Authenticated Key Exchange and Signatures with Tight Security in the Standard Model

Shuai Han[1,2], Tibor Jager[3], Eike Kiltz[4], Shengli Liu[1,2,5](✉), Jiaxin Pan[6], Doreen Riepel[4], and Sven Schäge[4]

[1] School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
{dalen17,slliu}@sjtu.edu.cn
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[3] Bergische Universität Wuppertal, Germany
tibor.jager@uni-wuppertal.de
[4] Ruhr-Universität Bochum, Germany
{eike.kiltz,doreen.riepel,sven.schaege}@rub.de
[5] Westone Cryptologic Research Center, Beijing 100070, China
[6] Department of Mathematical Sciences,
NTNU – Norwegian University of Science and Technology, Trondheim, Norway
jiaxin.pan@ntnu.no

Abstract. We construct the first authenticated key exchange protocols that achieve tight security in the *standard model*. Previous works either relied on techniques that seem to inherently require a random oracle, or achieved only "Multi-Bit-Guess" security, which is not known to compose tightly, for instance, to build a secure channel.

Our constructions are generic, based on digital signatures and key encapsulation mechanisms (KEMs). The main technical challenges we resolve is to determine suitable KEM security notions which on the one hand are strong enough to yield tight security, but at the same time weak enough to be efficiently instantiable in the standard model, based on standard techniques such as universal hash proof systems.

Digital signature schemes with tight multi-user security in presence of adaptive corruptions are a central building block, which is used in all known constructions of tightly-secure AKE with full forward security. We identify a subtle gap in the security proof of the only previously known efficient standard model scheme by Bader *et al.* (TCC 2015). We develop a new variant, which yields the currently most efficient signature scheme that achieves this strong security notion without random oracles and based on standard hardness assumptions.

Keywords: Authenticated key exchange, digital signatures, tightness

1 Introduction

A *tight* security proof establishes a close relation between the security of a cryptosystem and its underlying building blocks, *independent* of deployment parameters such as the number of users, protocol sessions, issued signatures, etc. This enables a theoretically-sound instantiation with optimal parameters, without the need to compensate a security loss by increasing key lengths or group sizes.

AKE. Authenticated key exchange (AKE) protocols enable two parties to authenticate each other and compute a shared session key. In comparison to many other cryptographic primitives, standard security models for AKE are extremely complex. Following the approach of Bellare-Rogaway [5] and Canetti-Krawczyk [7], a very strong active adversary is considered, which essentially "carries" all protocol messages between parties running the protocol and thus is able to modify, replace, replay, drop, or inject arbitrary messages. Furthermore, the adversary may adaptively corrupt parties and reveal session keys while adaptively choosing which session(s) to "attack".

Achieving security in such a strong and complex model gives very strong security guarantees, but it also makes *tightness* particularly difficult to achieve. Indeed, most security proofs of AKE protocols are extremely lossy, often even with a *quadratic* security loss in the total number of sessions established over the entire lifetime of the protocol. Considering for instance the huge number of TLS connections per day in practice, this loss may be too large to compensate in practice because the resulting increase of deployment parameters would incur an intolerable performance overhead. Hence, such protocols could not be instantiated in a theoretically-sound way.

Therefore tight security of AKE protocols is a well-established research area, with several known constructions [2, 19, 29, 23, 13, 11]. As recently pointed out by Jager et al. [23], some of these constructions [2, 19, 29] consider a *"Multi-Bit-Guess"* (MBG) security experiment, which is not known to compose tightly with primitives that apply the shared session key, e.g., to build a secure channel. The standard and well established security notion in the context of multiple challenges is *"Single-Bit Guess"* (SBG) security. Unfortunately, the only known constructions in the SBG model [23, 13, 11] apply proof techniques that seem to inherently require the random oracle model [4]. For instance, [23] is based on non-committing encryption, which is known to be not instantiable without random oracles [32], whereas [13, 11] use a similar approach based on adaptive reprogramming of the random oracle.

Currently, there exists no AKE protocol which achieves tight security in a standard (SBG) AKE security model, with a security proof in the standard model, without random oracles, not even an impractical one.

Digital Signatures. Digital signatures are a foundational cryptographic primitive and often used to build AKE protocols. All known tightly-secure AKE protocols with full forward security [2, 19, 13, 11, 29, 23] are based on signatures that provide tight existential unforgeability under chosen-message attacks (EUF-CMA), but in a *multi-user* setting and in the presence of an adversary that may *adaptively corrupt* users to obtain their secret keys (MU-EUF-CMA$^{\mathsf{corr}}$ security [2]). It is easy to prove that MU-EUF-CMA$^{\mathsf{corr}}$ security is non-tightly implied by standard EUF-CMA security, but with a linear security loss in the number of users.

The construction of a *tightly* MU-EUF-CMA$^{\mathsf{corr}}$ secure signature scheme has to overcome the following, seemingly paradoxical technical problem. On the one hand, the reduction must be able to output user secret keys to the adversary, to respond to adaptive secret key corruption queries. However, it cannot apply a guessing argument, as this would incur a tightness loss. Therefore it is forced to "know" the secret keys of *all* users. On the other hand, it must be able to extract a solution to a computationally hard problem from a forgery produced by an adversary. This seems to be in conflict with the fact that the reduction has to know secret keys for all users, as knowledge of the secret key should enable the reduction to compute a "forged" signature by itself, without the adversary. In fact, tight multi-user security is known to be impossible for many signature schemes, for example when the public key uniquely defines the matching secret key [3].

Several previous works have developed techniques to overcome this seeming paradox [1, 2, 19, 12]. Essentially, their approach is to build schemes where secret keys are not uniquely determined by public parameters, along with a reduction that exploits this to evade the paradox. However, all currently known constructions either use the random oracle model, and therefore cannot be used to build tightly-secure AKE in the standard model, or are based on tree-based signatures [2], which yields signatures with hundreds of group elements and thus would incur even more overhead than compensating the security loss with larger parameters. Jumping slightly ahead, we remark that [2] also describes a pairing-based signature scheme with short constant-size signatures, but we identify a gap in the security proof. Hence, currently there is no practical signature scheme which achieves tight security in the multi-user setting with adaptive corruptions.

1.1 Contributions

Summarizing the previous paragraphs, we can formulate the following natural questions related to AKE and signatures:

> Do there exist efficient AKEs and signature schemes with tight multi-user security in the standard model?

Tightly-secure signatures. We identify a subtle gap in the MU-EUF-CMA$^{\mathsf{corr}}$ security proof of the scheme from [2] with constant-size signatures (namely, $\mathsf{SIG_C}$ in [2, Section 2.3]). We did not find a way to close this gap and therefore develop a new variant of this scheme and prove tight MU-EUF-CMA$^{\mathsf{corr}}$ security in the standard model. More precisely, $\mathsf{SIG_C}$ follows the blueprint of the Blazy-Kiltz-Pan (BKP) identity-based encryption scheme [6], and transforms an algebraic message authentication code (MAC) scheme into a signature scheme with pairings. If the MAC is tightly-secure in a model with adaptive corruptions, so is the signature scheme. We notice, however, that their MAC does not achieve tight security with adaptive corruptions since the corruption queries leak too much secret information to the adversary.

To overcome this issue, we borrow recent techniques from tightly-secure hierarchical identity-based encryption schemes [26, 27] to construct a new MAC

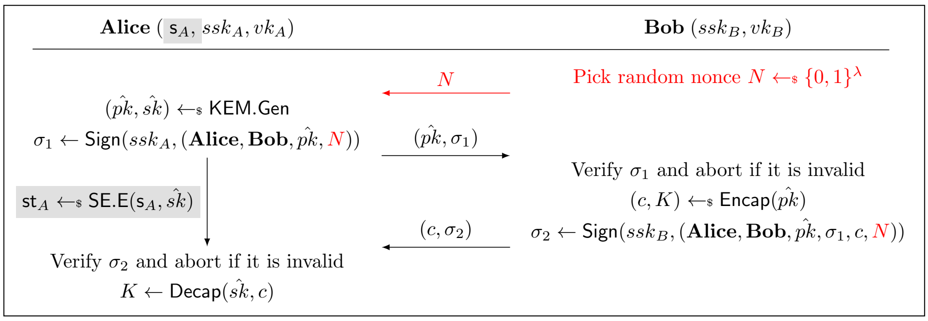

Fig. 1. The two-message protocol $\mathsf{AKE}_{\mathsf{2msg}}$ using the "$\mathsf{KEM}+2\times\mathsf{SIG}$" approach and the three-message protocols $\mathsf{AKE}_{\mathsf{3msg}}$ (including the red parts) and $\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$ (including the red and gray parts) using the "Nonce$+\mathsf{KEM}+2\times\mathsf{SIG}$" approach. ($\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$ additionally uses a symmetric encryption scheme SE.)

scheme that can be proven tightly secure under adaptive corruptions. Our construction is based on pairings and general random self-reducible matrix Diffie-Hellman (MDDH) assumptions [15]. When instantiated based on the $\mathcal{D}_k$-MDDH assumption (e.g., k-Lin), a signature consists of $4k+1$ group elements. That is 5 group elements for $k = 1$ (SXDH). This yields the first tightly MU-EUF-CMA$^{\mathsf{corr}}$-secure signature in the standard model with practical efficiency.

We remark that our new signature scheme circumvents known impossibility results for signatures and MACs [3, 30], since these apply only to schemes with re-randomizable signatures or re-randomizable secret keys [3], or deterministic schemes [30]. Our construction is probabilistic and not efficiently re-randomizable in the sense of [3].[7]

Tightly-secure AKE in the standard model. The classical *"key encapsulation plus digital signatures"* ($\mathsf{KEM}+2\times\mathsf{SIG}$) paradigm to construct AKE protocols gives rise to efficient protocols and is the basis of many constructions, e.g., [7, 10, 19, 13, 11, 29, 23]. To establish a session key, two parties Alice and Bob proceed as follows (cf. Figure 1). Alice generates an ephemeral KEM key pair $(\hat{pk}, \hat{sk})$ and sends the signed public key to Bob. Bob then uses this public key to encapsulate a session key, signs the ciphertext, and sends it back to Alice. Alice then obtains the session key K by decapsulating with the KEM secret key. For example, one can view the classical "signed Diffie-Hellman" as a specific instantiation of this paradigm, by considering the Diffie-Hellman protocol as the ElGamal KEM.

Our approach to construct efficient AKE protocols with tight security is based on this $\mathsf{KEM}+2\times\mathsf{SIG}$ paradigm. Given a tightly MU-EUF-CMA$^{\mathsf{corr}}$ secure signature scheme, it remains to determine suitable security notions for the underlying KEM, which finds a balance between two properties. The security

[7] Our signatures are only re-randomizable over all strings output by the signing algorithm. The impossibility result from [3] requires uniform re-randomizability over all strings accepted by the verification algorithm, which does not hold for our scheme.

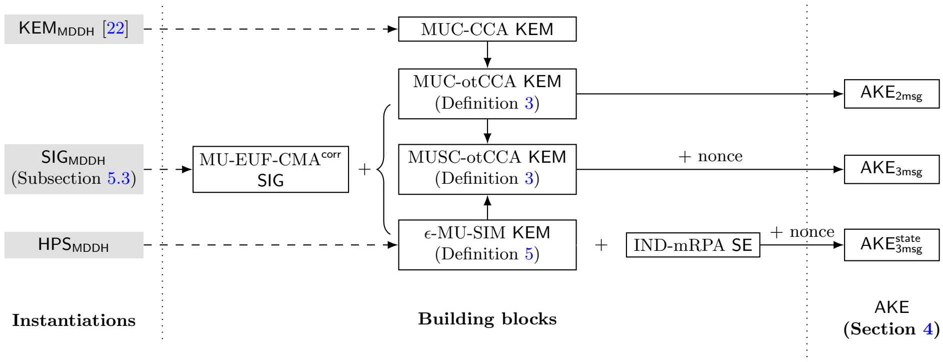

Fig. 2. Schematic overview of our AKE constructions.

notion must be *strong enough* to enable a *tight* security proof in presence of adaptive session key reveals and possibly even state reveals. At the same time, it must be *weak enough* to be achievable in the standard model. We now sketch the construction of our three AKE protocols along with the corresponding KEM security notions, see also Figure 2. In terms of AKE security, we consider a generic and versatile security model which provides strong properties, such as full forward security and key-compromise impersonation (KCI) security. "Partnering" of oracles is defined based on *original key partnering* [28]. The model is defined in pseudocode to avoid ambiguity.

