathbf{A}'(\mathbf{r}' + \mathbf{W}_\beta + \mathbf{R}_\beta) = \mathbf{x}_{j,m_j}^\top \mathbf{B}(\mathbf{r}' + \mathbf{W}_\beta + \mathbf{R}_\beta) = \mathbf{x}_{j,m_j}^\top \mathbf{r} + \mathbf{x}'^\top \mathbf{R}_\beta$$

such that $u = \mathsf{RF}_{i,j-1}(m_{|j-1}) + \sum_l \mathbf{x}_{l,m_l}^\top \cdot \mathbf{r} + \mathbf{x}'^\top \mathbf{R}_\beta$. Hence, if $\mathbf{H}$ comes from the Q-fold MDDH distribution, then $\mathbf{R}_\beta = 0$ and u is distributed as in H_{j-1}; if $\mathbf{H}$ comes from the uniform distribution, then u is distributed as in H_j with $R_{i,m_{|j}} := \mathbf{x}'^\top \mathbf{R}_\beta$.

A verification query on (i^*, m^*, σ^*) is answered with $([h^*]_1, [h^* \cdot \mathsf{RF}_{i^*,j}(m^*_{|j})]_1, [h^* \cdot \mathbf{x}(m^*)]_1)$, for uniform h^*. Note that $\mathbf{x}(m^*)$ can be computed as all $\mathbf{x}_{l,m_l^*}$ are known to $\mathcal{B}_2$.

Finally, a Corrupt query for user i is answered with $[a_i]_2 = [\mathsf{RF}_{i,j}(m(i)_{|j})]_2$. Note that $[\mathsf{RF}_{i,j}(m_{|j})]_2$ can be computed for all m.

D Basic Definitions for Public-Key Encryption and KEMs

D.1 Public-Key Encryption

A PKE scheme consists of four algorithms $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.KGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ with the following syntax:

- $\Pi \xleftarrow{\$} \mathsf{PKE.Setup}(1^\kappa)$: The algorithm $\mathsf{PKE.Setup}$, on input the security parameter 1^κ, outputs a set, Π, of system parameters. Π determines the message space $\mathcal{M}$, the ciphertext space $\mathcal{C}$, the randomness space $\mathcal{R}$, and the key space $\mathcal{PK} \times \mathcal{SK}$.
- $(sk, pk) \xleftarrow{\$} \mathsf{PKE.KGen}(\Pi)$: This algorithm takes as input Π and outputs a key pair $(sk, pk) \in \mathcal{SK} \times \mathcal{PK}$.

- $c \xleftarrow{\$} \mathsf{PKE.Enc}(pk, m)$: This probabilistic algorithm takes as input a public key and a message $m \in \mathcal{M}$, and outputs a ciphertext $c \in \mathcal{C}$.
- $m = \mathsf{PKE.Dec}(sk, c)$: This deterministic algorithm takes as input a secret key sk and a ciphertext c, and outputs a plaintext $m \in \mathcal{M}$ or an error symbol, $\perp$.

D.2 Key Encapsulation Mechanisms

Definition 10. A key encapsulation mechanism consists of four probabilistic algorithms:

- $\Pi \xleftarrow{\$} \mathsf{KEM.Setup}(1^\kappa)$: The algorithm $\mathsf{KEM.Setup}$, on input the security parameter 1^κ, outputs public parameters Π, which determine the session key space $\mathcal{K}$, the ciphertext space $\mathcal{C}$, the randomness space $\mathcal{R}$, and key space $\mathcal{SK} \times \mathcal{PK}$.
- $(sk, pk) \xleftarrow{\$} \mathsf{KEM.Gen}(\Pi)$: This algorithm takes as input parameters Π and outputs a key pair $(sk, pk) \in \mathcal{SK} \times \mathcal{PK}$.
- $(K, C) \xleftarrow{\$} \mathsf{KEM.Encap}(pk)$ takes as input a public key pk, and outputs a ciphertext $C \in \mathcal{C}$ along with a key $K \in \mathcal{K}$.
- $K = \mathsf{KEM.Decap}(sk, C)$ takes as input a secret key sk and a ciphertext C, and outputs a key $K \in \mathcal{K}$ or an error symbol $\perp$.

We require the usual correctness properties.

E Proof of Theorem 3

Proof. Again we proceed in a sequence of games. Let χ_i denote the event that $\mathcal{A}$ outputs (i, s, b) with $b_i^s = b \wedge (i, s) \notin \mathcal{S}^{\mathsf{corr}}$ in Game i.

Game 0. This is the real game that is played between $\mathcal{A}$ and $\mathcal{C}$. Thus we have

$$\Pr[\chi_0] = 1/2 + \epsilon_{\mathsf{kem}}$$

Game 1. This game is identical to Game 0, except that now $\mathcal{C}$ changes the way how challenge ciphertexts are generated. For the moment let $C = (C_0, C_1)$ denote an arbitrary challenge KEM-ciphertext, which consists of two PKE-ciphertexts (C_0, C_1). Let K_1 denote the "real" key encapsulated in C and let K_0 be the corresponding independent random key chosen by the challenger. Let $sk = (sk_\delta, \delta)$ be the secret.

Recall that in Game 0, both ciphertexts (C_0, C_1) encrypt the "real" key K_1. In Game 1 we change this. Now C_δ will be an encryption of the "real" key K_1, while $C_{1-\delta}$ will encrypt the "random" key K_0. More precisely, the challenge ciphertext $C = (C_0, C_1)$ is computed as $C_\delta \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}_\delta, K_1)$ and $C_{1-\delta} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}_{1-\delta}, K_0)$. Except for this modification, Game 1 proceeds exactly like Game 0. In particular, note that since $\mathcal{C}$ still has knowledge of sk_δ, $\mathcal{C}$ can respond to corruption queries.

Claim 3. $|\Pr[\chi_1] - \Pr[\chi_0]| \leq \epsilon_{\mathsf{PKE}}$.

Before we prove this claim, let us finish the proof of Theorem 3

Note that since $\mathcal{A}$ is not allowed to corrupt pk_{i^*}, the choice of $\delta^{(i^*)}$ is perfectly hidden from $\mathcal{A}$. Therefore $\mathcal{A}$ receives no information (in an information-theoretic sense) about the bit b_{i^*} chosen by the challenger in Game 1. This implies that we have $\Pr[\chi_1] = 1/2$. Tracing through the sequence of games, we thus see that $\epsilon_{\mathsf{kem}} = \Pr[\chi_0] \leq \frac{1}{2} + \epsilon_{\mathsf{PKE}}$. $\square$

Proof of Claim 3. Attacker $\mathcal{A}_{\mathsf{PKE}} = \mathcal{A}_{\mathsf{PKE}}^{\mathcal{O}_{\mathsf{Encrypt}}}$ acts as a challenger for an adversary $\mathcal{A}_{\mathsf{KEM}} = \mathcal{A}_{\mathsf{KEM}}^{\mathcal{O}_{\mathsf{Encap}},\mathcal{O}_{\mathsf{Corrupt}}}$. The idea behind the proof is that $\mathcal{A}_{\mathsf{PKE}}$ defines the KEM-public keys such that each key consists of one PKE-public-key that $\mathcal{A}_{\mathsf{PKE}}$ generated on its own (which enables $\mathcal{A}_{\mathsf{PKE}}$ to answer corrupt-queries of $\mathcal{A}_{\mathsf{KEM}}$), and one PKE-public-key that $\mathcal{A}_{\mathsf{PKE}}$ has received from its challenger (which allows to reduce the KEM-security to the PKE-security).

For clarity let us in the sequel denote with $(pk^s_{i,\mathsf{KEM}}, sk^s_{i,\mathsf{KEM}})$ key pairs of the KEM, and with $(pk^s_{i,\mathsf{PKE}}, sk^s_{i,\mathsf{PKE}})$ of the PKE-scheme. Moreover, we denote with $C^s_{i,\mathsf{KEM}}$ ciphertexts of the KEM-scheme, and with $C^s_{i,\mathsf{PKE}}$ ciphertexts of the PKE-scheme.

