Tightly-Secure Authenticated Key Exchange

Christoph Bader[1], Dennis Hofheinz[2], Tibor Jager[1], Eike Kiltz[1], and Yong Li[1]

[1] Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
{christoph.bader,tibor.jager,eike.kiltz,yong.li}@rub.de
[2] Karlsruhe Institute of Technology, Germany
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, it 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 with only 14 group elements and 4 exponents communication complexity (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. [19].

Probably the most prominent example of an AKE protocol is the TLS Handshake [16,17,18], 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.[3] The wide application of AKE protocols makes it necessary and interesting to study their security in large-scale settings with many millions of users.

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

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 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, e.g., digital signatures [8,34,28,2], public-key encryption [4,25,31], or identity-based encryption [15,11]. 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 [6].

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. We will elaborate on the difficulty of constructing tightly secure AKE ini the next paragraph.

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 [33] 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 [13] based on the (assumed) security of the signature scheme and the hardness of the decisional Diffie-Hellman problem, which asks for a 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 [4], 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 Section 4.1. 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 respond 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.

Our contribution. We construct the first AKE protocols whose security does not degrade in the number of users and instances. Following [5] 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* [9,29]: the corruption of a long-term secret does not foil the security of previously established session keys. In addition to that, we prevent *key-compromise impersonation* attacks [10,23]: 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 [13,30]. 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.[4] The 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

[4] Intuitively, the matching conversations property, introduced by Bellare and Rogaway [5] 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 [13,14]) that separate different protocol instances.

$\mathcal{A}$ 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 implies 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., [25] 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 [25] 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 [33] 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.[5] 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 [25,1] and the Groth-Sahai non-interactive proof system [24] 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 [11] 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

[5] This can be seen as a variation of the approach of "two-signatures" approach of [21]. Concretely, [21] construct a signature scheme in which the simulation – by cleverly programming a random oracle – knows one out of two possible signatures for each message.

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.

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 [33]. (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 [25] 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 key 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* $b \in \{0,1\}$*.*

We note that our security definition below assumes a trusted setup of public parameters (using $\mathsf{SIG.Setup}$). Moreover, 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 [22]. 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*

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

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 [32].

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*

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

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 game it holds that*

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

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.

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 efficient algorithms.

- Algorithm $\mathsf{NIPS.Gen}$ takes as input the security parameter and ouputs a *common reference string* $\mathsf{CRS} \xleftarrow{\$} \mathsf{NIPS.Gen}(1^\kappa)$.
- Algorithm $\mathsf{NIPS.Prove}$ takes as input the CRS, a statement x and a witness w, and outputs a proof $\pi \xleftarrow{\$} \mathsf{NIPS.Prove}(\mathsf{CRS}, x, w)$.
- 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 $(x, w) \in R$, $\kappa \in \mathbb{N}$,* $\mathsf{CRS} \xleftarrow{\$} \mathsf{NIPS.Gen}(1^\kappa)$*, and all proofs π computed as* $\pi \xleftarrow{\$} \mathsf{NIPS.Prove}(\mathsf{CRS}, x, w)$ *holds that*

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

Perfect Witness Indistinguishability. *For all* $\mathsf{CRS} \xleftarrow{\$} \mathsf{NIPS.Gen}(1^\kappa)$*, 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*

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

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

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

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

$$\Pr\Big[w \xleftarrow{\$} \mathcal{E}_1(\mathsf{CRS}_{\mathit{sim}}, \pi, x, \tau) : (x, w) \in R\Big] = 1$$

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 $\kappa \in \mathbb{N}$, $\mathsf{CRS}_{\mathit{real}} \xleftarrow{\$} \mathsf{NIPS.Gen}(1^\kappa)$, all $(\mathsf{CRS}_{\mathit{sim}}, \tau) \xleftarrow{\$} \mathcal{E}_0(1^\kappa)$, it holds that*

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

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. The following is proven in the full version of our paper [3].