- Our first result is a new tight security proof for the two-message protocol $\mathsf{AKE}_{2\mathsf{msg}}$, which follows the $\mathsf{KEM}{+}2 \times \mathsf{SIG}$ paradigm. $\mathsf{AKE}_{2\mathsf{msg}}$ is exactly the LLGW protocol [29] and the main technical difficulty is to adopt the LLGW proof strategy from the "Multi-Bit-Guess" to the standard "Single-Bit-Guess" setting. This requires significant modifications to the proof outline and the underlying KEM security definition. Our new proof relies on Multi-User/Challenge one-time CCA (MUC-otCCA) security for KEMs, allowing the adversary to ask many challenge queries but only one decapsulation query per user. Even though this is a slightly weaker version of the standard Multi-User/Challenge CCA (MUC-CCA) security notion for KEMs (allowing for unbounded decapsulation queries [17]), the most efficient instantiations we could find are the MUC-CCA-secure schemes with tight security from [17, 18, 22].[8]
- Our second result is a three-message protocol $\mathsf{AKE}_{3\mathsf{msg}}$ resisting replay attacks, which extends the $\mathsf{KEM} + 2 \times \mathsf{SIG}$ protocol $\mathsf{AKE}_{2\mathsf{msg}}$ with an additional nonce sent at the beginning of the protocol ("$\mathsf{Nonce} + \mathsf{KEM} + 2 \times \mathsf{SIG}$"). For our security proof we require the KEM security notion of Multi-User Single-Challenge one-time CCA (MUSC-otCCA) security, allowing the adversary

[8] We are aware of the generic constructions of bounded-CCA secure KEMs from CPA-secure KEMs [8], but they do not seem to offer tight security in a multi-challenge setting.

to ask only one challenge and one decapsulation query per user. This notion is considerably weaker than MUC-otCCA security and it is achievable from any universal$_2$ hash proof system [9]. (For example, based on a standard assumption such as Matrix DDH (MDDH) [15] which yields highly efficient KEMs.)

– Our third result is a three-message protocol $\mathsf{AKE}_{\mathsf{3msg}}^{\mathsf{state}}$, which extends the $\mathsf{Nonce} + \mathsf{KEM} + 2 \times \mathsf{SIG}$ protocol $\mathsf{AKE}_{\mathsf{3msg}}$ by encrypting the state with a symmetric encryption (SE) scheme. $\mathsf{AKE}_{\mathsf{3msg}}^{\mathsf{state}}$ has tight security in a very strong model that even allows the adversary to obtain session states of oracles [7]. The fact that the reduction must be able to respond to adaptive queries for session states by an adversary makes it significantly more difficult to achieve *tight* security. Our key technical contribution is a new "Multi-User SIMulatability" (ϵ-MU-SIM) security notion for KEMs, which we also show to be tightly achievable by universal$_2$ hash proof systems. We stress that the reduction to the security of the symmetric encryption scheme is the only part of the security proof which is *not* tight. We tolerate this, since compensating a security loss for symmetric encryption incurs significantly less performance penalty than for public key primitives.[9]

Note that our $\mathsf{AKE}_{\mathsf{3msg}}$ and $\mathsf{AKE}_{\mathsf{3msg}}^{\mathsf{state}}$ use nonce to resist replay attacks and admit KEM security with one challenge per user. This can also be achieved generically by assuming synchronized counters between parties, following the approach of [29]. Consequently, we can also use counter instead of nonce in $\mathsf{AKE}_{\mathsf{3msg}}$ and $\mathsf{AKE}_{\mathsf{3msg}}^{\mathsf{state}}$, and obtain two two-message counter-based AKE protocols which have the same efficiency and security as $\mathsf{AKE}_{\mathsf{3msg}}$ and $\mathsf{AKE}_{\mathsf{3msg}}^{\mathsf{state}}$, respectively.

INSTANTIATIONS. Table 1 gives example instantiations of our protocols from universal$_2$ hash proof systems from the MDDH assumption and compares them to known protocols. The protocols BHJKL [2] and LLGW [29] only offer tight security in the MBG model which implies security in our standard SBG model with a loss of T, the number of test queries [23]. For more details on our instantiations we refer to Section 6. Note that there are other works which study AKE in the standard model (e.g., [16, 24]). However, they do not focus on tightness and have a quadratic security loss.

TECHNICAL APPROACH TO AKE. In the following, we give a brief overview of our technical approach to tight security under our SBG-type security definition and show how our protocols prevent replay attacks and support state reveals.

To obtain an AKE protocol with a tight security reduction in the $\mathsf{KEM} + 2 \times \mathsf{SIG}$ framework, we rely on the tight MU-EUF-CMA$^{\mathsf{corr}}$ security of the signature scheme to guarantee authentication and deal with corruptions, and on the tight MUC-CCA security of KEM to deal with session key reveals. To this end, recall

[9] For instance, `openssl speed aes` shows that AES-256 is only about 1.5 times slower than AES-128 on a standard laptop computer. Given that the cost of symmetric key operations is already small in comparison to the public key operations, we consider this as negligible.

Table 1. Comparison of standard model AKE protocols with full forward security, where T refers to the number of test queries. Protocols $\mathsf{AKE}_{\mathsf{3msg}}^{\mathsf{state}}$ and $\mathsf{AKE}_{\mathsf{2msg}}$ refer to our protocols given in Fig. 1, instantiated from $\mathcal{D}_k$-MDDH. The column **Communication** counts the communication complexity of the protocols in terms of the number of group elements, exponents and nonces, where we instantiate all protocols with our new signature scheme from Subsection 5.3. The column **Security Loss** lists the security loss of the reduction in the "Single-Bit-Guess" (SBG) model, ignoring all symmetric bounds.

Protocol	**Communication**	**#Msg.**	**Assumption**	**State Reveal**	**Security Loss**
BHJKL [2]	$11 + 11$ $(2k^2 + 6k + 5) + (6k + 9)$	3	SXDH $\mathcal{D}_k$-MDDH	no	$O(\lambda T)$
LLGW [29]	$9 + 10$ $(k^2 + 7k + 1) + (6k + 4)$	2	SXDH $\mathcal{D}_k$-MDDH	no	$O(\lambda T)$
$\mathsf{AKE}_{\mathsf{3msg}}^{\mathsf{state}}$	$8 + 7$ $(5k + 3) + (5k + 2)$	3	SXDH $\mathcal{D}_k$-MDDH	yes	$O(\lambda)$
$\mathsf{AKE}_{\mathsf{2msg}}$ (= LLGW)	$9 + 10$ $(k^2 + 7k + 1) + (6k + 4)$	2	SXDH $\mathcal{D}_k$-MDDH	no	$O(\lambda)$

that the SBG-style security game for MUC-CCA security allows multiple encapsulation and decapsulation queries per user, but considers only a single challenge bit. At the same time, observe that the reduction algorithm can always use the challenge key (which is either the real encapsulated key or a random key) as the session key of the simulated AKE protocol. In combination, these observations immediately lead to a tight security proof for $\mathsf{AKE}_{\mathsf{2msg}}$. We remark that $\mathsf{AKE}_{\mathsf{2msg}}$ can also be proved secure under an even weaker security notion for KEM, namely MUC-otCCA, which allows only one decapsulation query per user. This assumes that parties choose to "close" a session once this session accepts or rejects. In this way we can guarantee that the adversary has only a single opportunity to submit a ciphertext per $\hat{pk}$.

To prevent replay attacks we make use of an exchange of nonces resulting in protocol $\mathsf{AKE}_{\mathsf{3msg}}$. As a byproduct of using nonces (in combination with the signature scheme), we can now guarantee that the adversary cannot replay any message anymore. This includes $\hat{pk}$, and thus we can ensure that the simulator only needs to respond to one encapsulation query per $\hat{pk}$ in the security game. In this way we can further weaken the security requirement that we need from the KEM to MUSC-otCCA.

Now, to support state reveals, we use a symmetric encryption scheme SE that is used to encrypt the ephemeral secret key $\hat{sk}$ of each session, similar to [23]. More concretely, we require that the state is computed as $\mathsf{st} = \mathsf{SE.E}(\mathsf{s}, \hat{sk})$, where s is the secret key of SE that is made part of the long-term secret key. This modification yields protocol $\mathsf{AKE}_{\mathsf{3msg}}^{\mathsf{state}}$. Having introduced such a state, we now also consider a security model that allows the adversary to issue state reveal queries to obtain the state st. But now the reduction to the MUSC-otCCA security of the KEM cannot work as before, since the reduction algorithm cannot output $\mathsf{SE.E}(\mathsf{s}, \hat{sk})$ to the adversary. A natural way to address this problem is to make use of the security of SE, and make the reduction change the state to an encryption of some dummy random key r , i.e., $\mathsf{st} = \mathsf{SE.E}(\mathsf{s}, r)$. However, now the SE reduction algorithm is faced with a critical decision: If the adversary asks a state reveal query, should the reduction output $\mathsf{st} = \mathsf{SE.E}(\mathsf{s}, \hat{sk})$ or $\mathsf{st} = \mathsf{SE.E}(\mathsf{s}, r)$? It seems that both choices are problematic. If the reduction responds with the encryption of KEM secret key $\hat{sk}$, then the reduction to the KEM will fail in case the adversary asks a test query. If on the other hand the reduction outputs an encryption of a dummy random key, then the reduction will fail in case the adversary queries the secret key via a corrupt query. To solve this problem, the existing approaches rely on a non-committing symmetric encryption scheme that is proven secure in the random oracle model [23].

To obtain a tight security supporting state reveals in the standard model, we enhance the MUSC-otCCA security of KEM to our new ϵ-MU-SIM-security, so that a symmetric encryption scheme SE with comparatively weak security guarantees suffices. The idea is to rely on a security notion for the symmetric encryption scheme that is as weak as possible while still being able to compensate for this via a stronger KEM. Somewhat surprisingly, our proof shows that when relying on an ϵ-MU-SIM-secure KEM, we only need to require IND-mRPA security (indistinguishability against random plaintext attacks) from SE. Such a symmetric encryption scheme can be easily instantiated using any weakly secure (deterministic) encryption scheme like as AES or even using a weak PRF. Let us now describe ϵ-MU-SIM-secure KEM in slightly more detail. In a nutshell, an ϵ-MU-SIM-secure KEM provides the reduction with access to an additional encapsulation algorithm Encap^* that is keyed with the secret key. We have security requirements as follows:

- Computational indistinguishability between Encap and Encap^*: We require that the reduction can switch to using Encap^* without the adversary noticing even given the secret key $\hat{sk}$ of the KEM. In particular, the resulting indistinguishability notion must tightly reduce to an underlying security assumption.
- Statistical ϵ-uniformity: When using the alternative encapsulation mechanism Encap^*, we require that the encapsulated key in the challenge ciphertext c^* will be indistinguishable from random with *statistical distance* ϵ (even if a decapsulation of some distinct ciphertext $c \neq c^*$ of its choice is given). This is particularly useful when aiming at tight security reductions.
- Since we want to apply ϵ-MU-SIM-secure KEMs in a protocol setting with multiple parties, security must in general hold in a multi-user setting.

Fortunately, such a KEM can be instantiated from universal$_2$ hash proof systems (HPS). In particular, we show that the ϵ-MU-SIM-security is implied by the hardness of subset membership problems and the universal$_2$-property of HPS.

Our new ϵ-MU-SIM-secure KEM now allows us to avoid the above mentioned decision when dealing with state reveals and in this way opens a new avenue towards a tight security reduction. To this end, we use a novel strategy in our security proof.

1. We first switch from using Encap to Encap^*. By the security properties of our KEM, the adversary cannot notice this, even given $\hat{sk}$.
2. Next, we replace the session keys of tested sessions with random keys – one user at a time. We apply a hybrid argument over all users. In the η-th hybrid ($\eta = 1, ..., \mu$ with μ being the number of users), we replace the test session keys related to the η-th user with random keys. We can show that this is not recognizable by the adversary since the key K^* generated by Encap^* is statistically close to uniform even if the adversary gets to see another key for a ciphertext of its choice. We distinguish the following cases.
   Case 1: The adversary corrupts the η-th user. For each session related to this user, the adversary can either reveal the session state or test this session, but not both. If the adversary reveals the state, we do not have to replace the session key at all, so the change is in fact only a conceptual one. If the session is tested, the adversary does not know the state $\mathsf{SE.E}(\mathsf{s}, \hat{sk})$ and thus we can replace the session key by exploiting the ϵ-uniformity of Encap^*.
   Case 2: The adversary does not corrupt the η-th user. In this case, we rely on the IND-mRPA security of SE and replace $\hat{sk}$ in the encrypted state with a random dummy key for this user. Then, we can use ϵ-uniformity to replace all tested keys for that user with random keys, as the state does not contain any information about $\hat{sk}$. After that, we have to switch back to using the original state encryption mechanism and encrypt the real secret key $\hat{sk}$, getting ready for the next hybrid.
   After μ hybrids, we change all tested keys to random. At this point the adversary has no advantage in the security game.

Overall, this security proof loses a factor of 2μ – but only when reducing to the IND-mRPA security of the symmetric encryption scheme. All other steps of the proof feature tight security reductions.