$\mathcal{A}_{\mathsf{PKE}}$ receives as input $\mu \cdot \ell$ public keys $pk^s_{i,\mathsf{PKE}}$, $(i,s) \in [\mu] \times [\ell]$, of the PKE scheme from the PKE challenger. It generates $\mu \cdot \ell$ additional key pairs $(\hat{pk}^s_{i,\mathsf{PKE}}, \hat{sk}^s_{i,\mathsf{PKE}}) \xleftarrow{\$} \mathsf{PKE.KGen}(\Pi_{\mathsf{PKE}})$ for $(i,s) \in [\mu] \times [\ell]$, chooses a random vector $\delta = (\delta^1_1, \ldots, \delta^\ell_\mu) \in \{0,1\}^{\mu\cdot\ell}$, and sets

$$(pk^s_{i,\mathsf{KEM}}, sk^s_{i,\mathsf{KEM}}) \leftarrow \begin{cases} \left(\left(\hat{pk}^s_{i,\mathsf{PKE}}, pk^s_{i,\mathsf{PKE}}\right), (\hat{sk}^s_{i,\mathsf{PKE}}, \delta^s_i)\right), & \text{if } \delta^s_i = 0 \\ \left(\left(pk^s_{i,\mathsf{PKE}}, \hat{pk}^s_{i,\mathsf{PKE}}\right), (\hat{sk}^s_{i,\mathsf{PKE}}, \delta^s_i)\right), & \text{if } \delta^s_i = 1. \end{cases}$$

Note that now each KEM public key consists of one PKE public key obtained from the PKE-challenger and one PKE public key generated by $\mathcal{A}_{\mathsf{PKE}}$, their order depends on δ.

In the sequel let us assume $\delta^s_i = 0$, the case $\delta^s_i = 1$ is analogous. Each time $\mathcal{A}_{\mathsf{KEM}}$ asks $\mathcal{O}_{\mathsf{Encap}}(pk^s_{i,\mathsf{KEM}})$, $\mathcal{A}_{\mathsf{PKE}}$ chooses two keys K_0, K_1 at random. It computes $\hat{C}_{\mathsf{PKE}} \xleftarrow{\$} \mathsf{PKE.Enc}(\hat{pk}^s_{i,\mathsf{PKE}}, K_{i,0})$ and $C_{\mathsf{PKE}} = \mathcal{O}_{\mathsf{Encrypt}}(pk^s_{i,\mathsf{PKE}}, K_0, K_1)$, and reponds with $C_{\mathsf{KEM}} = (\hat{C}_{\mathsf{PKE}}, C_{\mathsf{PKE}})$.

If $\mathcal{A}^{\mathsf{KEM}}$ issues a corrupt-query for $pk^s_{i,\mathsf{KEM}}$, then $\mathcal{A}^{\mathsf{PKE}}$ responds with $sk^s_{i,\mathsf{KEM}}$.

When $\mathcal{A}^{\mathsf{KEM}}$ terminates, then $\mathcal{A}^{\mathsf{PKE}}$ outputs whatever $\mathcal{A}^{\mathsf{KEM}}$ returns. Observe that if $b = 0$, then the view of $\mathcal{A}^{\mathsf{KEM}}$ is identical to the view in Game 0 from the proof of Theorem 3, while if $b = 1$ then it is identical to Game 1. The claim follows. $\square$

F Efficiency Analysis of the AKE Protocol

Here, we analyze the efficiency of our AKE when implemented with the building blocks described above. The following messages are exchanged for each run of the protocol:

$$\begin{aligned} m_1 &= (\sigma_{\text{MU-EUF-CMA}^{\mathsf{Corr}}}, vk_{\mathsf{OTS}}, pk_{\mathsf{KEM}}, i, \mathsf{pid}) \\ m_2 &= (vk_{\mathsf{OTS}}, \sigma_{\text{MU-EUF-CMA}^{\mathsf{Corr}}}, C_{\mathsf{KEM}}, \sigma_{\mathsf{OTS}}) \\ m_3 &= \sigma_{\mathsf{OTS}} \end{aligned}$$

This leads to an overall communication complexity of $2 \cdot (|\sigma_{\text{MU-EUF-CMA}^{\mathsf{Corr}}}| + |\mathsf{vk}_{\mathsf{OTS}}| + |\sigma_{\mathsf{OTS}}|) + |\mathsf{pk}_{\mathsf{KEM}}| + |C_{\mathsf{KEM}}|$ (plus the size of i and pid). If we use the $\mathcal{D}_k$-MDDH-based signature scheme from Section 2.3, then $\sigma_{\text{MU-EUF-CMA}^{\mathsf{Corr}}}$ consists of $2k+1$ group elements. (For efficiency, we could set $k = 1$ to obtain an efficient SXDH-based signature with 3 group elements per signature.)

Furthermore, if we use the discrete-log-based tightly secure one-time signature scheme from [HJ12a], then $\mathsf{vk}_{\mathsf{OTS}}$ consists of two group elements, and σ_{OTS} consists of two exponents. Finally, we can base our double-encryption KEM on the $\mathcal{D}_k$-MDDH-based IND-CPA secure encryption scheme from [EHK+13]. In that case, $\mathsf{pk}_{\mathsf{KEM}}$ consists of $2 \cdot \mathrm{RE}(\mathcal{D}_k)$ group elements, where $\mathrm{RE}(\mathcal{D}_k)$ denotes the number of group elements necessary to represent one $\mathcal{D}_k$-element. (For instance, for the k-Linear assumption, we have $\mathrm{RE}(\mathcal{D}_k) = k$.) C_{KEM} consists of $2(k + 2)$ group elements. In the SXDH-case with $k = 1$, we obtain a double ElGamal KEM, which can be optimized – by reusing randomness, and using the fact that one ElGamal instance can be directly interpreted as a KEM – to $\mathsf{pk}_{\mathsf{KEM}}$ and C_{KEM} that contain only 2 group elements each.)

In total, we thus obtain a communication complexity of $2(2k+1+2)+2\cdot\mathrm{RE}(\mathcal{D}_k)+2(k+2) = 2\cdot\mathrm{RE}(\mathcal{D}_k)+6k+10$ group elements and $2\cdot 2 = 4$ exponents under the $\mathcal{D}_k$-MDDH assumption. Furthermore, with the optimizations sketched above, we can get an SXDH-based construction with a communication complexity of only $2(3+2)+2+2 = 14$ group elements and $2 \cdot 2 = 4$ exponents.

G Secure Authenticated Key-Exchange

In this section we present a formal security model for authenticated key-exchange (AKE) protocols. We follow the approach of Bellare and Rogaway [BR93a] and use oracles to model concurrent and multiple protocol executions within a party and the concept of *matching conversations* to define partnership between oracles.

Essentially our model is a strenghtened version of the AKE-security model of [JKSS12], which allows an additional $\mathsf{RegisterCorrupt}$-query. Moreover, we let the adversary issue more than one Test-query, in order to achieve tightness also in this dimension.

Execution Environment. In our security model, we consider μ parties $P_1, \ldots, P_\mu$. In order to formalize several sequential and parallel executions of an AKE protocol, each party P_i is represented by a set of ℓ oracles, $\{\pi_i^1, \ldots, \pi_i^\ell\}$, where $\ell \in \mathbb{N}$ is the maximum number of protocol executions per party.

Each oracle π_i^s has access to the long-term key pair $(sk^{(i)}, pk^{(i)})$ of party P_i and to the public keys of all other parties. Let $\mathcal{K}$ be the session key space. Each oracle π_i^s maintains a list of internal state variables that are described in the following:

- Pid_i^s stores the identity of the intended communication partner.
- $\Psi_i^s \in \{\mathtt{accept}, \mathtt{reject}\}$ is a boolean variable indicating wether oracle π_i^s succesfully completed the protocol execution.
- $k_i^s \in \mathcal{K}$ is used to store the session key that is computed by π_i^s.
- Γ_i^s is a variable that stores all messages sent and received by π_i^s in the order of appearance. We call Γ_i^s the $\mathsf{transcript}$.