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

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

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

$$R := \left\{((\mathsf{vk}_0, \mathsf{vk}_1, m), (\sigma_0, \sigma_1)) : \begin{array}{r} \mathsf{SIG.Vfy}(\mathsf{vk}_0, m, \sigma_0) = 1 \\ \vee \mathsf{SIG.Vfy}(\mathsf{vk}_1, m, \sigma_1) = 1 \end{array}\right\}.$$

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:

– $\Pi_{\mathsf{SIG}_{\mathsf{MU}}} \xleftarrow{\$} \mathsf{SIG.Setup}_{\mathsf{MU}}(1^\kappa)$: The setup algorithm runs $\Pi_{\mathsf{SIG}} \xleftarrow{\$} \mathsf{SIG.Setup}(1^\kappa)$ and $\mathsf{CRS} \xleftarrow{\$} \mathsf{NIPS.Gen}(1^\kappa)$. 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: For $i \in \{0, 1\}$, it runs $(\mathsf{vk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{SIG.Gen}(\Pi_{\mathsf{SIG}})$. Then it flips a random coin $\delta \xleftarrow{\$} \{0, 1\}$ 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 $\sigma_\delta \xleftarrow{\$} \mathsf{SIG.Sign}(\mathsf{sk}_\delta, m)$. 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 returns a signature as $\sigma = \pi \xleftarrow{\$} \mathsf{NIPS.Prove}\big(\mathsf{CRS},\ (\mathsf{vk}_0, \mathsf{vk}_1, m), w\big)$.
– $\mathsf{SIG.Vfy}_{\mathsf{MU}}(vk, m, \sigma)$: 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. *Let $\mathsf{SIG}_{\mathsf{MU}}$ be as described above. From any attacker $\mathcal{A}_{\mathsf{SIG}_{\mathsf{MU}}}$ that (t, ϵ, μ)-breaks the* MU-EUF-CMA$^{\mathsf{Corr}}$*-security (with corruptions) of $\mathsf{SIG}_{\mathsf{MU}}$, we can construct algorithms $\mathcal{B}_{\mathsf{NIPS}}$ and $\mathcal{B}_{\mathsf{SIG}}$ 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, where*

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

We have $t_{\mathsf{CRS}} = t + t'_{\mathsf{CRS}}$ and $t_{\mathsf{SIG}} = t + t'_{\mathsf{SIG}}$, where t'_{CRS} and t'_{SIG} correspond to the respective runtimes required to provide $\mathcal{A}_{\mathsf{SIG}_{\mathsf{MU}}}$ with the simulated experiment as described below.

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 denote by χ_i the event that $\mathcal{A}_{\mathsf{SIG}_{\mathsf{MU}}}$ outputs (m^*, i^*, σ^*) such that $\mathsf{SIG.Vfy}(vk^{(i^*)}, m^*, \sigma^*) \wedge i^* \notin \mathcal{S}^{\mathsf{corr}} \wedge (m^*, \cdot) \notin \mathcal{S}_{i^*}$ in Game i.

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 $\mathsf{SIG.Gen}_{\mathsf{MU}}$, 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 $\mathsf{SIG}_{\mathsf{MU}}$-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 $(\mathsf{CRS}_{\mathsf{sim}}, \tau) \xleftarrow{\$} \mathcal{E}_0$ and all proofs are generated with respect to $\mathsf{CRS}_{\mathsf{sim}}$. Since the contrary would allow $\mathcal{B}_{\mathsf{NIPS}}$ to break the $(t, \epsilon_{\mathsf{CRS}})$-security of NIPS we have

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

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 $\mathcal{A}$ is valid, i.e., satisfies $\mathsf{SIG.Vfy}_{\mathsf{MU}}\left(vk^{(i^*)}, m^*, \sigma^*\right) = 1$, 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. For any attacker $\mathcal{A}_{\mathsf{SIG}_{\mathsf{MU}}}$ that breaks the $(t, \Pr[\chi_4], \mu)$-MU-EUF-CMA$^{\mathsf{Corr}}$-security of $\mathsf{SIG}_{\mathsf{MU}}$ 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 4. 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

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

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 $\mathcal{B}_{\mathsf{SIG}}$. We note further that, given $vk^{(i)}$, $sk^{(i)}$ is distributed correctly and may be returned by $\mathcal{B}_{\mathsf{SIG}}$ when $\mathcal{A}_{\mathsf{SIG}_{\mathsf{MU}}}$ issues a corrupt query (since it is generated by $\mathcal{B}_{\mathsf{SIG}}$ itself).