2 Security Notions for KEMs

2.1 Preliminaries

Let $\emptyset$ denote an empty string. If x is defined by y or the value of y is assigned to x, we write $x := y$. For $\mu \in \mathbb{N}$, define $[\mu] := \{1, 2, ..., \mu\}$. Denote by $x \leftarrow_\$ \mathcal{X}$ the procedure of sampling x from set $\mathcal{X}$ uniformly at random. If $\mathcal{D}$ is distribution, $x \leftarrow \mathcal{D}$ means that x is sampled according to $\mathcal{D}$. All our algorithms are probabilistic unless states otherwise. We use $y \leftarrow_\$ \mathcal{A}(x)$ to define the random variable y obtained by executing algorithm $\mathcal{A}$ on input x. We use $y \in \mathcal{A}(x)$ to indicate that y lies in the support of $\mathcal{A}(x)$. If $\mathcal{A}$ is deterministic we write $y \leftarrow \mathcal{A}(x)$. We also use $y \leftarrow \mathcal{A}(x; r)$ to make the random coins r used in the probabilistic computation explicit. Denote by $\mathbf{T}(\mathcal{A})$ the running time of $\mathcal{A}$. For two distributions X and Y, the statistical distance between them is defined by $\Delta(X;Y) := \frac{1}{2} \cdot \sum_x |\Pr[X = x] - \Pr[Y = x]|$, and conditioned on $Z = z$, the statistical distance between X and Y is denoted by $\Delta(X;Y|Z = z)$. For $0 \leq \epsilon \leq 1$, X and Y are said to be ϵ-close, denoted by $X \approx_\epsilon Y$, if $\Delta(X;Y) \leq \epsilon$.

Definition 1 (Collision-resistant hash functions). *A family of hash functions $\mathcal{H}$ is collision resistant if for any adversary $\mathcal{A}$,*

$$\mathsf{Adv}^{\mathsf{cr}}_{\mathcal{H}}(\mathcal{A}) := \Pr[x_1 \neq x_2 \wedge H(x_1) = H(x_2) | (x_1, x_2) \leftarrow_\$ \mathcal{A}(H), H \leftarrow_\$ \mathcal{H}].$$

2.2 Key Encapsulation Mechanisms

Definition 2 (KEM). *A key encapsulation mechanism (KEM) scheme* $\mathsf{KEM} = (\mathsf{KEM.Setup}, \mathsf{KEM.Gen}, \mathsf{Encap}, \mathsf{Decap})$ *consists of four algorithms:*

- $\mathsf{KEM.Setup}$*: The setup algorithm outputs public parameters* $\mathsf{pp}_{\mathsf{KEM}}$*, which determine an encapsulation key space $\mathcal{K}$, public key & secret key spaces $\mathcal{PK} \times \mathcal{SK}$, and a ciphertext space $\mathcal{CT}$.*
- $\mathsf{KEM.Gen}(\mathsf{pp}_{\mathsf{KEM}})$*: Taking* $\mathsf{pp}_{\mathsf{KEM}}$ *as input, the key generation algorithm outputs a pair of public key and secret key $(pk, sk) \in \mathcal{PK} \times \mathcal{SK}$. W.l.o.g., we assume that* $\mathsf{KEM.Gen}$ *first samples $sk \leftarrow_\$ \mathcal{SK}$ uniformly, and then computes pk from sk deterministically via a polynomial-time algorithm* $\mathsf{KEM.PK}$*, i.e., $pk :=$* $\mathsf{KEM.PK}(sk)$*. This is reasonable since we can always take the randomness used by* $\mathsf{KEM.Gen}$ *as the secret key.*
- $\mathsf{Encap}(pk)$*: Taking pk as input, the encapsulation algorithm outputs a pair of ciphertext $c \in \mathcal{CT}$ and encapsulated key $K \in \mathcal{K}$.*
- $\mathsf{Decap}(sk, c)$*: Taking as input sk and c, the deterministic decapsulation algorithm outputs $K \in \mathcal{K} \cup \{\perp\}$.*

We require that for all $\mathsf{pp}_{\mathsf{KEM}} \in \mathsf{KEM.Setup}$, $(pk, sk) \in \mathsf{KEM.Gen}(\mathsf{pp}_{\mathsf{KEM}})$, $(c, K) \in \mathsf{Encap}(pk)$*, it holds that* $\mathsf{Decap}(sk, c) = K$*.*

We define two security notions for KEMs, the first one in the Multi-User/Challenge (MUC) setting, the second one in the Multi-User and Single Challenge (MUSC) setting. Both notions only allow for one single decapsulation query per user.

Definition 3 (MUC-otCCA/MUSC-otCCA Security for KEM). *To* KEM*, the number of users $\mu \in \mathbb{N}$, and an adversary $\mathcal{A}$ we associate the advantage functions* $\mathsf{Adv}^{\mathsf{muc\text{-}otcca}}_{\mathsf{KEM},\mu}(\mathcal{A}) := \left|\Pr[\mathsf{Exp}^{\mathsf{muc\text{-}otcca}}_{\mathsf{KEM},\mu,\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right|$ *and* $\mathsf{Adv}^{\mathsf{musc\text{-}otcca}}_{\mathsf{KEM},\mu}(\mathcal{A}) := \left|\Pr[\mathsf{Exp}^{\mathsf{musc\text{-}otcca}}_{\mathsf{KEM},\mu,\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right|$*, where the experiments are defined in Figure 3.*

Below we recall the definition of the *diversity property* from [29].

Definition 4 (γ-Diversity of KEM). *A KEM scheme* KEM *is called γ-diverse if for all* $\mathsf{pp}_{\mathsf{KEM}} \in \mathsf{KEM.Setup}$*, it holds that*

$$\Pr\left[\begin{matrix}(pk, sk) \leftarrow_\$ \mathsf{KEM.Gen}(\mathsf{pp}_{\mathsf{KEM}}); \\ r, r' \leftarrow_\$ \mathcal{R}; (c, K) \leftarrow \mathsf{Encap}(pk; r); (c', K') \leftarrow \mathsf{Encap}(pk; r')\end{matrix} : K = K'\right] \leq 2^{-\gamma},$$

$$\Pr\left[\begin{matrix}(pk, sk) \leftarrow_\$ \mathsf{KEM.Gen}(\mathsf{pp}_{\mathsf{KEM}}); (pk', sk') \leftarrow_\$ \mathsf{KEM.Gen}(\mathsf{pp}_{\mathsf{KEM}}); \\ r \leftarrow_\$ \mathcal{R}; (c, K) \leftarrow \mathsf{Encap}(pk; r); (c', K') \leftarrow \mathsf{Encap}(pk'; r)\end{matrix} : K = K'\right] \leq 2^{-\gamma},$$

where $\mathcal{R}$ is the randomness space of Encap*. If $\gamma = \log |\mathcal{K}|$, then* KEM *is perfectly diverse.*

$\mathsf{Exp}^{\mathsf{muc\text{-}otcca}}_{\mathsf{KEM},\mu,\mathcal{A}}$, $\boxed{\mathsf{Exp}^{\mathsf{musc\text{-}otcca}}_{\mathsf{KEM},\mu,\mathcal{A}}}$:	$\mathcal{O}^b_{\text{ENCAP}}(i)$: $\boxed{\text{//at most once per user } i}$
$\mathsf{pp_{KEM}} \leftarrow_{\$} \mathsf{KEM.Setup}$ For $i \in [\mu]$: $(pk_i, sk_i) \leftarrow_{\$} \mathsf{KEM.Gen}(\mathsf{pp_{KEM}})$ $\mathsf{EncList} := \varnothing$ //Records the encapsulation queries $b \leftarrow_{\$} \{0,1\}$ //Single challenge bit $\mathsf{PKList} := \{pk_i\}_{i\in[\mu]}$ $b' \leftarrow_{\$} \mathcal{A}^{\mathcal{O}^b_{\text{ENCAP}}(\cdot),\mathcal{O}_{\text{DECAP}}(\cdot,\cdot)}(\mathsf{pp_{KEM}}, \mathsf{PKList})$ If $b' = b$: Return 1; Else: Return 0	$(c, K) \leftarrow_{\$} \mathsf{Encap}(pk_i)$ $\mathsf{EncList} := \mathsf{EncList} \cup \{(i, c)\}$ $K_0 := K$; $K_1 \leftarrow_{\$} \mathcal{K}$ Return (c, K_b) $\mathcal{O}_{\text{DECAP}}(i, c')$: // at most once per user i If $(i, c') \notin \mathsf{EncList}$: Return $K' \leftarrow \mathsf{Decap}(sk_i, c')$ Else: Return $\perp$

Fig. 3. The MUC-otCCA security experiment $\mathsf{Exp}^{\mathsf{muc\text{-}otcca}}_{\mathsf{KEM},\mu,\mathcal{A}}$ and the MUSC-otCCA security experiment $\mathsf{Exp}^{\mathsf{musc\text{-}otcca}}_{\mathsf{KEM},\mu,\mathcal{A}}$ of KEM, where in the latter the adversary can query the encapsulation oracle only once for each user.

We also propose a new security notion for KEMs called ϵ-MU-SIM (ϵ-multi-user simulatable) security. Jumping ahead, ϵ-MU-SIM secure KEMs will serve as the main building block in our generic AKE construction with state reveal later. We present the formal definition of ϵ-MU-SIM security (Definition 5). We also present simple constructions of ϵ-MU-SIM secure KEMs from universal$_2$-HPS in the full version [21].

Informally, ϵ-MU-SIM security requires that there exists a simulated encapsulation algorithm $\mathsf{Encap}^*(sk)$ which returns simulated ciphertext/key pairs (c^*, K^*) satisfying the following two properties. Firstly, they should be computationally indistinguishable from real ciphertext/key pairs. Secondly, given c^* and an arbitrary single decryption query, the simulated key K^* should be ϵ-close to uniform.

Definition 5 (ϵ-MU-SIM Security for KEM). *We require that there exists a simulated encapsulation algorithm* $\mathsf{Encap}^*(sk)$ *which takes the secret key* sk *as input, and outputs a pair of simulated* $c^* \in \mathcal{CT}$ *and simulated* $K^* \in \mathcal{K}$*. For* ϵ*-uniformity we require that for any (unbounded) adversary* $\mathcal{A}$*, it holds that*

$$\begin{aligned} \Big| \ &\Pr[c \leftarrow_{\$} \mathcal{A}(pk, c^*, K^*) \ : \ c \neq c^* \wedge \mathcal{A}(pk, c^*, K^*, \mathsf{Decap}(sk, c)) \Rightarrow 1] \\ - \ &\Pr[c \leftarrow_{\$} \mathcal{A}(pk, c^*, R) \ : \ c \neq c^* \wedge \mathcal{A}(pk, c^*, R, \mathsf{Decap}(sk, c)) \Rightarrow 1] \ \Big| \leq \epsilon, \end{aligned} \tag{1}$$

where the probability is over $\mathsf{pp_{KEM}} \leftarrow_{\$} \mathsf{KEM.Setup}$, $(pk, sk) \leftarrow_{\$} \mathsf{KEM.Gen}(\mathsf{pp_{KEM}})$, $(c^*, K^*) \leftarrow_{\$} \mathsf{Encap}^*(sk)$, $R \leftarrow_{\$} \mathcal{K}$ *and the internal randomness of* $\mathcal{A}$.

Furthermore, to KEM*, a simulated encapsulation algorithm* Encap^**, an adversary* $\mathcal{A}$*, and* $\mu \in \mathbb{N}$ *we associate the advantage function* $\mathsf{Adv}^{\mathsf{mu\text{-}sim}}_{\mathsf{KEM},\mathsf{Encap}^*,\mu}(\mathcal{A}) :=$

$$\left| \Pr\left[\mathcal{A}(\{pk_i, sk_i, c_i^{(0)}, K_i^{(0)}\}_{i\in[\mu]}) \Rightarrow 1\right] - \Pr\left[\mathcal{A}(\{pk_i, sk_i, c_i^{(1)}, K_i^{(1)}\}_{i\in[\mu]}) \Rightarrow 1\right] \right|, \tag{2}$$

where $\mathsf{pp_{KEM}} \leftarrow_{\$} \mathsf{KEM.Setup}$, $(pk_i, sk_i) \leftarrow_{\$} \mathsf{KEM.Gen}(\mathsf{pp_{KEM}})$, $(c_i^{(0)}, K_i^{(0)}) \leftarrow_{\$} \mathsf{Encap}(pk_i)$*, and* $(c_i^{(1)}, K_i^{(1)}) \leftarrow_{\$} \mathsf{Encap}^*(sk_i)$ *for* $\forall i \in [\mu]$.

Note that ϵ-MU-SIM security tightly implies MUSC-otCCA$^{\mathsf{rev\&corr}}$ security which is a stronger variant of MUSC-otCCA security supporting key reveal and user corrupt queries. Reveal and corrupt queries can be tolerated since in the security experiment (2), adversary $\mathcal{A}$ also obtains secret keys $sk_1, \ldots, sk_\mu$. By (1) one can see that one single decapsulation query is supported. In particular, ϵ-MU-SIM security tightly implies MUSC-otCCA security. In the full version [21], we will define universal$_2$ hash proof systems, construct $\mathsf{HPS}_{\mathsf{MDDH}}$ schemes from the MDDH assumptions, and show how they imply ϵ-MU-SIM secure KEMs.