For each oracle π_i^s these variables are initialized as $(\mathsf{Pid}_i^s, \Psi_i^s, k_i^s, \Gamma_i^s) = (\emptyset, \emptyset, \emptyset, \emptyset)$, where $\emptyset$ denotes the empty string. The computed session key is assigned to the variable k_i^s if and only if π_i^s reaches the $\mathtt{accept}$ state, i.e., if $\Psi_i^s = \mathtt{accept}$.

Adversarial Model. The attacker $\mathcal{A}$ interacts with these oracles through oracle queries. We consider an active attacker that has full control over the communication network, i.e., $\mathcal{A}$ can schedule all sessions between the parties, delay, drop, change or replay messages at will and inject own generated messages of its choice. This is modeled by the Send-query defined below.

To model further real world capabilites of $\mathcal{A}$, such as break-ins, we provide further types of queries. The $\mathsf{Corrupt}$-query allows the adversary to compromise the long-term key of a party. The

Reveal-query may be used to obtain the session key that was computed in a previous protocol instance. The RegisterCorrupt enables the attacker to register maliciously-generated public keys. Note that we do not require the adversary to know the corresponding secret key. The Test-query does not correspond to a real world capability of $\mathcal{A}$, but it is used to evaluate the advantage of $\mathcal{A}$ in breaking the security of the key exchange protocol.

More formally, the attacker may ask the following queries:

- $\mathsf{Send}(i, s, m)$: $\mathcal{A}$ can use this query to send any message m of its choice to oracle π_i^s. The oracle will respond according to the protocol specification and depending on its internal state. If $m = (\top, j)$ is sent to π_i^s, then π_i^s will send the first protocol message to P_j.
  If $\mathsf{Send}(i, s, m)$ is the τ-th query asked by $\mathcal{A}$, and oracle π_i^s sets variable $\Psi_i^s = \texttt{accept}$ after this query, then we say that π_i^s has τ-*accepted*.
- $\mathsf{Corrupt}(i)$: This query returns the long-term secret key sk_i of party P_i.
  If the τ-th query of $\mathcal{A}$ is $\mathsf{Corrupt}(P_i)$, then we call P_i τ-corrupted. If $\mathsf{Corrupt}(P_i)$ has never been issued by $\mathcal{A}$, then we say that party i is ∞-corrupted.
- $\mathsf{RegisterCorrupt}(P_i, pk^{(i)})$: This query allows $\mathcal{A}$ to register a new party P_i, $i > \mu$, with public key $pk^{(i)}$. If the same party P_i is already registered (either via RegisterCorrupt-query or $i \in [\mu]$), a failure symbol $\perp$ is returned to $\mathcal{A}$. Otherwise, P_i is registered, the pair $(P_i, pk^{(i)})$ is distributed to all other parties, and the symbol $\top$ is returned.
  Parties registered by this query are called *adversarially-controlled*.
  All adversarially-controlled parties are defined to be 0-corrupted.
- $\mathsf{Reveal}(i, s)$: In response to this query π_i^s returns the contents of k_i^s. Recall that we have $k_i^s \neq \emptyset$ if and only if $\Psi_i^s = \texttt{accept}$.
  If $\mathsf{Reveal}(i, s)$ is the τ-th query issued by $\mathcal{A}$, we call π_i^s τ-revealed. If $\mathsf{Reveal}(i, s)$ has never been issued by $\mathcal{A}$, then we say that party i is ∞-revealed.
- $\mathsf{Test}(i, s)$: If $\Psi_i^s \neq \texttt{accept}$, then a failure symbol $\perp$ is returned. Otherwise π_i^s flips a fair coin b_i^s, samples $k_0 \xleftarrow{\$} \mathcal{K}$ at random, sets $k_1 = k_i^s$, and returns $k_{b_i^s}$.
  If $\mathsf{Test}(i, s)$ is the τ-th query issued by $\mathcal{A}$, we call π_i^s τ-tested. If $\mathsf{Test}(i, s)$ has never been issued by $\mathcal{A}$, then we say that party i is ∞-tested.
  The attacker may ask many Test-queries to different oracles, but only once to each oracle.

Security Definitions. We recall the concept of *matching conversations* here that was first introduced by Bellare and Rogaway [BR93a]. We adopt the refinement from [JKSS12].

Recall that Γ_i^s be the transcript of oracle π_i^s. By $|\Gamma_i^s|$ we denote the number of the messages in Γ_i^s. Assume that there are two transcripts, Γ_i^s and Γ_j^t, where $|\Gamma_i^s| = w$ and $|\Gamma_j^t| = n$. We say that Γ_i^s is a *prefix* of Γ_j^t if $0 < w \leq n$ and the first w messages in transcripts Γ_i^s and Γ_j^t are identical.

Definition 11 (Matching conversations)**.** We say that π_i^s has a matching conversation to oracle π_j^t, if

- π_i^s has sent all protocol messages and Γ_j^t is a prefix of Γ_i^s, or
- π_j^t has sent all protocol messages and $\Gamma_i^s = \Gamma_j^t$.

We say that two oracles, π_i^s and π_j^t, have matching conversations if π_i^s has a matching conversation to process π_j^t and vice versa.

Definition 12 (Correctness)**.** We say that a two-party AKE protocol, Σ, is *correct*, if for any two oracles, π_i^s and π_j^t, that have matching conversations it holds that $\Psi_i^s = \Psi_j^t = \texttt{accept}$, $\mathsf{Pid}_i^s = j$ and $\mathsf{Pid}_j^t = i$ and $k_i^s = k_j^t$.

Security Game. Consider the following game that is played between an adversary, $\mathcal{A}$, and a challenger, $\mathcal{C}$, and that is parametrized by two numbers, μ (the number of honest identities) and ℓ (the maximum number of protocol executions per identity).

1. At the beginning of the game, $\mathcal{C}$ generates system parameters that are specified by the protocol and μ long-term key pairs $(sk^{(i)}, pk^{(i)}), i \in [\mu]$. Then $\mathcal{C}$ implements a collection of oracles $\{\pi_i^s : i \in [\mu], s \in [\ell]\}$. It passes to $\mathcal{A}$ all public keys, $pk^{(1)}, \ldots, pk^{(\mu)}$, and the public parameters.
2. Then the adversary may adaptively issue Send, $\mathsf{Corrupt}$, Reveal, $\mathsf{RegisterCorrupt}$ and Test queries to $\mathcal{C}$.
3. At the end of the game, $\mathcal{A}$ terminates with outputting a tuple (i, s, b') where π_i^s is an oracle and b' is its guess for b_i^s.

For a given protocol Σ by $\mathbb{G}_\Sigma(\mu, \ell)$ we denote the security game that is carried out with parameters μ, ℓ as described above and where the oracles implement protocol Σ.

Definition 13 (Freshness)**.** Oracle π_i^s is said to be τ-*fresh* if the following requirements satisfied:

- π_i^s has $\tilde{\tau}$-accepted, where $\tilde{\tau} \leq \tau$.
- π_i^s is $\hat{\tau}$-revealed, where $\hat{\tau} > \tau$.
- If there is an oracle, π_j^t, that has matching conversation to π_i^s, then π_j^t is ∞-revealed and ∞-tested.
- If $\mathsf{Pid}_i^s = j$ then P_j is $\tau^{(j)}$-corrupted with $\tau^{(j)} > \tau$ [5].