Over that $\mathcal{B}_{\mathsf{SIG}}$ 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. $\square$

(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 signatures can be found in [26,1] (based on the DLIN assumption in pairing-friendly groups).
– A suitable tightly MU-sEUF-1-CMA-secure one-time signature scheme is described in [26, Section 4.2]. Its security is based on the discrete logarithm assumption.
– Finally, a compatible NIWI-PoK is given by Groth-Sahai proofs [24]. (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

Here, 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* [20], which encompasses, e.g., the SXDH, DLIN, and k-Linear assumptions. The following definitions are taken from [11].

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 exponents by group elements as introduced in [20]. 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 under the discrete logarithm assumption in $\mathbb{G}_s$ it is hard to compute a from $[a]_s \in \mathbb{G}_s$ 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 [20].

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 $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$ 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 $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$, $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_q^k$ and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^{k+1}$.

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 $\mathcal{A}$ (ϵ, t)-breaks the $\mathcal{D}_k$-Matrix Diffie-Hellman ($\mathcal{D}_k$-MDDH) Assumption relative to* GGen *in group $\mathbb{G}_s$ if it runs in time at most t and*

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

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

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 $\mathsf{SIG}_\mathsf{C} = (\mathsf{SIG.Setup}_\mathsf{C}, \mathsf{SIG.Gen}_\mathsf{C}, \mathsf{SIG.Sign}_\mathsf{C}, \mathsf{SIG.Vfy}_\mathsf{C})$ for message $m \in \{0,1\}^\ell$ is based on a tightly-secure signature scheme from [11]. Whereas [11] obtained their signature scheme from a tightly-secure single-user algebraic MAC, we implicitly construct a tightly-secure *multi-user* algebraic MAC. More precisely, the signatures consist of the algebraic MAC part (elements $[\mathbf{r}]_2, [u]_2$) plus a NIZK proof $[\mathbf{v}]_2$ showing that the MAC is correct with respect to the committed MAC secret key $[\mathbf{c}]_1$.

The scheme works as follows.

– $\Pi \xleftarrow{\$} \mathsf{SIG.Setup}_\mathsf{C}(1^\kappa)$: The parameter generation algorithm $\mathsf{SIG.Setup}_\mathsf{C}$ runs $\mathcal{G} \xleftarrow{\$} \mathsf{GGen}$, $\mathbf{A}, \mathbf{A}' \xleftarrow{\$} \mathcal{D}_k$ 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 $\mathbf{x}_{i,b} \xleftarrow{\$} \mathbb{Z}_q^k$, $\mathbf{Y}_{i,b} \xleftarrow{\$} \mathbb{Z}_q^{k \times k}$, 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

$$\mathbf{x}(m) := \sum_{i=1}^{\ell} \mathbf{x}_{i,m_i}^\top \in \mathbb{Z}_q^{1 \times k}, \quad \mathbf{Y}(m) := \sum_{i=1}^{\ell} \mathbf{Y}_{i,m_i} \in \mathbb{Z}_q^{k \times k},$$
$$\mathbf{Z}(m) := \sum_{i=1}^{\ell} \mathbf{Z}_{i,m_i} = (\mathbf{Y}(m)^\top || \mathbf{x}(m)^\top) \cdot \mathbf{A} \in \mathbb{Z}_q^{k \times k}. \tag{2}$$

– $\mathsf{SIG.Gen}_\mathsf{C}(\Pi)$: The key generation algorithm picks $a \xleftarrow{\$} \mathbb{Z}_q$, $\mathbf{b} \xleftarrow{\$} \mathbb{Z}_q^k$, and defines $\mathbf{c}^\top = (\mathbf{b}^\top || a) \cdot \mathbf{A} \in \mathbb{Z}_q^{1 \times k}$. It returns $(\mathsf{vk}, \mathsf{sk}) = \big([\mathbf{c}]_1, ([a]_2, [\mathbf{b}]_2)\big) \in \mathbb{G}_1^k \times \mathbb{G}_2^{k+1}$.