3 Authenticated Key Exchange

3.1 Definition of Authenticated Key Exchange

Definition 6 (AKE). *An authenticated key exchange (AKE) scheme* $\mathsf{AKE} = (\mathsf{AKE.Setup}, \mathsf{AKE.Gen}, \mathsf{AKE.Protocol})$ *consists of two probabilistic algorithms and an interactive protocol.*

- $\mathsf{AKE.Setup}$*: The setup algorithm outputs the public parameter* $\mathsf{pp}_{\mathsf{AKE}}$.
- $\mathsf{AKE.Gen}(\mathsf{pp}_{\mathsf{AKE}}, P_i)$*: The generation algorithm takes as input* $\mathsf{pp}_{\mathsf{AKE}}$ *and a party* P_i*, and outputs a key pair* (pk_i, sk_i).
- $\mathsf{AKE.Protocol}(P_i(\mathsf{res}_i) \rightleftharpoons P_j(\mathsf{res}_j))$*: The protocol involves two parties* P_i *and* P_j*, who have access to their own resources,* $\mathsf{res}_i := (sk_i, \mathsf{pp}_{\mathsf{AKE}}, \{pk_u\}_{u\in[\mu]})$ *and* $\mathsf{res}_j := (sk_j, \mathsf{pp}_{\mathsf{AKE}}, \{pk_u\}_{u\in[\mu]})$*, respectively. Here* μ *is the total number of users. After execution,* P_i *outputs a flag* $\Psi_i \in \{\emptyset, \mathbf{accept}, \mathbf{reject}\}$*, and a session key* k_i *(*k_i *might be the empty string* $\emptyset$*), and* P_j *outputs* (Ψ_j, k_j) *similarly.*

Correctness of AKE. For any distinct and honest parties P_i and P_j, they share the same session key after the execution of $\mathsf{AKE.Protocol}(P_i(\mathsf{res}_i) \rightleftharpoons P_j(\mathsf{res}_j))$, i.e., $\Psi_i = \Psi_j = \mathbf{accept}, k_i = k_j \neq \emptyset$.

3.2 Security Model of AKE

We will adapt the security model formalized by [2, 28, 19], which in turn followed the model proposed by Bellare and Rogaway [5]. We also include replay attacks [29] and multiple test queries with respect to the same random bit [23].

First, we will define oracles and their static variables in the model. Then we describe the security experiment and the corresponding security notions.

Oracles. Suppose there are at most μ users $P_1, P_2, ..., P_\mu$, and each user will involve at most ℓ instances. P_i is formalized by a series of oracles, $\pi_i^1, \pi_i^2, ..., \pi_i^\ell$. Oracle π_i^s formalizes P_i's execution of the s-th protocol instance.

Each oracle π_i^s has access to P_i's resource $\mathsf{res}_i := (sk_i, \mathsf{pp}_{\mathsf{AKE}}, \mathsf{PKList} := \{pk_u\}_{u\in[\mu]})$. π_i^s also has its own variables $\mathsf{var}_i^s := (\mathsf{st}_i^s, \mathsf{Pid}_i^s, k_i^s, \Psi_i^s)$.

- st_i^s: State information that has to be stored between two rounds in order to execute the protocol.
- Pid_i^s: The intended communication peer's identity.

- $k_i^s \in \mathcal{K}$: The session key computed by π_i^s. Here $\mathcal{K}$ is the session key space. We assume that $\emptyset \in \mathcal{K}$.
- $\Psi_i^s \in \{\emptyset, \mathbf{accept}, \mathbf{reject}\}$: Ψ_i^s indicates whether π_i^s has completed the protocol execution and accepted k_i^s.

At the beginning, $(\mathsf{st}_i^s, \mathsf{Pid}_i^s, k_i^s, \Psi_i^s)$ are initialized to $(\emptyset, \emptyset, \emptyset, \emptyset)$. We declare that $k_i^s \neq \emptyset$ if and only if $\Psi_i^s = \mathbf{accept}$.

Security Experiment. To define the security notion of AKE, we first formalize the security experiment $\mathsf{Exp}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ with the help of the oracles defined above. $\mathsf{Exp}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ is a game played between an AKE challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. $\mathcal{C}$ will simulate the executions of the ℓ protocol instances for each of the μ users with oracles π_i^s. We give a formal description in Figure 4.

Adversary $\mathcal{A}$ may copy, delay, erase, replay, and interpolate the messages transmitted in the network. This is formalized by the query Send to oracle π_i^s. With Send, $\mathcal{A}$ can send arbitrary messages to any oracle π_i^s. Then π_i^s will execute the AKE protocol according to the protocol specification for P_i. The $\mathsf{StateReveal}(i, s)$ oracle allows $\mathcal{A}$ to reveal π_i^s's session state.

We also allow the adversary to observe session keys of its choices. This is reflected by a $\mathsf{SessionKeyReveal}$ query to oracle π_i^s.

A $\mathsf{Corrupt}$ query allows $\mathcal{A}$ to corrupt a party P_i and get its long-term secret key sk_i. With a $\mathsf{RegisterCorrupt}$ query, $\mathcal{A}$ can register a new party without public key certification. The public key is then known to all other users.

We introduce a Test query to formalize the pseudorandomness of k_i^s. Therefore, the challenger chooses a bit $b \leftarrow_{\$} \{0, 1\}$ at the beginning of the experiment. When $\mathcal{A}$ issues a Test query for π_i^s, the oracle will return $\perp$ if the session key k_i^s is not generated yet. Otherwise, π_i^s will return k_i^s or a truly random key, depending on b. The task of $\mathcal{A}$ is to tell whether the key is the true session key or a random key. The adversary is allowed to make multiple test queries.

Formally, the queries by $\mathcal{A}$ are described as follows.

- $\mathsf{Send}(i, s, j, \mathsf{msg})$: If $\mathsf{msg} = \top$, it means that $\mathcal{A}$ asks oracle π_i^s to send the first protocol message to P_j. Otherwise, $\mathcal{A}$ impersonates P_j to send message msg to π_i^s. Then π_i^s executes the AKE protocol with msg as P_i does, computes a message msg', and updates its own variables $\mathsf{var}_i^s = (\mathsf{st}_i^s, \mathsf{Pid}_i^s, k_i^s, \Psi_i^s)$. The output message msg' is returned to $\mathcal{A}$.
  If $\mathsf{Send}(i, s, j, \mathsf{msg})$ is the τ-th query asked by $\mathcal{A}$ and π_i^s changes Ψ_i^s to **accept** after that, then we say that π_i^s is τ-*accepted*.
- $\mathsf{Corrupt}(i)$: $\mathcal{C}$ reveals party P_i's long-term secret key sk_i to $\mathcal{A}$. After corruption, $\pi_i^1, ..., \pi_i^\ell$ will stop answering any query from $\mathcal{A}$.
  If $\mathsf{Corrupt}(i)$ is the τ-th query asked by $\mathcal{A}$, we say that P_i is τ-*corrupted*.
  If $\mathcal{A}$ has never asked $\mathsf{Corrupt}(i)$, we say that P_i is ∞-*corrupted*.
- $\mathsf{RegisterCorrupt}(i, pk_i)$: It means that $\mathcal{A}$ registers a new party P_i $(i > \mu)$. $\mathcal{C}$ distributes (P_i, pk_i) to all users. In this case, we say that P_i is 0-*corrupted*.
- $\mathsf{StateReveal}(i, s)$: The query means that $\mathcal{A}$ asks $\mathcal{C}$ to reveal π_i^s's session state. $\mathcal{C}$ returns st_i^s to $\mathcal{A}$.

$\mathsf{Exp}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$:
$\mathsf{pp}_{\mathsf{AKE}} \leftarrow \mathsf{AKE.Setup}$
For $i \in [\mu]$:
  $(pk_i, sk_i) \leftarrow \mathsf{AKE.Gen}(\mathsf{pp}_{\mathsf{AKE}}, P_i)$;
  $crp_i := \mathbf{false}$ //Corruption variable
$\mathsf{PKList} := \{pk_i\}_{i \in [\mu]}$; $b \leftarrow_{\$} \{0,1\}$
For $(i,s) \in [\mu] \times [\ell]$:
  $\mathsf{var}_i^s := (\mathsf{st}_i^s, \mathsf{Pid}_i^s, k_i^s, \Psi_i^s) := (\emptyset, \emptyset, \emptyset, \emptyset)$;
  $\mathsf{Aflag}_i^s := \mathbf{false}$ //Whether Pid_i^s is corrupted when π_i^s accepts
  $T_i^s := \mathbf{false}$; $kRev_i^s := \mathbf{false}$ // Test, Key Reveal variables
  $stRev_i^s := \mathbf{false}$, $FirstAcc_i^s := \emptyset$
  // State Reveal & First Acceptance variables
$b^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{AKE}}(\cdot)}(\mathsf{pp}_{\mathsf{AKE}}, \mathsf{PKList})$

$\mathsf{Win}_{\mathsf{Auth}} := \mathbf{false}$
$\mathsf{Win}_{\mathsf{Auth}} := \mathbf{true}$, If $\exists (i,s) \in [\mu] \times [\ell]$ s.t.
(1) $\Psi_i^s = \mathbf{accept}$ //π_i^s is τ-accepted
(2) $\mathsf{Aflag}_i^s = \mathbf{false}$ //P_j is $\hat{\tau}$-corrupted with $j := \mathsf{Pid}_i^s$ and $\hat{\tau} > \tau$
(3) (3.1) $\vee$ (3.2) $\vee$ (3.3). Let $j := \mathsf{Pid}_i^s$
  (3.1) $\nexists\, t \in [\ell]$ s.t. $\mathsf{Partner}(\pi_i^s \leftarrow \pi_j^t)$
  (3.2) $\exists\, t \in [\ell], (j', t') \in [\mu] \times [\ell]$ with $(j,t) \neq (j',t')$ s.t.
    $\mathsf{Partner}(\pi_i^s \leftarrow \pi_j^t) \wedge \mathsf{Partner}(\pi_i^s \leftarrow \pi_{j'}^{t'})$
  (3.3) $\exists\, t \in [\ell], (i', s') \in [\mu] \times [\ell]$ with $(i,s) \neq (i',s')$ s.t.
    $\mathsf{Partner}(\pi_i^s \leftarrow \pi_j^t) \wedge \mathsf{Partner}(\pi_{i'}^{s'} \leftarrow \pi_j^t)$ //Replay attacks

$\mathsf{Win}_{\mathsf{Ind}} := \mathbf{false}$
If $b^* = b$:
  $\mathsf{Win}_{\mathsf{Ind}} := \mathbf{true}$; Return 1
Else: Return 0

$\mathsf{Partner}(\pi_i^s \leftarrow \pi_j^t)$: //Checking whether $\mathsf{Partner}(\pi_i^s \leftarrow \pi_j^t)$
If π_i^s sent the first message and $k_i^s = \mathsf{K}(\pi_i^s, \pi_j^t) \neq \emptyset$: Return 1
If π_i^s received the first message and $k_i^s = \mathsf{K}(\pi_j^t, \pi_i^s) \neq \emptyset$: Return 1
Return 0

$\pi_i^s(\mathsf{msg}, j)$:
//π_i^s executes AKE according to the protocol specification
If $\mathsf{Pid}_i^s = \emptyset$: $\mathsf{Pid}_i^s := j$
If $\mathsf{Pid}_i^s = j$:
  π_i^s receives msg and uses $\mathsf{res}_i, \mathsf{var}_i^s$ to generate the next message msg' of AKE, and updates $(\mathsf{st}_i^s, \mathsf{Pid}_i^s, k_i^s, \Psi_i^s)$;
  If $\mathsf{msg} = \top$: π_i^s generates the first message msg' as initiator;
  If msg is the last message of AKE: $\mathsf{msg}' := \emptyset$;
  Return msg'
If $\mathsf{Pid}_i^s \neq j$: Return $\perp$

$\mathcal{O}_{\mathsf{AKE}}(\mathsf{query})$:
If query=$\mathsf{RegisterCorrupt}(u, pk_u)$:
  If $u \in [\mu]$: Return $\perp$
  $\mathsf{PKList} := \mathsf{PKList} \cup \{pk_u\}$
  $crp_u := \mathbf{true}$
  Return PKList