Definition 14 (AKE Security)**.** We say that an attacker (t, μ, ℓ, ϵ)-breaks the security of a two-party AKE protocol, Σ, if it runs in time t in the above security game $\mathbb{G}_\Sigma(\mu, \ell)$ and it holds that:

1. Let $\mathcal{Q}$ denote the event that there exists a τ and a τ-fresh oracle π_i^s and there is no *unique* oracle π_j^t such that π_i^s and π_j^t have matching conversations. Then $Pr[\mathcal{Q}] \geq \epsilon$, or
2. When $\mathcal{A}$ returns (i, s, b') such that $\mathsf{Test}(\pi_i^s)$ was $\mathcal{A}$s τ-th query and π_i^s is a τ-fresh oracle that is ∞-revealed throughout the security game then the probability that b' equals b_i^s is upper bounded by
$$\left|\Pr[b_i^s = b'] - 1/2\right| \geq \epsilon.$$

Remark 2. We note that, according to Definition 13, a τ-fresh oracle π_i^s or its partner oracle π_j^t may be corrupted later on. This allows us to model perfect forward secrecy [CK01]. Strictly speaking, we do not even require a τ-fresh oracle π_i^s not to be $\hat{\tau}$-corrupted, where $\hat{\tau} < \tau$. I.e., π_i^s may be τ-fresh, even if there exists a $\hat{\tau} < \tau$ such that party i is $\hat{\tau}$-corrupted. This allows us to model key compromise impersonation attacks [JV96]. Since we do not require the adversary to supply a proof of knowledge of a matching secret key when issuing a $\mathsf{RegisterCorrupt}$-query, we model also PKI-related attacks, e.g., PKI-related unknown key share attacks. Such attacks can have serious security effects [BWM99]. We note further that the adversary may issue more than one Test-query throughout the security game and, over that, that it may issue a Reveal-query, $\mathsf{Reveal}(i, s)$, even if π_i^s is tested.

H Proof of Theorem 4

We prove the security of the proposed protocol AKE using the sequence-of-games approach, following [Sho04, BR06]. The first game is the original attack game that is played between a challenger

[5] We note that for any $P_i, i > \mu$, we have $\tau^{(i)} = 0$. Therefore for any $\tau \geq 1$, the intended partner of a τ- fresh oracle must not be adversarially controlled.

and an attacker. We then describe a sequence of games where we modify the original game step by step. We show that the advantage of distinguishing between two successive games is negligible.

We prove Theorem 4 in two stages. First, we show that the AKE protocol is a secure authentication protocol except for probability ϵ_{Auth}. That is, the protocol fulfills security property 1.) of the AKE security definition Definition 14. Informally, the authentication property is guaranteed by the uniqueness of the transcript and the security of the MU-EUF-CMA secure signature scheme SIG and the security of the one-time signature scheme OTSIG. We show that for any τ and any τ-accepted oracle π_i^s with internal state $\Psi_i^s = \mathtt{accept}$ and $\mathsf{Pid}_i^s = j$ there exists an oracle, π_j^t, such that π_i^s and π_j^t have matching conversations. Otherwise the attacker $\mathcal{A}$ has forged a signature for either SIG or OTSIG.

In the next step, we show that the session key of the AKE protocol is secure except for probability ϵ_{Ind} in the sense of the Property 2.) of the AKE security Definition 14. The security of the authentication protocol guarantees that there can only be passive attackers on the test oracles, so that we can conclude the security for key indistinguishability from the security of the underlying KEM. We recall that μ denotes the number of honest identities and that ℓ denotes the maximum number of protocol executions per party. In the proof of Theorem 4, we consider the following two lemmas. Lemma 3 bounds the probability ϵ_{Auth} that an attacker breaks the authentication property of AKE and Lemma 4 bounds the probability ϵ_{Ind} that an attacker is able to distinguish real from random keys. It holds:

$$\epsilon_{\mathsf{AKE}} \leq \epsilon_{\mathsf{Auth}} + \epsilon_{\mathsf{Ind}}.$$

H.1 Authentication

Lemma 3. *For all attackers $\mathcal{A}$ that $(t, \mu, \ell, \epsilon_{\mathsf{Ind}})$-break the AKE protocol by breaking Property 1.) of Definition 14 there exists an algorithm $\mathcal{B} = (\mathcal{B}_{\mathsf{SIG}}, \mathcal{B}_{\mathsf{OTSIG}})$ such that either $\mathcal{B}_{\mathsf{SIG}}$ $(t', \mu, \epsilon_{\mathsf{SIG}})$-breaks the security of SIG or $\mathcal{B}_{\mathsf{OTSIG}}$ $(t'$, $\epsilon_{\mathsf{OTSIG}}$, $\mu\ell)$-breaks the security of OTSIG where $t \approx t'$ and*

$$\epsilon_{\mathsf{Auth}} \leq \epsilon_{\mathsf{SIG}} + 2 \cdot \epsilon_{\mathsf{OTSIG}}.$$

Proof. Let $\mathsf{break}_\delta^{(\mathsf{Auth})}$ be the event that there exists a τ and a τ-fresh oracle π_i^s that has internal state $\Psi_i^s = \mathtt{accept}$ and $\mathsf{Pid}_i^s = j$, but there is no unique oracle π_j^t such that π_i^s and π_j^t have matching conversations, in Game δ. If $\mathsf{break}_\delta^{(\mathsf{Auth})}$ occurs, we say that $\mathcal{A}$ wins in Game δ.

Game G_0. This is the original game that is played between an attacker $\mathcal{A}$ and a challenger $\mathcal{C}$, as described in Appendix G. Thus we have:

$$\Pr[\mathsf{break}_0^{(\mathsf{Auth})}] = \epsilon_{\mathsf{Auth}}$$

Game G_1. In this game, the challenger proceeds exactly like in the previous game, except that we add an abort rule. Let π_i^s be a τ-accepted oracle with internal state $\mathsf{Pid}_i^s = j$, where P_j is $\hat{\tau}$-corrupted with $\hat{\tau} > \tau$. We want to ensure that the OTSIG public key $vk_{\mathsf{OTSIG}}^{(j)}$ received by π_i^s was output by an oracle π_j^t (and not generated by the attacker).

Technically, we abort and raise the event $\mathsf{abort}_{\mathsf{SIG}}$, if the following condition holds:

- there exists a τ and a τ-fresh oracle π_i^s with internal state $\mathsf{Pid}_i^s = j$[6] and

[6]Since π_i^s is τ-fresh it holds that P_j is $\hat{\tau}$-corrupted, where $\hat{\tau} > \tau$.

- π_i^s received a signature $\sigma^{(j)}$ that satisfies $\mathsf{SIG.Vfy}(vk^{(j)}, vk_{\mathsf{OTS}}^{(j)}, \sigma^{(j)})$, but there exists no oracle π_j^t which has previously output a signature $\sigma^{(j)}$ over $vk_{\mathsf{OTS}}^{(j)}$.

Clearly we have

$$\left|\Pr[\mathsf{break}_0^{(\mathsf{Auth})}] - \Pr[\mathsf{break}_1^{(\mathsf{Auth})}]\right| \leq \Pr[\mathsf{abort}_{\mathsf{SIG}}].$$

Claim 4. $\Pr[\mathsf{abort}_{\mathsf{SIG}}] \leq \epsilon_{\mathsf{SIG}}$.

Proof. To show that $\Pr[\mathsf{abort}_{\mathsf{SIG}}] \leq \epsilon_{\mathsf{SIG}}$, we construct a signature forger $\mathcal{B}_{\mathsf{SIG}}$ that breaks the MU-EUF-CMA$^{\mathsf{Corr}}$-security of SIG.

$\mathcal{B}_{\mathsf{SIG}}$ simulates the challenger for attacker $\mathcal{A}_{\mathsf{AKE}}$ in Game 1. $\mathcal{B}_{\mathsf{SIG}}$ receives as input the public keys $\{vk_{\mathsf{SIG}}^{(i)}, i \in [\mu]\}$ from the SIG challenger. It simulates the AKE challenger for $\mathcal{A}_{\mathsf{AKE}}$ as follows: It sets $vk^{(i)}$ as public key for user i. Every time it needs to sign a message under a long-term key it lets the SIG challenger sign that message. $\mathcal{B}_{\mathsf{SIG}}$ can answer to corrupt queries made by $\mathcal{A}_{\mathsf{AKE}}$ by just forwarding them to the SIG challenger and then forwarding the response back to $\mathcal{A}_{\mathsf{AKE}}$. Except for this, $\mathcal{B}_{\mathsf{SIG}}$ acts exactly like the challenger in Game 0. We note that since $\mathcal{B}_{\mathsf{SIG}}$ has the power of answering to corrupt queries "on the fly", it does not need to guess which party will be corrupted by $\mathcal{A}_{\mathsf{AKE}}$ beforehand.