– $\mathsf{SIG.Sign}_\mathsf{C}(\Pi, \mathsf{sk}, m)$: The signing algorithm parses sk as $\mathsf{sk} = ([a]_2, [\mathbf{b}]_2)$. Next, it picks $\mathbf{r}' \xleftarrow{\$} \mathbb{Z}_q^k$ and defines

$$\mathbf{r} := \mathbf{B} \cdot \mathbf{r}' \in \mathbb{Z}_q^k, \quad u = a + \mathbf{x}(m) \cdot \mathbf{r} \in \mathbb{Z}_q, \quad \mathbf{v} = \mathbf{b} + \mathbf{Y}(m) \cdot \mathbf{r} \in \mathbb{Z}_q^k. \tag{3}$$

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 $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^k$ and returns 1 iff the equation

$$e([\mathbf{c}^\top \cdot \mathbf{s}]_1, [1]_2) = e([\mathbf{A} \cdot \mathbf{s}]_1, \begin{bmatrix} \mathbf{v} \\ \mathbf{u} \end{bmatrix}_2) \cdot e([\mathbf{Z}(m) \cdot \mathbf{s}]_1, [\mathbf{r}]_2)^{-1} \quad (4)$$

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 $\epsilon < \epsilon_1 + 2\ell\epsilon_2 + 2/q$. We have $t_1 = t + t_1'$ and $t_2 = t + t_2$, where t_1' and t_2' correspond to the respective runtimes required to provide $\mathcal{A}_{\mathsf{SIG_{MU}}}$ with the simulated experiment as described below.*

Proof. As before, we proceed in a sequence of games where the first game is the 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 denote by χ_i the event that $\mathcal{A}_{\mathsf{SIG_{MU}}}$ outputs (m^*, i^*, σ^*) such that $\mathsf{SIG.Vfy}(vk^{(i^*)}, m^*, \sigma^*) \wedge i^* \notin \mathcal{S}^{\mathsf{corr}} \wedge (m^*, \cdot) \notin \mathcal{S}_{i^*}$ in Game i.

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 the final 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 (4) and returns 1 iff equation

$$e([(\mathbf{b}_{i^*}^\top || a_{i^*}) \cdot \mathbf{t}^*]_1, [1]_2) = e([\mathbf{t}^*]_1, \begin{bmatrix} \mathbf{v}^* \\ u^* \end{bmatrix}_2) \cdot e([(\mathbf{Y}^\top(m^*) || \mathbf{x}(m^*)^\top) \cdot \mathbf{t}^*]_1, [\mathbf{r}^*]_2)^{-1} \quad (5)$$

holds. By equation (2) and by the definition of $\mathbf{c}_{i^*}^\top = (\mathbf{b}_{i^*}^\top || a_{i^*}) \cdot \mathbf{A}$, equations (4) and (5) are equivalent. Hence,

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

Game 2. In this game, we again change the way the experiment treats the final forgery. 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 the formulas

$$\mathbf{Y}_{i,b}^\top = (\mathbf{Z}_{i,b} - \mathbf{x}_{i,b} \cdot \underline{\mathbf{A}})\overline{\mathbf{A}}^{-1}, \quad \mathbf{b}_i^\top = (\mathbf{c}_i^\top - a_i \cdot \underline{\mathbf{A}})\overline{\mathbf{A}}^{-1}, \tag{6}$$

respectively. The concrete changes are as follows. 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 (6). 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 (6).

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

$$\mathbf{v}^\top = (\mathbf{r}^\top \mathbf{Z}(m) + \mathbf{c}_i^\top - u \cdot \underline{\mathbf{A}}) \cdot \overline{\mathbf{A}}^{-1}. \tag{7}$$

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 (5) to

$$\begin{aligned} &e([\mathbf{c}_{i^*}^\top \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{t}^*} + a_{i^*} \cdot h^*]_1, [1]_2) \\ =&e([\mathbf{t}^*]_1, \begin{bmatrix} \mathbf{v}^* \\ \mathbf{u}^* \end{bmatrix}_2) \cdot e([\mathbf{Z}(m^*)\overline{\mathbf{A}}^{-1}\overline{\mathbf{t}^*} + \mathbf{x}(m^*)h^*]_1, [\mathbf{r}^*]_2)^{-1} \end{aligned} \tag{8}$$