$\mathcal{O}_{\mathsf{AKE}}(\mathsf{query})$:
If query=$\mathsf{Send}(i, s, j, \mathsf{msg})$:
  If $\Psi_i^s = \mathbf{accept}$: Return $\perp$
  $\mathsf{msg}' \leftarrow \pi_i^s(\mathsf{msg}, j)$
  If $\Psi_i^s = \mathbf{accept}$:
    If $crp_j = \mathbf{true}$: $\mathsf{Aflag}_i^s := \mathbf{true}$;
    // Determine whether π_i^s accepts before its partner
    If $crp_j = \mathbf{false} \wedge \exists t \in [\ell]$ s.t. $\mathsf{Partner}(\pi_i^s \leftarrow \pi_j^t)$:
      If $\Psi_j^t \neq \mathbf{accept}$:
        $FirstAcc_i^s := \mathbf{true}$; $FirstAcc_j^t := \mathbf{false}$
      If $\Psi_j^t = \mathbf{accept}$:
        $FirstAcc_i^s := \mathbf{false}$; $FirstAcc_j^t := \mathbf{true}$
  Return msg'

If query=$\mathsf{Corrupt}(i)$:
  If $i \notin [\mu]$: Return $\perp$
  For $s \in [\ell]$
    If $FirstAcc_i^s = \mathbf{false} \wedge stRev_i^s = \mathbf{true}$:
      If $T_i^s = \mathbf{true}$: Return $\perp$; //avoid **TA6**
      If $\exists t \in [\ell]$ s.t. $\mathsf{Partner}(\pi_j^t \leftarrow \pi_i^s)$:
        If $T_j^t = \mathbf{true}$: Return $\perp$; //avoid **TA7**
  $crp_i := \mathbf{true}$
  Return sk_i

If query=$\mathsf{SessionKeyReveal}(i, s)$:
  If $\Psi_i^s \neq \mathbf{accept}$: Return $\perp$
  If $T_i^s = \mathbf{true}$: Return $\perp$ //avoid **TA2**
  Let $j := \mathsf{Pid}_i^s$
  If $\exists t \in [\ell]$ s.t. $\mathsf{Partner}(\pi_i^s \leftrightarrow \pi_j^t)$:
    If $T_j^t = \mathbf{true}$: Return $\perp$ //avoid **TA4**
  $kRev_i^s := \mathbf{true}$; Return k_i^s

If query=$\mathsf{StateReveal}(i, s)$
  If $FirstAcc_i^s = \mathbf{false} \wedge crp_i = \mathbf{true}$:
    If $T_i^s = \mathbf{true}$: Return $\perp$; //avoid **TA6**
    Let $j := \mathsf{Pid}_i^s$
    If $\exists t \in [\ell]$ s.t. $\mathsf{Partner}(\pi_j^t \leftarrow \pi_i^s)$:
      If $T_j^t = \mathbf{true}$: Return $\perp$; //avoid **TA7**
  $stRev_i^s := \mathbf{true}$; Return st_i^s

If query=$\mathsf{Test}(i, s)$:
  If $\Psi_i^s \neq \mathbf{accept} \vee \mathsf{Aflag}_i^s = \mathbf{true} \vee kRev_i^s = \mathbf{true}$
  $\vee\, T_i^s = \mathbf{true}$: Return $\perp$ //avoid **TA1**, **TA2**, **TA3**
  If $FirstAcc_i^s = \mathbf{false}$:
    If $crp_i = \mathbf{true} \wedge stRev_i^s = \mathbf{true}$:
      Return $\perp$ //avoid **TA6**
  Let $j := \mathsf{Pid}_i^s$
  If $\exists t \in [\ell]$ s.t. $\mathsf{Partner}(\pi_i^s \leftrightarrow \pi_j^t)$:
    If $kRev_j^t = \mathbf{true} \vee T_j^t = \mathbf{true}$:
      Return $\perp$ //avoid **TA4**, **TA5**
  If $\exists t \in [\ell]$ s.t. $\mathsf{Partner}(\pi_i^s \leftarrow \pi_j^t)$:
    If $FirstAcc_j^t = \mathbf{false} \wedge crp_j = \mathbf{true}$
    $\wedge\, stRev_j^t = \mathbf{true}$: Return $\perp$ //avoid **TA7**
  $T_i^s := \mathbf{true}$; $k_0 := k_i^s$; $k_1 \leftarrow_{\$} \mathcal{K}$; Return k_b

Fig. 4. The security experiments $\mathsf{Exp}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$, $\boxed{\mathsf{Exp}^{\mathsf{replay}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}}$ (both without red text) and $\boxed{\mathsf{Exp}^{\mathsf{replay,\,state}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}}$ (with red text). The list of trivial attacks is given in Table 2.

– $\mathsf{SessionKeyReveal}(i, s)$: The query means that $\mathcal{A}$ asks $\mathcal{C}$ to reveal π_i^s's session key. If $\Psi_i^s \neq \mathbf{accept}$, $\mathcal{C}$ returns $\perp$. Otherwise, $\mathcal{C}$ returns the session key k_i^s of π_i^s.
– $\mathsf{Test}(i, s)$: If $\Psi_i^s \neq \mathbf{accept}$, $\mathcal{C}$ returns $\perp$. Otherwise, $\mathcal{C}$ sets $k_0 = k_i^s$, samples $k_1 \leftarrow_\$ \mathcal{K}$, and returns k_b to $\mathcal{A}$. We require that $\mathcal{A}$ can ask $\mathsf{Test}(i, s)$ to each oracle π_i^s only once.

Informally, the pseudorandomness of k_i^s asks that any PPT adversary $\mathcal{A}$ with access to $\mathsf{Test}(i, s)$ cannot distinguish k_i^s from a uniformly random key. Yet, we have to exclude some trivial attacks. We will define them later and first introduce partnering.

Definition 7 (Original Key [28]). *For two oracles π_i^s and π_j^t, the original key, denoted as $\mathsf{K}(\pi_i^s, \pi_j^t)$, is the session key computed by the two peers of the protocol under a passive adversary only, where π_i^s is the initiator.*

Remark 1. We note that $\mathsf{K}(\pi_i^s, \pi_j^t)$ is determined by the identities of P_i and P_j and the internal randomness.

Definition 8 (Partner [28]). *Let $\mathsf{K}(\cdot, \cdot)$ denote the original key function. We say that an oracle π_i^s is partnered to π_j^t, denoted as $\mathsf{Partner}(\pi_i^s \leftarrow \pi_j^t)$[3], if one of the following requirements holds:*

– *π_i^s has sent the first message and $k_i^s = \mathsf{K}(\pi_i^s, \pi_j^t) \neq \emptyset$, or*
– *π_i^s has received the first message and $k_i^s = \mathsf{K}(\pi_j^t, \pi_i^s) \neq \emptyset$.*

We write $\mathsf{Partner}(\pi_i^s \leftrightarrow \pi_j^t)$ if $\mathsf{Partner}(\pi_i^s \leftarrow \pi_j^t)$ and $\mathsf{Partner}(\pi_j^t \leftarrow \pi_i^s)$.

Trivial Attacks. In order to prevent the adversary from trivial attacks, we keep track of the following variables for each party P_i and oracle π_i^s:

– crp_i: whether P_i is corrupted.
– Aflag_i^s: whether the intended partner is corrupted when π_i^s accepts.
– T_i^s: whether π_i^s was tested.
– $kRev_i^s$: whether the session key k_i^s was revealed.
– $stRev_i^s$: whether the session state st_i^s was revealed.
– $FirstAcc_i^s$: whether P_i or its partner is the first to accept the key in the session.

Based on that we give a list of trivial attacks **TA1-TA7** in Table 2.

Remark 2. We introduced variable $FirstAcc$ to indicate whether the party or its partner is the first to accept the key. This is used to determine whether the state of an oracle is allowed to be revealed when the oracle or its partner is tested.

[3] The arrow notion $\pi_i^s \leftarrow \pi_j^t$ means π_i^s (not necessarily π_j^t) has computed and accepted the original key.

– In general, the session key of the party which accepts the key after its partner (i.e., $FirstAcc = \textbf{false}$), by the correctness of AKE, must be identical to its partner's. Thus, the session key is fully determined by the state and long-term key of that party (as well as transcripts).
– However, the session key of the party which accepts the key before its partner (i.e., $FirstAcc = \textbf{true}$) might involve fresh randomness beyond its state and long-term key.

Thus, it is a trivial attack to reveal the state and the long-term key of the same oracle, if the oracle or its partner is tested and the oracle accepts the key after its partner (i.e., $FirstAcc = \textbf{false}$). This is a minimal trivial attack regarding state reveal[10], and it is formalized as **TA6** and **TA7** in Table 2.

The following definition also captures replay attacks. Given $\mathsf{Partner}(\pi_{i'}^{s'} \leftarrow \pi_j^t)$, a successful replay attack means that $\mathcal{A}$ resends to π_i^s the messages, which were sent from π_j^t to $\pi_{i'}^{s'}$, and π_i^s is fooled to compute a session key, i.e., $\mathsf{Partner}(\pi_i^s \leftarrow \pi_j^t)$. Note that a stateless 2-pass AKE protocol cannot be secure against replay attacks [29]. Therefore, we also define security without replay attacks in Definition 11.

Furthermore, we distinguish between security with state reveals (Definition 9) and without state reveals (Definition 10), where in the latter the adversary cannot query the session state of an oracle.

Table 2. Trivial attacks **TA1-TA7** for security experiments $\mathsf{Exp}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$, $\mathsf{Exp}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}^{\mathsf{replay}}$ and $\mathsf{Exp}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}^{\mathsf{replay,\,state}}$, where **TA6** and **TA7** are only defined in $\mathsf{Exp}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}^{\mathsf{replay,\,state}}$. Note that "$\mathsf{Aflag}_i^s = \textbf{false}$" is implicitly contained in **TA2-TA7** because of **TA1**.

Types	Trivial attacks	Explanation
TA1	$T_i^s = \textbf{true} \wedge \mathsf{Aflag}_i^s = \textbf{true}$	π_i^s is tested but π_i^s's partner is corrupted when π_i^s accepts session key k_i^s
TA2	$T_i^s = \textbf{true} \wedge kRev_i^s = \textbf{true}$	π_i^s is tested and its session key k_i^s is revealed
TA3	$T_i^s = \textbf{true}$ when $\mathsf{Test}(i,s)$ is queried	$\mathsf{Test}(i,s)$ is queried at least twice
TA4	$T_i^s = \textbf{true} \wedge \mathsf{Partner}(\pi_i^s \leftrightarrow \pi_j^t) \wedge kRev_j^t = \textbf{true}$	π_i^s is tested, π_i^s and π_j^t are partnered to each other, and π_j^t's session key k_j^t is revealed
TA5	$T_i^s = \textbf{true} \wedge \mathsf{Partner}(\pi_i^s \leftrightarrow \pi_j^t) \wedge T_j^t = \textbf{true}$	π_i^s is tested, π_i^s and π_j^t are partnered to each other, and π_j^t is tested
TA6	$T_i^s = \textbf{true} \wedge FirstAcc_i^s = \textbf{false}$ $\wedge\, stRev_i^s = \textbf{true} \wedge crp_i = \textbf{true}$	π_i^s is tested, π_i^s accepts its key after its partner, and π_i^s is both corrupted and has its state st_i^s revealed
TA7	$T_i^s = \textbf{true} \wedge \mathsf{Partner}(\pi_i^s \leftarrow \pi_j^t)$ $\wedge\, FirstAcc_j^t = \textbf{false} \wedge stRev_j^t = \textbf{true} \wedge crp_j = \textbf{true}$	π_i^s is tested, π_i^s accepts its session key before its partner, but its partner π_j^t is both corrupted and state revealed

Definition 9 (Security of AKE with Replay Attacks and State Reveal). *Let μ be the number of users and ℓ the maximum number of protocol executions*

[10] It is also possible to define the trivial attack regardless of $FirstAcc$, but our definition of **TA6** and **TA7** is minimal and makes our security model stronger.

per user. The security experiment $\mathsf{Exp}^{\mathsf{replay,\,state}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ *(see Fig. 4) is played between the challenger* $\mathcal{C}$ *and the adversary* $\mathcal{A}$*.*

1. $\mathcal{C}$ *runs* $\mathsf{AKE.Setup}$ *to get AKE public parameter* $\mathsf{pp}_{\mathsf{AKE}}$*.*
2. *For each party* P_i*,* $\mathcal{C}$ *runs* $\mathsf{AKE.Gen}(\mathsf{pp}_{\mathsf{AKE}}, P_i)$ *to get the long-term key pair* (pk_i, sk_i)*. Next it chooses a random bit* $b \leftarrow_{\$} \{0,1\}$ *and provides* $\mathcal{A}$ *with the public parameter* $\mathsf{pp}_{\mathsf{AKE}}$ *and the list of public keys* $\mathsf{PKList} := \{pk_i\}_{i\in[\mu]}$*.*
3. $\mathcal{A}$ *asks* $\mathcal{C}$ Send*,* $\mathsf{Corrupt}$*,* $\mathsf{RegisterCorrupt}$*,* $\mathsf{SessionKeyReveal}$*,* $\mathsf{StateReveal}$ *and* Test *queries adaptively.*
4. *At the end of the experiment,* $\mathcal{A}$ *terminates with an output* b^**.*