If event $\mathsf{abort}_{\mathsf{SIG}}$ occurs, this means that $\mathcal{A}_{\mathsf{AKE}}$ has issued a Send-query containing $(\sigma^{(j)}, vk_{\mathsf{OTS}}^{(j)})$, where j is not corrupted, and $(\sigma^{(j)}, vk_{\mathsf{OTS}}^{(j)})$ was not output by any oracle π_j^t. Thus, $\mathcal{B}_{\mathsf{SIG}}$ has never requested a signature for $vk_{\mathsf{OTS}}^{(j)}$ from its challenger. Therefore $\mathcal{A}_{\mathsf{SIG}}$ can use the signature $\sigma^{(j)}$ to break the MU-EUF-CMA$^{\mathsf{Corr}}$ security of the signature scheme. This implies $\Pr[\mathsf{abort}_{\mathsf{sig}}] \leq \epsilon_{\mathsf{SIG}}$. □

Game G_2. In this game, the challenger proceeds exactly like the challenger in Game 1, except that we add an abort rule. Let $\mathsf{abort}_{\mathsf{collision}}$ denote the event that two oracles, π_i^s and π_j^t, sample the same verification key, vk_{OTS}, for the one-time signature scheme. More formally, let

$$\mathsf{abort}_{\mathsf{collision}} := \left\{\exists (i,j) \in [\mu \cdot \ell]^2 : vk_{\mathsf{OTS}}^{(i)} = vk_{\mathsf{OTS}}^{(j)} \wedge i \neq j\right\}.$$

The simulator aborts if $\mathsf{abort}_{\mathsf{collision}}$ occurs and $\mathcal{A}$ loses the game. Clearly, we have

$$\left|\Pr[\mathsf{break}_1^{(\mathsf{Auth})}] - \Pr[\mathsf{break}_2^{(\mathsf{Auth})}]\right| \leq \Pr[\mathsf{abort}_{\mathsf{collision}}].$$

Claim 5. $\Pr\left[\mathsf{abort}_{\mathsf{collision}}\right] \leq \epsilon_{\mathsf{OTSIG}}$

Proof. Note that the number of OTSIG verification keys appearing in the experiment, and thus the probability of the event $\mathsf{abort}_{\mathsf{collision}}$, depends on $\mathcal{A}_{\mathsf{AKE}}$. Therefore we construct an attacker $\mathcal{B}_{\mathsf{OTSIG}}$, which runs $\mathcal{A}_{\mathsf{AKE}}$ as a subroutine by implementing the challenger for $\mathcal{A}_{\mathsf{AKE}}$ in Game 2.

$\mathcal{B}_{\mathsf{OTSIG}}$ proceeds exactly like the challenger in Game 2, except that it does not generate the verification keys $vk_{\mathsf{OTS}}^{(1)}, \ldots, vk_{\mathsf{OTS}}^{(\mu\ell)}$ used in the experiment on its own, but instead receives them from its OTSIG-challenger. With probability $\Pr\left[\mathsf{abort}_{\mathsf{collision}}\right]$ there exists $(i,j) \in [\mu \cdot \ell]^2$ such that $vk^{(i)} = vk^{(j)}$ for $i \neq j$. If this happens, then $\mathcal{B}_{\mathsf{OTSIG}}$ issues a sign query $\mathsf{sign}(m, j)$ to the OTSIG challenger, who will respond with a signature σ. Then $\mathcal{B}_{\mathsf{OTSIG}}$ outputs (i, m, σ), which is a valid forgery.

□

Game G_3. In this game, the challenger proceeds exactly like in the previous game, except that we add an abort rule. Let π_i^s be a τ-accepted oracle, for some τ, that received a one-time signature key, $vk_{\mathsf{OTS}}^{(j)}$, from an uncorrupted oracle, π_j^t. Informally, we want to make sure that if π_i^s accepts then π_j^t has previously output *the same* one-time signature $\sigma_{\mathsf{OTS}}^{(j)}$ over (m_1, m_2) that is valid under $vk_{\mathsf{OTS}}^{(j)}$. Note that in this case π_i^s confirms the "view on the transcript" of π_j^t.

Technically, we raise the event $\mathsf{abort}_{\mathsf{OTSIG}}$ and abort (and $\mathcal{A}$ loses), if the following condition holds:

- there exists a τ-fresh oracle π_i^s that has internal state $\mathsf{Pid}_i^s = j$ and
- π_i^s receives a valid one-time signature $\sigma_{\mathsf{OTS}}^{(j)}$ for (m_1, m_2) and accepts, but there exists no unique oracle, π_j^t, which has previously output $\left((m_1, m_2), \sigma_{\mathsf{OTS}}^{(j)}\right)$.

Clearly we have
$$\left|\Pr[\mathsf{break}_2^{(\mathsf{Auth})}] - \Pr[\mathsf{break}_3^{(\mathsf{Auth})}]\right| \leq \Pr[\mathsf{abort}_{\mathsf{OTSIG}}].$$

Claim 6. $\Pr[\mathsf{abort}_{\mathsf{OTSIG}}] \leq \epsilon_{\mathsf{OTSIG}}$

Proof. We construct a forger $\mathcal{B}_{\mathsf{OTSIG}}$ that $(t', \epsilon_{\mathsf{OTSIG}}, \mu \cdot \ell)$-breaks the MU-sEUF-1-CMA-security of OTSIG with $t' \approx t$ and $\epsilon_{\mathsf{OTSIG}} \geq \Pr[\mathsf{abort}_{\mathsf{OTSIG}}]$.

$\mathcal{B}_{\mathsf{OTSIG}}$ proceeds exactly like the challenger in Game 3, except that it does not generate the OTSIG verification keys on its own. Instead, it receives them from its challenger. Whenever it needs to compute a one-time signature, $\mathcal{B}_{\mathsf{OTSIG}}$ asks the OTSIG-challenger. With probability $\Pr[\mathsf{abort}_{\mathsf{OTSIG}}]$ there exists an oracle π_i^s which receives as input a valid signature $\sigma_{\mathsf{OTS}}^{(j)}$ for (m_1, m_2), but there exists no unique oracle, π_j^t, which has previously output $\left((m_1, m_2), \sigma_{\mathsf{OTS}}^{(j)}\right)$. If this happens, then $\mathcal{B}_{\mathsf{OTSIG}}$ outputs $(\sigma_{\mathsf{OTS}}^{(j)}, (m_1, m_2), j)$ to its challenger. Note that this is a valid forgery.

$\square$

Claim 7. $\Pr[\mathsf{break}_3^{(\mathsf{Auth})}] = 0$

Proof. Note that $\mathsf{break}_3^{(\mathsf{Auth})}$ occurs only if there exists a τ-fresh oracle π_i^s and there is no *unique* oracle π_j^t such that π_i^s and π_j^t have matching conversations.

Consider a τ-fresh oracle π_i^s. Due to Game 1 there exists (at least one) oracle π_j^t which has output the verification key $vk_{\mathsf{OTS}}^{(j)}$ received by π_i^s, along with a valid SIG-signature $\sigma^{(j)}$ over $vk_{\mathsf{OTS}}^{(j)}$, as otherwise the game is aborted. $vk_{\mathsf{OTS}}^{(j)}$ (and therefore also π_j^t) is unique due to Game 2.