By the substitution formulas for $\mathbf{Y}_{i,b}$ and $\mathbf{b}_i$ and be the definition of h and $\overline{\mathbf{t}^*}$, equations (3) and (7) and equations (5) and (8) 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 $\mathbf{c}_{i^*}^\top = (\mathbf{b}_{i^*}^\top || a_{i^*}) \cdot \mathbf{A}$. 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 (5) (which is equivalent to (8)) is uniform and independent from $\mathcal{A}$'s view. Hence,

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

The following lemma which essentially proves that the underlying message authentication code is tightly secure in a multi-user setting with corruptions completes the proof of the Theorem 2. It follows [11,15].

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, $0 \leq j \leq \ell$ that are played with an adversary $\mathcal{C}$. 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, 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) to adversary $\mathcal{C}$. Note that the value $\mathbf{v}$ is not provided by the oracle. On a Corrupt query called with i, hybrid H_j returns $[a_i]_2 = [\mathsf{RF}_{i,j}(m(i)_{|j})]_2$ to $\mathcal{C}$, 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 $h^* \overset{\$}{\leftarrow} \mathbb{Z}_q$. 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 $\mathbf{A} \overset{\$}{\leftarrow} \mathcal{D}_k$, inputs $[\mathbf{B}]_2$ and $([\mathbf{x}_{i,b}\mathbf{B}]_2)_{i,b}$ from the hybrid game H_0, picks $\mathbf{Z}_{i,b}$ at random, and computes $[\mathbf{Y}_{i,b}\mathbf{B}]_2$ via (6). 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 via (7). Forgery and Corrupt queries can be simulated the same way by defining $\mathsf{RF}_{i,0}(\varepsilon) =: a_i$. Hence $\Pr[\chi_3] = \Pr[H_0 = 1]$. Similarly, $\Pr[\chi_4] = \Pr[H_\ell = 1]$ 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. $|\Pr[H_{j-1} = 1] - \Pr[H_j = 1]| \leq 2\epsilon_2$, for a suitable adversary $\mathcal{B}_2$.

The proof of this claim essentially follows verbatim from Lemma B.3 of [11]. 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 $\mathbf{H} \overset{\$}{\leftarrow} \mathbb{Z}_q^{(k+1)\times Q}$. The Q-fold $\mathcal{D}_k$-MDDH assumption has been proved tightly equivalent to the $\mathcal{D}_k$-MDDH assumption in [20].

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

$$\mathsf{RF}_{i,j}(m_{|j}) := \begin{cases} \mathsf{RF}_{i,j-1}(m_{|j-1}) & m_j = \alpha \\ \mathsf{RF}_{i,j-1}(m_{|j-1}) + R_{i,m_{|j}} & m_j = 1-\alpha \end{cases}, \tag{9}$$

where $R_{i,m_{|j}} \overset{\$}{\leftarrow} \mathbb{Z}_q$. 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 $\mathbf{x}_{i,b} \overset{\$}{\leftarrow} \mathbb{Z}_q^k$ and implicitly defines $\mathbf{x}_{j,1-\alpha}^\top \mathbf{B} := \mathbf{x}'^\top \mathbf{A}'$ for $\mathbf{x}' \overset{\$}{\leftarrow} \mathbb{Z}_q^{k+1}$. 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 values $\mathit{\Pi}_{\mathsf{MAC}} = ([\mathbf{B}]_2, ([\mathbf{x}_{i,b}^\top \mathbf{B}]_2)_{i,b})$.

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 (9), 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 random $\mathbf{r}' \in \mathbb{Z}_q^k$, 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

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

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.

3 KEMs in the Multi-User Setting with Corruptions

In this section we will describe a generic construction of a key encapsulation mechanism (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 [4,25]. In particular, a tight security proof for the DLIN-based scheme from [12] is given in [25]. A similar scheme was generalized to hold under any MDDH-assumption [20].

Due to space limitations, we refer to the full version of our paper [3] for standard definitions of public key encryption (PKE) and KEMs.

Before we proceed let us first recall public key encryption and key encapsulation mechanisms.

3.1 Public-Key Encryption

A PKE scheme is a four-tuple of 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, $\perp$.