- **Strong Authentication.** *Let* $\mathsf{Win}_{\mathsf{Auth}}$ *denote the event that* $\mathcal{A}$ *breaks authentication in the security experiment.* $\mathsf{Win}_{\mathsf{Auth}}$ *happens iff* $\exists (i,s) \in [\mu]\times[\ell]$ *s.t.*
  (1) π_i^s *is* τ*-accepted.*
  (2) P_j *is* $\hat{\tau}$*-corrupted with* $j := \mathsf{Pid}_i^s$ *and* $\hat{\tau} > \tau$*.*
  (3) *Either (3.1) or (3.2) or (3.3) happens*[11]*. Let* $j := \mathsf{Pid}_i^s$*.*
    (3.1) *There is no oracle* π_j^t *that* π_i^s *is partnered to.*
    (3.2) *There exist two distinct oracles* π_j^t *and* $\pi_{j'}^{t'}$*, to which* π_i^s *is partnered.*
    (3.3) *There exist two oracles* $\pi_{i'}^{s'}$ *and* π_j^t *with* $(i', s') \neq (i, s)$*, such that both* π_i^s *and* $\pi_{i'}^{s'}$ *are partnered to* π_j^t*.*
- **Indistinguishability.** *Let* $\mathsf{Win}_{\mathsf{Ind}}$ *denote the event that* $\mathcal{A}$ *breaks indistinguishability in* $\mathsf{Exp}^{\mathsf{replay,\,state}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ *above. Let* b^* *be* $\mathcal{A}$*'s output. Then* $\mathsf{Win}_{\mathsf{Ind}}$ *happens iff* $b^* = b$*. Trivial attacks are already considered during the execution of the experiment. A list of trivial attacks is given in Table 2.*

Note that $\mathsf{Exp}^{\mathsf{replay,\,state}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}} \Rightarrow 1$ *iff* $\mathsf{Win}_{\mathsf{Ind}}$ *happens. Hence, the advantage of* $\mathcal{A}$ *is defined as*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{replay,\,state}}_{\mathsf{AKE},\mu,\ell}(\mathcal{A}) :&= \max\{\Pr[\mathsf{Win}_{\mathsf{Auth}}], |\Pr[\mathsf{Win}_{\mathsf{Ind}}] - 1/2|\}\\ &= \max\{\Pr[\mathsf{Win}_{\mathsf{Auth}}], |\Pr[\mathsf{Exp}^{\mathsf{replay,\,state}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}} \Rightarrow 1] - 1/2|\}.\end{aligned}$$

Definition 10 (Security of AKE with Replay Attacks and without State Reveal). *The security experiment* $\mathsf{Exp}^{\mathsf{replay}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ *(see Fig. 4) is defined like* $\mathsf{Exp}^{\mathsf{replay,\,state}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ *except that we disallow state reveal queries. Similarly, the advantage of an adversary* $\mathcal{A}$ *in* $\mathsf{Exp}^{\mathsf{replay}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ *is defined as*

$$\mathsf{Adv}^{\mathsf{replay}}_{\mathsf{AKE},\mu,\ell}(\mathcal{A}) := \max\{\Pr[\mathsf{Win}_{\mathsf{Auth}}], |\Pr[\mathsf{Exp}^{\mathsf{replay}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}} \Rightarrow 1] - 1/2|\}.$$

Definition 11 (Security of AKE without Replay Attack and State Reveal). *The security experiment* $\mathsf{Exp}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ *(see Fig. 4) is defined like* $\mathsf{Exp}^{\mathsf{replay,\,state}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$

[11] Given (1) $\wedge$ (2), (3.1) indicates a successful impersonation of P_j, (3.2) suggests one instance of P_i has multiple partners, and (3.3) corresponds to a successful replay attack.

except that we disallow replay attacks and state reveal queries. Similarly, the advantage of an adversary $\mathcal{A}$ in $\mathsf{Exp}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ is defined as

$$\mathsf{Adv}_{\mathsf{AKE},\mu,\ell}(\mathcal{A}) := \max\{\Pr[\mathsf{Win}_{\mathsf{Auth}}], |\Pr[\mathsf{Exp}_{\mathsf{AKE},\mu,\ell,\mathcal{A}} \Rightarrow 1] - 1/2|\}.$$

Remark 3 (Perfect Forward Security and KCI Resistance). The security model of AKE supports (perfect) forward security (a.k.a. forward secrecy [20]). That is, if P_i or its partner P_j has been corrupted at some moment, then the exchanged session keys computed before the corruption remain hidden from the adversary. Meanwhile, π_i^s may be corrupted before $\mathsf{Test}(i,s)$, which provides resistance to key-compromise impersonation (KCI) attacks [25].

4 AKE Protocols

We construct AKE protocols $\mathsf{AKE}_{\mathsf{2msg}}$, $\mathsf{AKE}_{\mathsf{3msg}}$ and $\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$ from a signature scheme SIG and a key encapsulation mechanism KEM. Additionally, we use a symmetric encryption scheme SE with key space $\mathcal{K}_{\mathsf{SE}}$ to encrypt the state in protocol $\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$. Apart from that, $\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$ and $\mathsf{AKE}_{\mathsf{3msg}}$ are the same. The protocols are given in Figure 5.

The setup algorithm generates the public parameter $\mathsf{pp}_{\mathsf{AKE}} := (\mathsf{pp}_{\mathsf{SIG}}, \mathsf{pp}_{\mathsf{KEM}})$ by running $\mathsf{SIG.Setup}$ and $\mathsf{KEM.Setup}$. The key generation algorithm inputs the public parameter and a party P_i and generates a signature key pair (vk_i, ssk_i). In $\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$, it also chooses a symmetric key s_i uniformly from the key space $\mathcal{K}_{\mathsf{SE}}$. It returns the public key vk_i and the secret key (ssk_i, s_i).

The protocol is executed between two parties P_i and P_j. Each party has access to their own resources res_i and res_j which contain the corresponding secret key, the public parameter and a list PKList consisting of the public keys of all parties. Each party initializes its local variables Ψ_i, k_i and st_i with the empty string. To initiate a session in $\mathsf{AKE}_{\mathsf{3msg}}$ and $\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$, the party P_j chooses a bitstring N uniformly from $\{0,1\}^\lambda$ and sends it to P_i. The next message and the first message in protocol $\mathsf{AKE}_{\mathsf{2msg}}$ is sent by P_i. It generates an ephemeral key pair $(\hat{pk}, \hat{sk})$ by running $\mathsf{KEM.Gen}(\mathsf{pp}_{\mathsf{KEM}})$ and computes a signature σ_1 over the identities of P_i and P_j, the ephemeral public key and the nonce (only in $\mathsf{AKE}_{\mathsf{3msg}}$ and $\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$). When using state encryption, it also encrypts the ephemeral secret key using its symmetric key s_i and stores the ciphertext in st_i. It then sends $(\hat{pk}, \sigma_1)$ to P_j. P_j verifies the signature using vk_i and rejects if it is not valid. Otherwise, it continues the protocol by computing $(c, K) \leftarrow_{\$} \mathsf{Encap}(\hat{pk})$. It computes a signature σ_2 over the identities as well as the previous message, c and the nonce (only in $\mathsf{AKE}_{\mathsf{3msg}}$ and $\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$). P_j accepts the session key and sets k_j to K. It sends (c, σ_2) to P_i. P_i verifies the signature and rejects if it is invalid. Otherwise, it retrieves the ephemeral secret key by decrypting the state, computes the session key K from c and accepts.

Correctness. Correctness of $\mathsf{AKE}_{\mathsf{2msg}}$, $\mathsf{AKE}_{\mathsf{3msg}}$ and $\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$ follows directly from the correctness of SIG, KEM and SE.

$\underline{\mathsf{AKE.Setup}}$	$\underline{\mathsf{AKE.Gen}(\mathsf{pp}_{\mathsf{AKE}}, P_i)}$
$\mathsf{pp}_{\mathsf{SIG}} \leftarrow\!\!\$\ \mathsf{SIG.Setup}$	$(vk_i, ssk_i) \leftarrow\!\!\$\ \mathsf{SIG.Gen}(\mathsf{pp}_{\mathsf{SIG}})$
$\mathsf{pp}_{\mathsf{KEM}} \leftarrow\!\!\$\ \mathsf{KEM.Setup}$	$\mathsf{s}_i \leftarrow\!\!\$\ \mathcal{K}_{\mathsf{SE}}$
Return $\mathsf{pp}_{\mathsf{AKE}} := (\mathsf{pp}_{\mathsf{SIG}}, \mathsf{pp}_{\mathsf{KEM}})$	Return $(vk_i, (ssk_i, \mathsf{s}_i))$

$\underline{\mathsf{AKE.Protocol}(P_i \rightleftharpoons P_j)}$

$P_i(\mathsf{res}_i)$

$\mathsf{res}_i = (ssk_i, \mathsf{s}_i, \mathsf{pp}_{\mathsf{AKE}}, \mathsf{PKList} = \{vk_u\}_{u\in[\mu]})$

$P_j(\mathsf{res}_j)$

$\mathsf{res}_j = (ssk_j, \mathsf{s}_j, \mathsf{pp}_{\mathsf{AKE}}, \mathsf{PKList} = \{vk_u\}_{u\in[\mu]})$

P_j: $\Psi_j := \emptyset;\ k_j := \emptyset;\ \mathsf{st}_j := \emptyset$

P_j: $N \leftarrow\!\!\$\ \{0,1\}^\lambda$

$P_i \xleftarrow{N} P_j$

P_i:
$\Psi_i := \emptyset;\ k_i := \emptyset;\ \mathsf{st}_i := \emptyset$
$(\hat{pk}, \hat{sk}) \leftarrow\!\!\$\ \mathsf{KEM.Gen}(\mathsf{pp}_{\mathsf{KEM}})$
$\sigma_1 \leftarrow\!\!\$\ \mathsf{Sign}(ssk_i, (P_i, P_j, \hat{pk}, N))$
$\mathsf{st}_i \leftarrow\!\!\$\ \mathsf{E}(\mathsf{s}_i, \hat{sk})$

$P_i \xrightarrow{(\hat{pk}, \sigma_1)} P_j$

P_i keeps st_i

P_j:
If $\Psi_j \neq \emptyset$: Return $\perp$
If $\mathsf{Ver}(vk_i, (P_i, P_j, \hat{pk}, N), \sigma_1) \neq 1$:
  $\Psi_j :=$ **reject**
Else:
  $(c, K) \leftarrow\!\!\$\ \mathsf{Encap}(\hat{pk})$
  $\sigma_2 \leftarrow\!\!\$\ \mathsf{Sign}(ssk_j, (P_i, P_j, \hat{pk}, \sigma_1, c, N))$
  $k_j := K;\ \Psi_j :=$ **accept**
Return (Ψ_j, k_j)

$P_i \xleftarrow{(c, \sigma_2)} P_j$

P_i:
If $\Psi_i \neq \emptyset$: Return $\perp$
If $\mathsf{Ver}(vk_j, (P_i, P_j, \hat{pk}, \sigma_1, c, N), \sigma_2) \neq 1$:
  $\Psi_j :=$ **reject**
Else:
  $\hat{sk} \leftarrow \mathsf{D}(\mathsf{s}_i, \mathsf{st}_i)$
  $K \leftarrow \mathsf{Decap}(\hat{sk}, c)$
  $k_i := K;\ \Psi_i :=$ **accept**
Return (Ψ_i, k_i)

Fig. 5. Generic construction of $\mathsf{AKE}_{\mathsf{2msg}}$ (without red and gray parts), $\mathsf{AKE}_{\mathsf{3msg}}$ (with red and without gray parts) and $\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$ (with red and gray parts) from KEM, SIG and SE. Note that the state of P_j only consists of public parts and is therefore omitted here.

Theorem 1 (Security of $\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$ with Replay Attacks and State Reveals). *For any adversary $\mathcal{A}$ against $\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$ with replay attacks and state reveals, there exist an MU-EUF-CMA$^{\mathsf{corr}}$ adversary $\mathcal{B}_{\mathsf{SIG}}$ against SIG, an ϵ-MU-SIM adversary $\mathcal{B}_{\mathsf{KEM}}$ against KEM and an IND-mRPA adversary $\mathcal{B}_{\mathsf{SE}}$ against SE such that*

$$\mathsf{Adv}^{\mathsf{replay,\,state}}_{\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}},\mu,\ell}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{mu\text{-}sim}}_{\mathsf{KEM},\mathsf{Encap}^*,\mu\ell}(\mathcal{B}_{\mathsf{KEM}}) + 2\cdot \mathsf{Adv}^{\mathsf{mu\text{-}corr}}_{\mathsf{SIG},\mu}(\mathcal{B}_{\mathsf{SIG}})$$
$$+ 2\mu\cdot \mathsf{Adv}^{\mathsf{mrpa}}_{\mathsf{SE},\mu}(\mathcal{B}_{\mathsf{SE}}) + 2\mu\ell\cdot\epsilon + 2(\mu\ell)^2\cdot 2^{-\gamma} + \mu\ell^2\cdot 2^{-\lambda}\ ,$$

where γ is the diversity parameter of KEM and λ is the length of the nonce N in bits. Furthermore, $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\mathsf{KEM}})$, $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\mathsf{SIG}})$ and $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\mathsf{SE}})$.