π_i^s accepts only if it receives a valid one-time signature $\sigma_{\mathsf{OTS}}^{(j)}$ over the transcript (m_1, m_2) of messages. Due to Game 3 there must exist an oracle which has output this signature $\sigma_{\mathsf{OTS}}^{(j)}$. Since (m_1, m_2) contains $vk_{\mathsf{OTS}}^{(j)}$, this can only be π_j^t. Thus, if π_i^s accepts, then it must have a matching conversation to π_j^t. $\square$

Summing up we see that:
$$\epsilon_{\mathsf{Auth}} \leq \epsilon_{\mathsf{SIG}} + 2\epsilon_{\mathsf{OTSIG}}$$

$\square$

H.2 Key Indistinguishability

Lemma 4. *For all attackers $\mathcal{A}$ that $(t, \mu, \ell, \epsilon_{\mathsf{Ind}})$-break the* AKE *protocol by breaking Property 2.) of Definition 14 there exists an algorithm $\mathcal{B} = (\mathcal{B}_{\mathsf{KEM}}, \mathcal{B}_{\mathsf{SIG}}, \mathcal{B}_{\mathsf{OTSIG}})$ such that either $\mathcal{B}_{\mathsf{KEM}}$ $(t', \mu\ell, \epsilon_{\mathsf{KEM}})$-breaks the security of* KEM*, or $\mathcal{B}_{\mathsf{SIG}}$ $(t', \mu, \epsilon_{\mathsf{SIG}})$- breaks the security of* SIG *or $\mathcal{B}_{\mathsf{OTSIG}}$ $(t'$, $\epsilon_{\mathsf{OTSIG}}$, $\mu\ell)$-breaks the security of* OTSIG *where $t \approx t'$ and*

$$\epsilon_{\mathsf{Ind}} \leq \epsilon_{\mathsf{SIG}} + 2 \cdot \epsilon_{\mathsf{OTSIG}} + \epsilon_{\mathsf{KEM}}.$$

PROOF. Let $\mathsf{break}_\delta^{(\mathsf{Ind})}$ denote the event that $\mathcal{A}$ returns (i, s, b') in Game δ such that $b_i^s = b'$, $\mathsf{Test}(\pi_i^s)$ was the τ-th query of $\mathcal{A}$, and π_i^s is a τ-fresh oracle that is ∞-revealed throughout the security game. If event $\mathsf{break}_\delta^{(\mathsf{Ind})}$ occurs we say that $\mathcal{A}$ wins in Game δ. Let $\mathsf{Adv}_\delta := \Pr[\mathsf{break}_\delta^{(\mathsf{Ind})}] - 1/2$ denote the advantage of $\mathcal{A}$ in Game δ. Consider the following sequence of games.

GAME 0. This is the original security game. Thus we have that

$$\epsilon_{\mathsf{Ind}} = \mathsf{Adv}_0.$$

GAME 1. The challenger in this game proceeds as before, but it aborts if there exists a τ-fresh oracle π_i^s that has internal state $\Psi_i^s = \mathtt{accept}$ and $\mathsf{Pid}_i^s = j$, but there is no unique oracle π_j^t such that π_i^s and π_j^t have matching conversations. Note that this is exactly the event $\mathsf{break}_0^{(\mathsf{Auth})}$ from the proof of Lemma 3, which implies

$$\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \epsilon_{\mathsf{SIG}} + 2 \cdot \epsilon_{\mathsf{OTSIG}}.$$

Lemma 5.

$$\mathsf{Adv}_1 \leq \epsilon_{\mathsf{KEM}}$$

Before we prove this lemma, let us finish the proof of Lemma 4.
Summing up probabilities, we obtain that

$$\epsilon_{\mathsf{Ind}} \leq \epsilon_{\mathsf{SIG}} + 2 \cdot \epsilon_{\mathsf{OTSIG}} + \epsilon_{\mathsf{KEM}}$$

□

Proof of Lemma 5 We describe an attacker $\mathcal{B}_{\mathsf{KEM}}$ against the MU-IND-CPA$^{\mathsf{Corr}}$-security of KEM that acts as a challenger for an attacker $\mathcal{A}_{\mathsf{AKE}}$ in Game 2. We show that the simulation of $\mathcal{B}_{\mathsf{KEM}}$ perfectly simulates the challenger in Game 2 from the adversarys point of view. We do this in three steps:

1. First, we describe how $\mathcal{B}_{\mathsf{KEM}}$ simulates the protocol execution within each oracle, step by step.
2. Next, we show how $\mathcal{B}_{\mathsf{KEM}}$ can answer to all queries issued by $\mathcal{A}_{\mathsf{AKE}}$.
3. Finally, we show that any tuple (i, s, b) output by $\mathcal{A}_{\mathsf{AKE}}$ can be used to break the security of KEM.

$\mathcal{B}_{\mathsf{KEM}}$ gets a set $\mathcal{L}$ of $\mu\ell$ public keys, $\mathcal{L} := \mathsf{pk}_1^1, \ldots, \mathsf{pk}_\mu^\ell$. It may adaptively query oracles $\mathcal{O}_{\mathsf{Encap}}$ and $\mathcal{O}_{\mathsf{Corrupt}}$ that will respond as specified in Definition 9. By $\mathcal{L}_i^s$ we will denote the set of challenge ciphertexts, corresponding to pk_i^s. These lists will be filled throughout the simulation.

Simulating protocol execution within the oracles. $\mathcal{B}_{\mathsf{KEM}}$ generates all system parameters and long-term keys according to the protocol specification. It passes all public parameters as well as all long-term verification keys to $\mathcal{A}_{\mathsf{AKE}}$. In the following, on the right side, we will formally describe the simulation, whereas on the left side, we explain the changes made. The line numbering refers to the algorithms on the righthand side.

Step 1: $\mathcal{B}_{\mathsf{KEM}}$ deviates from the protocol description of Figure 1 in Line 3.

$\mathcal{B}_{\mathsf{KEM}}$ proceeds exactly like in Game 1, except for the following. If π_i^s needs to sample a KEM key pair $(sk_i^s, pk_i^s) \xleftarrow{\$} \mathsf{KEM.Gen}()$, $\mathcal{B}_{\mathsf{KEM}}$ sets $pk_i^s \leftarrow \mathsf{pk}_i^s$ (recall that $\mathcal{B}_{\mathsf{KEM}}$ received $\mu\ell$ public keys $\mathsf{pk}_1^1, \ldots, \mathsf{pk}_\mu^\ell$ from its challenger). $sk_{i,\mathsf{KEM}}^s$ is not assigned.

```
1: $(vk_{i,\mathsf{OTS}}^s, sk_{i,\mathsf{OTS}}^s) \xleftarrow{\$} \mathsf{OTS.Gen}(\Pi_{\mathsf{OTS}})$
2: $\sigma^{(i)} \xleftarrow{\$} \mathsf{SIG.Sign}(sk^{(i)}, vk_{i,\mathsf{OTS}}^s)$
3: $(sk_i^s, pk_i^s) \leftarrow (\emptyset, \mathsf{pk}_i^s)$
4: $\mathsf{pid} \leftarrow j$
5: $m_1 \leftarrow (vk_{i,\mathsf{OTS}}^s, \sigma^{(i)}, pk_i^s, i, \mathsf{pid})$
```

Step 2: An oracle, π_j^t, that computes Step 2 of the protocol specification receives as input the message $m_1 = (vk_{i,\mathsf{OTS}}^s, \sigma^{(i)}, pk_i^s, i, \mathsf{pid})$. We denote by C_j^t the KEM ciphertext that is generated by oracle π_j^t. For the simulation of π_j^t, we distinguish between two cases (Line 7 ff.). $\mathcal{B}_{\mathsf{KEM}}$ deviates from the protocol description only if the if-condition in Line 7 is satisfied.