Security. The standard security notions for public key encryption in the multi-user setting (without corruptions) go back to Bellare, Boldyreva and Micali [4]. Security is formalized by a game that is played between an attacker $\mathcal{A}$ and a challenger $\mathcal{C}$.

1. After running $\Pi \xleftarrow{\$} \mathsf{PKE.Setup}(1^\kappa)$, $\mathcal{C}$ 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 Key Encapsulation Mechanisms

Definition 9. *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.

Multi User Security of KEMs. 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 10 (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 10 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.

3.3 Generic KEM Construction

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

– $\Pi \xleftarrow{\$} \mathsf{KEM.Setup_{MU}}(1^\kappa)$: The parameter generation algorithm $\mathsf{KEM.Setup_{MU}}$ 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}.
– $(sk, pk) \xleftarrow{\$} \mathsf{KEM.Setup_{MU}}(\Pi)$: 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 the secret key as $sk = (sk_\delta, \delta)$ and $C = (C_0, C_1)$. It computes $K \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_\delta, C_\delta)$ and returns K.

Theorem 3. *Let* $\mathsf{KEM}_{\mathsf{MU}}$ *be as described above. For each attacker* $\mathcal{A}^{KEM}$ *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* $\mathcal{A}^{PKE}$ *that* $(\epsilon_{\mathsf{pke}}, t_{\mathsf{pke}}, \mu, \ell)$*-breaks the* MU-IND-CPA*-security of* PKE *with* $t_{\mathsf{kem}} = t_{\mathsf{pke}} + t'_{\mathsf{kem}}$ *and* $\epsilon_{\mathsf{kem}} \leq \epsilon_{\mathsf{pke}}$*. Here* t'_{kem} *is the runtime required to provide* $\mathcal{A}^{\mathsf{KEM}}$ *with the simulation described below.*

Due to space limitations we omit the proof of Theorem 3 here. It can be found in the full version of our paper [3]. □

4 A Tightly-Secure AKE Protocol

4.1 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 [5] 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 [27], which allows an additional 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 $P_1, \dots, P_\mu$. 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, $\{\pi_i^1, \dots, \pi_i^\ell\}$, 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 $\mathsf{RegisterCorrupt}$ 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 $\Psi_i^s = \mathtt{accept}$ 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 $\mathsf{RegisterCorrupt}$-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 $\Psi_i^s = \mathtt{accept}$. 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 $\Psi_i^s \neq \mathtt{accept}$, 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 [5]. We adopt the refinement from [27].

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 $\mathcal{A}$ 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 $\tau^{(j)} > \tau$* [6].

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:

[6] 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.

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*

$$|\Pr[b_i^s = b'] - 1/2| \geq \epsilon.$$

We discuss and highlight properties of the model in the full version of our paper [3, Remark 2]

4.2 Our Tightly Secure AKE Protocol

Here, 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 and let KEM be a key-encapsulation mechanism. 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 possesses 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.

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 $m_1 := (vk^{(i)}_{\mathsf{OTS}}, \sigma^{(i)}, pk^{(i)}_{\mathsf{KEM}}, \mathsf{Pid}, i)$ and transmits m_1 to P_j.
2. Upon receiving m_1, P_j parses m_1 as the tuple $(vk^{(i)}_{\mathsf{OTS}}, \sigma^{(i)}, pk^{(i)}_{\mathsf{KEM}}, \mathsf{Pid}, i)$. 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 $(vk^{(j)}_{\mathsf{OTS}}, sk^{(j)}_{\mathsf{OTS}}) \xleftarrow{\$} \mathsf{OTSIG.Gen}(\Pi_{\mathsf{OTSIG}})$, encapsulates a key as $(K, C) \xleftarrow{\$} \mathsf{KEM.Encap}(pk^{(i)}_{\mathsf{KEM}})$ and computes a signature $\sigma^{(j)} := \mathsf{SIG.Sign}(sk^{(j)}, vk^{(j)}_{\mathsf{OTS}})$. Then it sets $m_2 := (vk^{(j)}_{\mathsf{OTS}}, \sigma^{(j)}, C)$ and computes

a one-time signature $\sigma_{\mathsf{OTS}}^{(j)} := \mathsf{OTSIG.Sign}(sk_{\mathsf{OTS}}^{(j)}, (m_1, m_2))$ and transmits the tuple $(m_2, \sigma_{\mathsf{OTS}}^{(j)})$ to P_i.