We will give a proof sketch below. The formal proof is given in the full version [21].

Proof Sketch. The signatures in the protocol ensure that the adversary can only forward messages for those sessions that it wants to test. Thus the experiment can control all ephemeral public keys $\hat{pk}$ and ciphertexts c that are used for test queries. Due to the nonce, the adversary can also not replay a message containing a particular $\hat{pk}$. Thus, each $\hat{pk}$ is used in at most one test query.

A party will close a session when it accepts or rejects the session. Thus, the adversary can submit at most one ciphertext c' which is different from the ciphertext used in the test query. Using a session key reveal query, the adversary will only see at most one more key decapsulated with $\hat{sk}$.

To deal with state reveals, the adversary $\mathcal{A}$ can additionally obtain the state which is the encrypted $\hat{sk}$. The reduction must know $\hat{sk}$ in order to answer those queries. The simulatability property of KEM ensures that Encap and Encap^* are indistinguishable, even given $\hat{sk}$. So, we first switch from Encap to Encap^*. Now, we want to replace the session keys of tested sessions with random keys. Therefore, we have to do a hybrid argument over all users. In the η-th hybrid, we replace the test session keys for party P_η. We can show that this is unnoticeable using that the key K^* generated by Encap^* is statistically close to uniform even if the adversary gets to see another key for a ciphertext of its choice. We distinguish the following cases.

Case 1: The adversary corrupts P_η. For each session, the adversary can either reveal the session state or test this session. If the adversary reveals the state, we do not have to replace the session key. If the session is tested, the adversary does not know the state $\mathsf{E}(\mathsf{s}_\eta, \hat{sk})$ and thus we can replace the session key by ϵ-uniformity of Encap^*.

Case 2: The adversary does not corrupt P_η. In this case, we use that SE is IND-mRPA secure and replace $\hat{sk}$ in the encrypted state with a random secret key for this party. Then we can use ϵ-uniformity to replace all tested keys for that party with random keys, as the state does not contain any information about $\hat{sk}$. After that, we have to switch back the state encryption to encrypt the real secret key $\hat{sk}$, getting ready for the next hybrid.

After these changes, the Test oracle will always output a random key, independent of the bit b.

Overall, the proof loses a factor of 2μ only in the IND-mRPA security of the symmetric encryption scheme. All other parts are tight.

Theorem 2 (Security of $\mathsf{AKE}_{\mathsf{3msg}}$ with Replay Attacks and without State Reveals). *For any adversary $\mathcal{A}$ against $\mathsf{AKE}_{\mathsf{3msg}}$ with replay attacks and without state reveals, there exist an MU-EUF-CMA$^{\mathsf{corr}}$ adversary $\mathcal{B}_{\mathsf{SIG}}$ against SIG and an MUSC-otCCA adversary $\mathcal{B}_{\mathsf{KEM}}$ against KEM such that*

$$\mathsf{Adv}^{\mathsf{replay}}_{\mathsf{AKE}_{\mathsf{3msg}},\mu,\ell}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{musc\text{-}otcca}}_{\mathsf{KEM},\mu\ell}(\mathcal{B}_{\mathsf{KEM}}) + 2 \cdot \mathsf{Adv}^{\mathsf{mu\text{-}corr}}_{\mathsf{SIG},\mu}(\mathcal{B}_{\mathsf{SIG}}) \\ + 2(\mu\ell)^2 \cdot 2^{-\gamma} + \mu\ell^2 \cdot 2^{-\lambda} \ ,$$

where γ is the diversity parameter of KEM and λ is the length of the nonce N in bits. Furthermore, $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\mathsf{KEM}})$ and $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\mathsf{SIG}})$.

Theorem 3 (Security of $\mathsf{AKE}_{\mathsf{2msg}}$ without State Reveals and Replay Attacks). *For any adversary $\mathcal{A}$ against $\mathsf{AKE}_{\mathsf{2msg}}$ without state reveals and replay attacks, there exist an MU-EUF-CMA$^{\mathsf{corr}}$ adversary $\mathcal{B}_{\mathsf{SIG}}$ against SIG and an MUC-otCCA adversary $\mathcal{B}_{\mathsf{KEM}}$ against KEM such that*

$$\mathsf{Adv}_{\mathsf{AKE}_{\mathsf{2msg}},\mu,\ell}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{muc\text{-}otcca}}_{\mathsf{KEM},\mu\ell}(\mathcal{B}_{\mathsf{KEM}}) + \mathsf{Adv}^{\mathsf{mu\text{-}corr}}_{\mathsf{SIG},\mu}(\mathcal{B}_{\mathsf{SIG}}) + (\mu\ell)^2 \cdot 2^{-\gamma} \ ,$$

where γ is the diversity parameter of KEM. Furthermore, $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\mathsf{KEM}})$ and $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_{\mathsf{SIG}})$.

The proofs of Theorem 2 and Theorem 3 are given in the full version [21], due to space limitations.

5 Signatures with Tight Adaptive Corruptions

5.1 Pairing Groups and MDDH Assumptions

Let GGen be a pairing group generation algorithm that returns a description $\mathcal{PG} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, \mathcal{P}_1, \mathcal{P}_2, e)$ of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic groups of order q for a λ-bit prime q, $\mathcal{P}_1$ and $\mathcal{P}_2$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2$ is an efficient computable (non-degenerated) bilinear map. $\mathcal{P}_T := e(\mathcal{P}_1, \mathcal{P}_2)$ is a generator in $\mathbb{G}_T$. In this paper, we only consider Type III pairings, where $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no efficient homomorphism between them. All constructions in this paper can be easily instantiated with Type I pairings by setting $\mathbb{G}_1 = \mathbb{G}_2$ and defining the dimension k to be greater than 1.

We use the implicit representation of group elements as in [14]. For $s \in \{1, 2, T\}$ and $a \in \mathbb{Z}_q$ define $[a]_s = a\mathcal{P}_s \in \mathbb{G}_s$ as the implicit representation of a in $\mathbb{G}_s$. Similarly, 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 $\mathbb{G}_s$. $\mathsf{Span}(\mathbf{A}) := \{\mathbf{Ar} \mid \mathbf{r} \in \mathbb{Z}_q^m\} \subset \mathbb{Z}_q^n$ denotes the linear span of $\mathbf{A}$, and similarly $\mathsf{Span}([\mathbf{A}]_s) := \{[\mathbf{Ar}]_s \mid \mathbf{r} \in \mathbb{Z}_q^m\} \subset \mathbb{G}_s^n$. Note that it is efficient to compute $[\mathbf{AB}]_s$ given $([\mathbf{A}]_s, \mathbf{B})$ or $(\mathbf{A}, [\mathbf{B}]_s)$ with matching dimensions. We define $[\mathbf{A}]_1 \circ [\mathbf{B}]_2 := e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{AB}]_T$, which can be efficiently computed given $[\mathbf{A}]_1$ and $[\mathbf{B}]_2$.

We recall the definition of the Matrix Decisional Diffie-Hellman (MDDH) and related assumptions from [14].

Definition 12 (Matrix distribution). *Let $k, \ell \in \mathbb{N}$ with $\ell > k$. We call $\mathcal{D}_{\ell,k}$ a matrix distribution if it outputs matrices in $\mathbb{Z}_q^{\ell\times k}$ of full rank k in polynomial time. Let $\mathcal{D}_k := \mathcal{D}_{k+1,k}$.*

For positive integers $k, \eta, n \in \mathbb{N}^+$ and a matrix $\mathbf{A} \in \mathbb{Z}_q^{(k+\eta)\times n}$, we denote the k rows of $\mathbf{A}$ by $\overline{\mathbf{A}} \in \mathbb{Z}_q^{k\times n}$ and the lower η rows of $\mathbf{A}$ by $\underline{\mathbf{A}} \in \mathbb{Z}_q^{\eta\times n}$. Without loss of generality, we assume $\overline{\mathbf{A}}$ for $\mathbf{A} \leftarrow_{\$} \mathcal{D}_{\ell,k}$ form an invertible square matrix in $\mathbb{Z}_q^{k\times k}$. The $\mathcal{D}_{\ell,k}$-MDDH problem is to distinguish the two distributions $([\mathbf{A}], [\mathbf{Aw}])$ and $([\mathbf{A}], [\mathbf{u}])$ where $\mathbf{A} \leftarrow_{\$} \mathcal{D}_{\ell,k}$, $\mathbf{w} \leftarrow_{\$} \mathbb{Z}_q^k$ and $\mathbf{u} \leftarrow_{\$} \mathbb{Z}_q^\ell$.

Definition 13 **($\mathcal{D}_{\ell,k}$-MDDH assumption).** *Let $\mathcal{D}_{\ell,k}$ be a matrix distribution and $s \in \{1,2,T\}$. We say that the $\mathcal{D}_{\ell,k}$-MDDH assumption holds relative to* GGen *in group $\mathbb{G}_s$ if for all adversaries $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{\mathrm{MDDH}}_{\mathsf{GGen},\mathcal{D}_{\ell,k},\mathbb{G}_s}(\mathcal{A}) := |\Pr[\mathcal{A}(\mathcal{PG},[\mathbf{A}]_s,[\mathbf{Aw}]_s) \Rightarrow 1] - \Pr[\mathcal{A}(\mathcal{PG},[\mathbf{A}]_s,[\mathbf{u}]_s) \Rightarrow 1]|$$

is negligible where the probability is taken over $\mathcal{PG} \leftarrow_\$ \mathsf{GGen}(1^\lambda)$, $\mathbf{A} \leftarrow_\$ \mathcal{D}_{\ell,k}, \mathbf{w} \leftarrow_\$ \mathbb{Z}_q^k$ and $\mathbf{u} \leftarrow_\$ \mathbb{Z}_q^\ell$.

Definition 14 **(Uniform distribution).** *Let $k, \ell \in \mathbb{N}^+$ with $\ell > k$. We call $\mathcal{U}_{\ell,k}$ a uniform distribution if it outputs uniformly random matrices in $\mathbb{Z}_q^{\ell \times k}$ of rank k in polynomial time. Let $\mathcal{U}_k := \mathcal{U}_{k+1,k}$.*

Lemma 1 **($\mathcal{D}_{\ell,k}$-MDDH $\Rightarrow$ $\mathcal{U}_k$-MDDH [14]).** *Let $\ell, k \in \mathbb{N}_+$ with $\ell > k$ and let $\mathcal{D}_{\ell,k}$ be a matrix distribution. A $\mathcal{U}_k$-MDDH instance is at least as hard as an $\mathcal{D}_{\ell,k}$ instance. More precisely, for each adversary $\mathcal{A}$ there exists an adversary $\mathcal{B}$ with*

$$\mathsf{Adv}^{\mathrm{MDDH}}_{\mathsf{GGen},\mathcal{U}_k,\mathbb{G}_s}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{MDDH}}_{\mathsf{GGen},\mathcal{D}_{\ell,k},\mathbb{G}_s}(\mathcal{B})$$

and $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$.

The Kernel-Diffie-Hellman assumption ($\mathcal{D}_k$-KMDH) [31] is a (weaker) computational analogue of the $\mathcal{D}_k$-MDDH Assumption.

Definition 15 **($\mathcal{D}_k$-KMDH).** *Let $\mathcal{D}_k$ be a matrix distribution. We say that the $\mathcal{D}_k$-Kernel Diffie-Hellman ($\mathcal{D}_k$-KMDH) assumption holds relative to a prime order group $\mathbb{G}_s$ for $s \in \{1,2\}$ if for all PPT adversaries $\mathcal{A}$,*

$$\mathsf{Adv}^{\mathrm{KMDH}}_{\mathsf{GGen},\mathcal{D}_k,\mathbb{G}_s}(\mathcal{A}) : = \Pr[\mathbf{c}^\top \mathbf{A} = \mathbf{0} \wedge \mathbf{c} \neq \mathbf{0} \mid [\mathbf{c}]_{3-s} \leftarrow_\$ \mathcal{A}(\mathcal{PG},[\mathbf{A}]_s)],$$

where the probabilities are taken over $\mathcal{PG} \leftarrow_\$ \mathsf{GGen}(1^\lambda)$ and $\mathbf{A} \leftarrow_\$ \mathcal{D}_k$.

5.2 Previous Schemes with Tight Adaptive Corruptions

We will construct a signature scheme with tight MU-EUF-CMA$^{\mathsf{corr}}$ security and only small constant number of elements in signatures. Such a scheme has been proposed in [2, Section 2.3] (called $\mathsf{SIG_C}$), but we identify a gap in their proof. We now present the gap in their security proof and why we think it will be hard to close it.