If $\mathcal{B}_{\mathsf{KEM}}$ has received the KEM key pk_i^s under that a session key is to be encapsulated from its challenger then $\mathcal{B}_{\mathsf{KEM}}$ deviates from the protocol specification. In this case it lets the KEM challenger generate a ciphertext C_j^t, together with a key K_j^t. We note that we have $\Pr[\mathsf{KEM.Decap}(sk_i^s, C_j^t) = K_j^t] = \frac{1}{2}$. Here sk_i^s "matches" pk_i^s. Therefore $\mathcal{B}_{\mathsf{KEM}}$ does not simulate the AKE challenger perfectly. However, this lack in the simulation will not be detected by the adversary (as we will later see).
We note that C_j^t is a valid ciphertext. The key K_j^t is later used to respond to $\mathsf{Test}(j, t)$. If $pk_i^s \notin \mathcal{L}$ then this KEM public key is generated by the adversary $\mathcal{A}_{\mathsf{AKE}}$. In this case $\mathcal{B}_{\mathsf{KEM}}$ will follow the protocol description for Step 2.
In the former case $\mathcal{B}_{\mathsf{KEM}}$ embeds a challenge in each ciphertext whereas in the latter case there is no challenge embedded in the ciphertext. We will show later that $\mathcal{A}_{\mathsf{AKE}}$ will not be allowed to issue $\mathsf{Test}(i, s)$ or $\mathsf{Test}(j, t)$ in the latter case.

```
1: $(vk_{i,\mathsf{OTS}}^s, \sigma^{(i)}, pk_i^s, i, \mathsf{pid}) := m_1$
2: $a \leftarrow \mathsf{SIG.Vfy}(vk^{(i)}, vk_{i,\mathsf{OTS}}^s, \sigma^{(i)})$
3: $b \leftarrow \mathsf{pid} \overset{?}{=} j$
4: IF NOT $(a \wedge b)$ RETURN $\perp$
5: $(vk_{j,\mathsf{OTS}}^t, sk_{j,\mathsf{OTS}}^t) \xleftarrow{\$} \mathsf{OTS.Gen}(\Pi_{\mathsf{OTS}})$
6: $\sigma^{(j)} := \mathsf{SIG.Sign}(sk^{(j)}, vk_{j,\mathsf{OTS}}^t)$
7: IF $pk_i^s \in \mathcal{L}$ DO
8:     $(K_j^t, C_j^t) \xleftarrow{\$} \mathcal{O}_{\mathsf{Encap}}(pk_i^s)$
9:     $\mathcal{L}_i^s \leftarrow \mathcal{L}_i^s \cup \{C_j^t\}$.
10: ELSE
11:     $(K_j^t, C_j^t) = \mathsf{KEM.Encap}(pk_{i,\mathsf{KEM}}^s)$
12: $m_2 := (vk_{j,\mathsf{OTS}}^t, \sigma^{(j)}, C_j^t)$
13: $\sigma_{j,\mathsf{OTS}}^t \leftarrow \mathsf{OTS.Sign}(sk_{j,\mathsf{OTS}}^t, (m_1, m_2))$
```

Step 3: An oracle, π_i^s, that computes Step 3 of the protocol description receives as input the message $m_2 = (vk_{j,\mathsf{OTS}}^t, \sigma^{(j)}, C_i^s)$ and a one-time signature σ_j^t over (m_1, m_2). We denote with

C_i^s the KEM ciphertext that π_i^s receives. That is, there is probably another oracle π_j^t such that $C_j^t = C_i^s$. For the simulation of π_i^s, we distinguish between two cases (Line 6 ff.). $\mathcal{B}_{\mathsf{KEM}}$ deviates from the protocol description only if the if-condition in Line 6 is not satisfied.

If the ciphertext that π_i^s needs to decrypt was *not* generated by the KEM challenger, then $\mathcal{B}_{\mathsf{KEM}}$ will corrupt the respective KEM key and decrypt the ciphertext. We note that this branch will be used only if the ciphertext C_i^s is generated by $\mathcal{A}_{\mathsf{AKE}}$. This way, $\mathcal{B}_{\mathsf{KEM}}$ performs a ciphertext validity check. Otherwise, no ciphertext validity check is performed since the ciphertext was received from the KEM challenger. We note that $\mathcal{B}_{\mathsf{KEM}}$ does not simulate oracle π_i^s perfectly since no session key is computed. However, we will show later that this will not be detected by $\mathcal{A}_{\mathsf{AKE}}$ since all legitimate queries can be answered correctly by $\mathcal{B}_{\mathsf{KEM}}$.

```
1:  $(vk_{j,\mathsf{OTS}}^t, \sigma^{(j)}, C_i^s) := m_2$
2:  $a \leftarrow \mathsf{SIG.Vfy}(vk^{(j)}, vk_{j,\mathsf{OTS}}^t, \sigma^{(j)})$
3:  $b \leftarrow \mathsf{OTS.Vfy}(vk_{j,\mathsf{OTS}}^t, (m_1, m_2), \sigma_{j,\mathsf{OTS}}^t)$
4:  IF NOT $(a \wedge b)$ RETURN $\perp$
5:  $\sigma_{i,\mathsf{OTS}}^s := \mathsf{OTS.Sign}(sk_{i,\mathsf{OTS}}^s, (m_1, m_2))$
6:  IF $C_i^s \notin \mathcal{L}_i^s$ DO
7:      $sk_{i,\mathsf{KEM}}^s \leftarrow \mathcal{O}_{\mathsf{Corrupt}}(pk_i^s)$
8:      $K_i^s \leftarrow \mathsf{KEM.Decap}(sk_i^s, C_i^s)$
9:  ELSE
10:     $K_i^s := \emptyset$
11: RETURN $K_i^s$ and accept
```

Step 4: An oracle, π_j^t, that computes Step 4 of the protocol description receives as input a one time signature $\sigma_{i,\mathsf{OTS}}^s$ over (m_1, m_2). Recall that $m_1 = (vk_{i,\mathsf{OTS}}^s, \sigma^{(i)}, pk_i^s, i, j)$ and $m_2 = (vk_{j,\mathsf{OTS}}^t, \sigma^{(j)}, C_j^t)$. $\mathcal{B}_{\mathsf{KEM}}$ deviates from protocol specification only if the if-condition in Line 3 is satisfied.

If the KEM public key pk_i^s that is received by π_j^t in step two was not received from the KEM challenger, then $\mathcal{B}_{\mathsf{KEM}}$ has encapsulated a session key K_j^t in step two. Otherwise $\mathcal{B}_{\mathsf{KEM}}$ does not simulate π_j^t according to the protocol specification. However, as above, we will show that this will not be detected.

```
1: $a \leftarrow \mathsf{OTS.Vfy}(vk_{i,\mathsf{OTS}}^s, (m_1, m_2), \sigma_{i,\mathsf{OTS}}^s)$
2: IF NOT a RETURN $\perp$
3: IF $pk_i^s \in \mathcal{L}$
4:     $K_j^t \leftarrow \emptyset$
5: RETURN $K_j^t$ and accept
```

Simulating the AKE-challenger. In this section we show how $\mathcal{B}_{\mathsf{KEM}}$ answers to the queries that are issued by $\mathcal{A}_{\mathsf{AKE}}$.

- $\mathsf{Corrupt}(i)$: $\mathcal{B}_{\mathsf{KEM}}$ answers to this query exactly as in Game 2.
- $\mathsf{RegisterCorrupt}(i, pk_i)$ This query is also answered exactly as in Game 2.
- $\mathsf{Reveal}(i, s)$: If $\mathcal{A}_{\mathsf{AKE}}$ issues a $\mathsf{Reveal}()$-query and oracle π_i^s has internal state $\Psi_i^s = \mathtt{accept}$ then $\mathcal{B}_{\mathsf{KEM}}$ needs to return the session key that is computed by π_i^s. We distinguish between two cases.
  If it holds that $K_i^s = \emptyset$ then there is a pair (i', s') such that $C_i^s \in \mathcal{L}_{i'}^{s'}$. This is due the simulation of Step 3 (Line 6) and Step 4 (Line 3). In this case $\mathcal{B}_{\mathsf{KEM}}$ issues a $\mathsf{Corrupt}(i', s')$-query to the KEM challenger and receives back the secret key $sk_{i'}^{s'}$ which it then uses to compute $K_i^s \leftarrow \mathsf{KEM.Decap}(sk_{i'}^{s'}, C_i^s)$. Finally it returns K_i^s which is the real session key.

If it holds that $K_i^s \neq \emptyset$, $\mathcal{B}_{\mathsf{KEM}}$ simply returns K_i^s. We note that due to the simulation of Steps 3 (Line 10), 2 (Line 10) and 4 (Line 5) this is the real session key computed by π_i^s.

- $\mathsf{Test}(i, s)$: To answer to $\mathsf{Test}()$-queries, $\mathcal{B}_{\mathsf{KEM}}$ proceeds as follows. We note that this query is only allowed if π_i^s is fresh at the point in time when this query is issued. Due to Game 1 this means that there is a unique partner oracle π_j^t. By the definition of freshness, j is τ-uncorrupted. Over that, it follows from the simulation of Step 2 (Lines 7 to 9) that for the ciphertext C_i^s it either holds that $C_i^s \in \mathcal{L}_j^t$ or $C_i^s \in \mathcal{L}_i^s$. We assume wlog that $C_i^s \in \mathcal{L}_i^s$. The other case works analogously. The KEM challenger supplied $\mathcal{B}_{\mathsf{KEM}}$ with a challenge key K_i^s along with C_i^s. We note that $\Pr[K_i^s = \mathsf{KEM.Decap}(sk_i^s, C_i^s)] = \frac{1}{2}$. $\mathcal{B}_{\mathsf{KEM}}$ returns K_i^s.