3. Upon receiving the message $(m_2, \sigma_{\mathsf{OTS}}^{(j)})$, P_i parses m_2 as $(vk_{\mathsf{OTS}}^{(j)}, \sigma^{(j)}, C)$ and checks whether $\mathsf{SIG.Vfy}\ (vk^{(j)}, vk_{\mathsf{OTS}}^{(j)}, \sigma^{(j)}) = 1$ and $\mathsf{OTSIG.Vfy}\ (vk_{\mathsf{OTS}}^{(j)}, (m_1, m_2), \sigma_{\mathsf{OTS}}^{(j)}) = 1$. If at least one of both check is not passed, then P_i outputs $\perp$ and rejects.
   Otherwise it computes $\sigma_{\mathsf{OTS}}^{(i)} := \mathsf{OTSIG.Sign}\ (sk_{\mathsf{OTS}}^{(i)}, (m_1, m_2))$ and sends $\sigma_{\mathsf{OTS}}^{(i)}$ to P_j. P_i outputs the session key as $K_{i,j} := \mathsf{KEM.Decap}(sk_{\mathsf{KEM}}^{(i)}, C)$.
4. Upon receiving $\sigma_{\mathsf{OTS}}^{(i)}$, P_j checks whether $\mathsf{OTSIG.Vfy}(vk_{\mathsf{OTS}}^{(i)}, (m_1, m_2), \sigma_{\mathsf{OTS}}^{(i)}) = 1$. If this fails, then $\perp$ is returned. Otherwise P_j outputs the session key $K_{i,j} := K$.

In the full version of the paper [3], we elaborate on the efficiency of our protocol when it is instantiated with building blocks from the literatur.

4.3 Proof of Security.

Theorem 4. *Let* AKE *be as described above. 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 10), 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*

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

Here, $t' = t + t''$ *where* t'' *corresponds to the runtime required to provide* $\mathcal{A}_{\mathsf{AKE}}$ *with the simulated experiment as described below.*

Proof. We prove the security of the proposed protocol AKE using the sequence-of-games approach, following [35,7]. The first game is the original attack game that is played between a challenger 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 $\Psi_i^s = \texttt{accept}$ 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:

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

4.4 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 Section 4.1. 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 $\mathsf{Pid}_i^s = j$[7] and
- 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

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

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

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

We refer to the full version of the paper [3] for a proof of the claim. □

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

We refer to the full version of the paper [3] for a proof of the claim. □

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

We refer to the full version of the paper [3] for a proof of the claim. □

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

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.

Summing up we see that:

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

4.5 Key Indistinguishability

Lemma 4. *For any attacker $\mathcal{A}$ that $(t, \mu, \ell, \epsilon_{\mathsf{Ind}})$-break* AKE *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*

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

The proof of Lemma 4 can be found in the full version of the paper [3]. Summing up probabilities, we obtain that

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

□

References

1. 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.
2. Christoph Bader. Efficient signatures with tight real world security in the random oracle model. In *Cryptology and Network Security*. Springer, 2014.
3. Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. Tightly secure authenticated key exchange. Cryptology ePrint Archive, Report 2014/797, 2014. `http://eprint.iacr.org/`.
4. 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.
5. 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.
6. 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.

7. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. pages 409–426, 2006.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. 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.
17. 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.
18. T. Dierks and E. Rescorla. RFC 5246: The transport layer security (tls) protocol; version 1.2, August 2008.
19. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
20. 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.
21. 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.
22. 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.

23. 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.
24. 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.
25. 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.
26. Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. Cryptology ePrint Archive, Report 2012/311, 2012. `http://eprint.iacr.org/`.
27. 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.
28. 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.
29. 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.
30. 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.
31. 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`.
32. Alfred Menezes and Nigel P. Smart. Security of signature schemes in a multi-user setting. *Des. Codes Cryptography*, 33(3):261–274, 2004.
33. 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.
34. 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.
35. Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`.