This completes our description of $\mathcal{B}_{\mathsf{KEM}}$. Next, we show how $\mathcal{B}_{\mathsf{KEM}}$ uses the triplet (i, s, b) output by $\mathcal{A}_{\mathsf{AKE}}$ to break the security of KEM.

Extracting a solution for the KEM game. Now, let (i, s, b) be the output of $\mathcal{A}_{\mathsf{AKE}}$ in Game 2. I.e., b is its guess for b_i^s. Then $\mathcal{B}_{\mathsf{KEM}}$ will just forward this triplet, i.e., it also outputs (i, s, b). We note that $\mathcal{A}_{\mathsf{AKE}}$ loses if there is no τ such that π_i^s is τ-fresh and ∞-revealed throughout Game 2. Stated differently, $\mathcal{A}_{\mathsf{AKE}}$ wins only if

- π_i^s has $\hat{\tau}$-accepted, $\hat{\tau} \leq \tau$, and there is a unique partner oracle π_j^t such that π_i^s and π_j^t have matching conversations.
- π_i^s is ∞-revealed throughout the security game.
- π_j^t is ∞-revealed and ∞-tested throughout the security game.
- Party j is $\tilde{\tau}$-corrupted where $\tilde{\tau} > \tau$.

We show that if $\mathcal{A}_{\mathsf{AKE}}$ wins in Game 2, then $\mathcal{B}_{\mathsf{KEM}}$ is able to win the KEM game. We distinguish between two cases. Either π_i^s is an *initiator* oracle, i.e., it computes Steps 1 and 3 of the protocol descritpion or π_i^s is a *responder* oracle, that is, it computes Steps 2 and 4 of the protcol description.

Consider for the moment that π_i^s is an initiator oracle. This means that the KEM public key pk_{KEM} that is used to encapsulate the session key by oracle π_j^t is generated by the KEM challenger, i.e., $pk_{\mathsf{KEM}} = \mathsf{pk}_i^s$. Recall that to simulate π_j^t, $\mathcal{A}_{\mathsf{AKE}}$ issues an Encap-query $\mathsf{KEM.Encap}(pk_i^s)$ to the KEM challenger. Let (K_i^s, C_i^s) denote the response to this query. C_i^s is then sent from π_j^t to π_i^s. Recall further that to answer to a Test-query, $\mathsf{Test}(i, s)$, $\mathcal{B}_{\mathsf{KEM}}$ outputs the key K_i^s that was output by the KEM-challenger along with C_i^s.

On the other hand, if π_i^s is a responder oracle then it generates a ciphertext for a KEM public key $pk_{\mathsf{KEM}} = \mathsf{pk}_j^t$ that was sampled by π_j^t and j (due to Game 1) is τ-uncorrupted. Therefore $pk_j^t \in \mathcal{L}$. Recall again that to simulate π_i^s, $\mathcal{B}_{\mathsf{KEM}}$ issued an Encap-query $\mathsf{KEM.Encap}(pk_j^t)$ to the KEM-challenger. Let (K_i^s, C_i^s) denote the response to this query. Then, C_i^s is sent from π_i^s to π_j^t and in order to answer to a query $\mathsf{Test}(i, s)$ issued by $\mathcal{A}_{\mathsf{AKE}}$, $\mathcal{B}_{\mathsf{KEM}}$ returns K_i^s.

We note that in either case, if $\mathcal{A}_{\mathsf{AKE}}$ has advantage ϵ to win in Game 2 then the advantage of $\mathcal{B}_{\mathsf{KEM}}$ in the KEM security game is at least ϵ.

□

I Illustration of the AKE Protocol

```
$P_j$ has long-term key pair $(vk^{(j)}, sk^{(j)})$
$P_i$ has long-term key pair $(vk^{(i)}, sk^{(i)})$

                                              Step 1:
                                              $(vk^{(i)}_{\mathsf{OTS}}, sk^{(i)}_{\mathsf{OTS}}) \overset{\$}{\leftarrow} \mathsf{OTSIG.Gen}(\Pi_{\mathsf{OTSIG}})$
                                              $\sigma^{(i)} \overset{\$}{\leftarrow} \mathsf{SIG.Sign}(sk^{(i)}, vk^{(i)}_{\mathsf{OTS}})$
                                              $(sk^{(i)}_{\mathsf{KEM}}, pk^{(i)}_{\mathsf{KEM}}) \overset{\$}{\leftarrow} \mathsf{KEM.Gen}(\Pi_{\mathsf{KEM}})$
                                              $\mathsf{pid} := j$
                                              $m_1 := (vk^{(i)}_{\mathsf{OTS}}, \sigma^{(i)}, pk^{(i)}_{\mathsf{KEM}}, i, \mathsf{pid})$

                        $\xleftarrow{\quad m_1 \quad}$

Step 2:
$(vk^{(i)}_{\mathsf{OTS}}, \sigma^{(i)}, pk^{(i)}_{\mathsf{KEM}}, i, \mathsf{pid}) := m_1$
$a \leftarrow \mathsf{SIG.Vfy}(vk^{(i)}, vk^{(i)}_{\mathsf{OTS}}, \sigma^{(i)})$
$b \leftarrow \mathsf{pid} \overset{?}{=} j$
**IF NOT** $(a \wedge b)$ **RETURN** $\perp$
$(vk^{(j)}_{\mathsf{OTS}}, sk^{(j)}_{\mathsf{OTS}}) \overset{\$}{\leftarrow} \mathsf{OTSIG.Gen}(\Pi_{\mathsf{OTSIG}})$
$\sigma^{(j)} := \mathsf{SIG.Sign}(sk^{(j)}, vk^{(j)}_{\mathsf{OTS}})$
$(K, C) = \mathsf{KEM.Encap}(pk^{(i)}_{\mathsf{KEM}})$
$m_2 := (vk^{(j)}_{\mathsf{OTS}}, \sigma^{(j)}, C)$
$\sigma^{(j)}_{\mathsf{OTS}} := \mathsf{OTSIG.Sign}(sk^{(j)}_{\mathsf{OTS}}, (m_1, m_2))$

                        $\xrightarrow{\quad m_2, \sigma^{(j)}_{\mathsf{OTS}} \quad}$

                                              Step 3:
                                              $(vk^{(j)}_{\mathsf{OTS}}, \sigma^{(j)}, C) := m_2$
                                              $a \leftarrow \mathsf{SIG.Vfy}(vk^{(j)}, vk^{(j)}_{\mathsf{OTS}}, \sigma^{(j)})$
                                              $b \leftarrow \mathsf{OTSIG.Vfy}(vk^{(j)}_{\mathsf{OTS}}, (m_1, m_2), \sigma^{(j)}_{\mathsf{OTS}})$
                                              **IF NOT** $(a \wedge b)$ **RETURN** $\perp$
                                              $\sigma^{(i)}_{\mathsf{OTS}} := \mathsf{OTSIG.Sign}(sk^{(i)}_{\mathsf{OTS}}, (m_1, m_2))$
                                              $K = \mathsf{KEM.Decap}(sk^{(i)}_{\mathsf{KEM}}, C)$
                                              $K_{i,j} := K$
                                              **RETURN** $K_{i,j}$ and **accept**

                        $\xleftarrow{\quad \sigma^{(i)}_{\mathsf{OTS}} \quad}$

Step 4:
$a \leftarrow \mathsf{OTSIG.Vfy}(vk^{(i)}_{\mathsf{OTS}}, (m_1, m_2), \sigma^{(i)}_{\mathsf{OTS}})$
**IF NOT** a **RETURN** $\perp$
$K_{j,i} := K$
**RETURN** $K_{j,i}$ and **accept**
```

Figure 1: Generic AKE-Construction for Extended BR-Security