The construction of $\mathsf{SIG_C}$ follows the BKP IBE schemes [6], namely, it tightly transforms an affine MAC [6] into a signature in the multi-user setting. In order to have a tightly MU-EUF-CMA$^{\mathsf{corr}}$ secure signature scheme, the underlying MAC needs to be tightly secure against adaptive corruption of its secret keys in the multi-user setting. We will now point to potential problems in formally proving it.

We abstract the underlying MAC of $\mathsf{SIG_C}$ as $\mathsf{MAC_{BHJKL}}$: For message space $\{0,1\}^\ell$, it chooses $\mathbf{A'} \leftarrow_\$ \mathcal{D}_k$ and random vectors $\mathbf{x}_{i,j} \leftarrow_\$ \mathbb{Z}_q^k$ (for $1 \leq i \leq \ell$ and $j = 0,1$). Then it defines $\mathbf{B} := \overline{\mathbf{A'}} \in \mathbb{Z}_q^{k \times k}$ and publishes system parameters

$\mathsf{pp} := ([\mathbf{B}]_1, ([\mathbf{B}^\top \mathbf{x}_{i,j}]_1)_{1 \le i \le \ell, j=0,1})$. For each user n, it chooses its MAC secret key as $[x'_n]_1 \leftarrow_\$ \mathbb{G}_1$, and its MAC tag consist of $([\mathbf{t}]_1, [u]_1)$, where

$$\begin{aligned} \mathbf{t} &= \mathbf{B}\mathbf{s} \in \mathbb{Z}_q^k \quad \text{for} \quad \mathbf{s} \leftarrow_\$ \mathbb{Z}_q^k \\ u &= x'_n + \mathbf{t}^\top \underbrace{\sum\nolimits_i \mathbf{x}_{i,\mathsf{m}_i}}_{=:\mathbf{x}(\mathsf{m})} \in \mathbb{Z}_q. \end{aligned} \tag{3}$$

In their security proof, they argue that $[u]_1$ in the MAC tagging queries is pseudorandom, given pp and some of the secret keys $[x'_n]_1$ (via the adaptive corruption queries) to an adversary.[12] In achieving this, they define a sequence of hybrids H_j for $1 \le j \le \ell$. In each H_j, they replace x'_n for each user n with $\mathsf{RF}_{n,j}(\mathsf{m}_{|j})$, where $\mathsf{RF}_{n,j} : \{0,1\}^j \to \mathbb{Z}_q$ is a random function and m is the first tagging query to user n, and generate the MAC tag of m' as

$$u = \mathsf{RF}_{n,j}(\mathsf{m'}_{|j}) + \mathbf{t}^\top \mathbf{x}(\mathsf{m}') \tag{4}$$

with $\mathbf{t}$ as in Equation (3).

In their final step (between H_ℓ and GAME 4), they argue that the distribution of $u = \mathsf{RF}_{n,\ell}(\mathsf{m}') + \mathbf{t}^\top \mathbf{x}(\mathsf{m}')$ is uniformly random (as in GAME 4) even for an unbounded adversary, given pp and adaptive corruptions. Then they conclude that H_ℓ (where $u = \mathsf{RF}_{n,\ell}(\mathsf{m}') + \mathbf{t}^\top \mathbf{x}(\mathsf{m}')$) and GAME 4 (where u is chosen uniformly at random) are identical and $\Pr[\chi_4] = \Pr[H_\ell = 1]$ (according to their notation). However, this is not the case: $\mathbf{B} \in \mathbb{Z}_q^{k \times k}$ is full-rank and thus, given $[\mathbf{B}^\top \mathbf{x}_{i,j}]_1$ in pp, $\mathbf{x}_{i,j} \in \mathbb{Z}_q^k$ is uniquely defined. (For concreteness, imagine a simple example where an (unbounded) adversary $\mathcal{A}$ only queries one MAG tag for message m for user n and then asks for the secret key $[x'_n]_1 := \mathsf{RF}_{n,\ell}(\mathsf{m})$ of user n. Then, $\mathcal{A}$ sees that $u = \mathsf{RF}_{n,\ell}(\mathsf{m}) + \mathbf{t}^\top \mathbf{x}(\mathsf{m})$ is uniquely defined by $[x'_n]_1, [\mathbf{t}]_1$ and pp in H_ℓ, while u is uniformly at random in GAME 4.) We suppose this gap is inherent, since the terms $\mathbf{B}^\top \mathbf{x}_{i,j}$ completely leak the information about $\mathbf{x}_{i,j}$. This is also the same reason why the BKP MAC cannot be used to construct a tightly secure hierarchical IBE (HIBE) (cf. [26] for more discussion).

To resolve this, we follow the tightly secure HIBE approach in [26] and choose $\mathbf{B} \leftarrow_\$ \mathbb{Z}_q^{3k \times k}$. Now, there is a non-zero kernel matrix $\mathbf{B}^\perp \in \mathbb{Z}_q^{3k \times 2k}$ for $\mathbf{B}$ (with overwhelming probability), and the mapping $\mathbf{x}_{i,j} \in \mathbb{Z}_q^{3k} \mapsto \mathbf{B}^\top \mathbf{x}_{i,j} \in \mathbb{Z}_q^k$ is lossy. In particular, the information about $\mathbf{x}_{i,j}$ in the orthogonal space of $\mathbf{B}$ is perfectly hidden from (unbounded) adversaries, given $\mathbf{B}^\top \mathbf{x}_{i,j}$.

5.3 Our Construction

Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a function chosen from a collision-resistant hash function family $\mathcal{H}$. Our signature scheme $\mathsf{SIG}_{\mathsf{MDDH}} := (\mathsf{SIG.Setup}, \mathsf{SIG.Gen}, \mathsf{Sign}, \mathsf{Ver})$ is defined in Figure 6. Correctness can be verified as

[12] This is different to the BKP IBE where $[\mathbf{B}^\top \mathbf{x}_{i,j}]_1$ and $[x'_n]_1$ are not available to an adversary.

$\underline{\mathsf{SIG.Setup}:}$
$\mathcal{PG} \leftarrow_{\$} \mathsf{GGen}$
$\mathbf{A} \leftarrow_{\$} \mathcal{D}_k$; $\mathbf{B} \leftarrow_{\$} \mathcal{U}_{3k,k}$
For $1 \leq i \leq \lambda$ and $j = 0, 1$:
  $\mathbf{x}_{i,j} \leftarrow_{\$} \mathbb{Z}_q^{3k}$; $\mathbf{Y}_{i,j} \leftarrow_{\$} \mathbb{Z}_q^{3k \times k}$
  $\mathbf{Z}_{i,j} := (\mathbf{Y}_{i,j} \parallel \mathbf{x}_{i,j}) \cdot \mathbf{A} \in \mathbb{Z}_q^{3k \times k}$
  $\mathbf{P}_{i,j} := \mathbf{B}^\top \cdot (\mathbf{Y}_{i,j} \parallel \mathbf{x}_{i,j}) \in \mathbb{Z}_q^{k \times (k+1)}$
$\mathsf{pp} := (\mathcal{PG}, [\mathbf{A}]_2, [\mathbf{B}]_1, ([\mathbf{Z}_{i,j}]_2, [\mathbf{P}_{i,j}]_1)_{1 \leq i \leq \lambda, j=0,1})$
Return pp

$\underline{\mathsf{SIG.Gen}(\mathsf{pp})}$
$x' \leftarrow_{\$} \mathbb{Z}_q$; $\mathbf{y}' \leftarrow_{\$} \mathbb{Z}_q^{1 \times k}$
$ssk := ([x']_1, [\mathbf{y}']_1)$
$vk := [\mathbf{z}']_2 := [(\mathbf{y}' \parallel x')\mathbf{A}]_2 \in \mathbb{G}_2^{1 \times k}$
Return (vk, ssk)

$\underline{\mathsf{Sign}(ssk, \mathsf{m}):}$
$\mathbf{s} \leftarrow_{\$} \mathbb{Z}_q^k$; $\mathbf{t} := \mathbf{B}\mathbf{s} \in \mathbb{Z}_q^{3k}$
$\mathsf{hm} := H(vk, \mathsf{m})$
$u := x' + \mathbf{s}^\top \mathbf{B}^\top \mathbf{x}(\mathsf{hm}) \in \mathbb{Z}_q$
$\mathbf{v} := \mathbf{y}' + \mathbf{s}^\top \mathbf{B}^\top \mathbf{Y}(\mathsf{hm}) \in \mathbb{Z}_q^{1 \times k}$
Return $\sigma := ([\mathbf{t}]_1, [u]_1, [\mathbf{v}]_1)$

$\underline{\mathsf{Ver}(vk, \mathsf{m}, \sigma := ([\mathbf{t}]_1, [u]_1, [\mathbf{v}]_1)):}$
$\mathsf{hm} := H(vk, \mathsf{m})$
If $[\mathbf{v}, u]_1 \circ [\mathbf{A}]_2 = [1]_1 \circ [\mathbf{z}']_2 + [\mathbf{t}^\top]_1 \circ [\mathbf{Z}(\mathsf{hm})]_2$:
  Return 1
Else: Return 0

Fig. 6. Our signature scheme with tight adaptive corruptions, where for $\mathsf{hm} \in \{0,1\}^\lambda$ we define the functions $\mathbf{x}(\mathsf{hm}) := \sum_{i=1}^{\lambda} \mathbf{x}_{i,\mathsf{hm}_i}$, $\mathbf{Y}(\mathsf{hm}) := \sum_{i=1}^{\lambda} \mathbf{Y}_{i,\mathsf{hm}_i}$, $\mathbf{Z}(\mathsf{hm}) := \sum_{i=1}^{\lambda} \mathbf{Z}_{i,\mathsf{hm}_i}$, and $\mathbf{P}(\mathsf{hm}) := \sum_{i=1}^{\lambda} \mathbf{P}_{i,\mathsf{hm}_i}$.

$$[\mathbf{v}, u]_1 \circ [\mathbf{A}]_2 = [(\mathbf{y}', x') \cdot \mathbf{A} + \mathbf{t}^\top \cdot (\mathbf{Y}(\mathsf{hm}) \mid \mathbf{x}(\mathsf{hm})) \cdot \mathbf{A}]_T$$

for $([\mathbf{t}]_1, [u]_1, [\mathbf{v}]_1) \leftarrow_{\$} \mathsf{Sign}(ssk, \mathsf{m})$.

Theorem 4 (Security of $\mathsf{SIG}_{\mathsf{MDDH}}$). *For any adversary $\mathcal{A}$ against the MU-EUF-CMA$^{\mathsf{corr}}$ security of $\mathsf{SIG}_{\mathsf{MDDH}}$, there are adversaries $\mathcal{B}$ against the collision resistance of $\mathcal{H}$, $\mathcal{B}_1$ against the $\mathcal{U}_{3k,k}$-MDDH assumption over $\mathbb{G}_1$ and $\mathcal{B}_2$ against the $\mathcal{D}_k$-KMDH assumption over $\mathbb{G}_2$ with*

$$\begin{aligned}\Pr[\mathsf{Exp}^{\mathsf{mu\text{-}corr}}_{\mathsf{SIG},\mu,\mathcal{A}} \Rightarrow 1] \leq & \mathsf{Adv}^{\mathsf{cr}}_{\mathcal{H}}(\mathcal{B}) + (8k\lambda + 2k)\mathsf{Adv}^{\mathrm{MDDH}}_{\mathsf{GGen},\mathcal{U}_{3k,k},\mathbb{G}_1}(\mathcal{B}_1) \\ & + \mathsf{Adv}^{\mathrm{KMDH}}_{\mathsf{GGen},\mathcal{D}_k,\mathbb{G}_2}(\mathcal{B}_2) + \frac{4\lambda + 2k + 2}{q - 1},\end{aligned}$$

where $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_1) \approx \mathbf{T}(\mathcal{B}_2)$.

Proof. We prove the tight MU-EUF-CMA$^{\mathsf{corr}}$ security of $\mathsf{SIG}_{\mathsf{MDDH}}$ with a sequence of games given in Figure 7. Let $\mathcal{A}$ be an adversary against the MU-EUF-CMA$^{\mathsf{corr}}$ security of $\mathsf{SIG}_{\mathsf{MDDH}}$, and let Win_i denote the probability that G_i returns 1.
Game G_0: G_0 is the original MU-EUF-CMA$^{\mathsf{corr}}$ security experiment $\mathsf{Exp}^{\mathsf{mu\text{-}corr}}_{\mathsf{SIG},\mu,\mathcal{A}}$ (see the full version [21] for the formal definition). In addition to the original game, we add a rejection rule if there is a collision between the forgery and a signing query, namely, $H(vk_{i^*}, \mathsf{m}^*) = H(vk_i, \mathsf{m})$ where (i, m) is one of the signing queries. By the collision resistance of H, we have

$$|\Pr[\mathsf{Exp}^{\mathsf{mu\text{-}corr}}_{\mathsf{SIG},\mu,\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{Win}_0]| \leq \mathsf{Adv}^{\mathsf{cr}}_{\mathcal{H}}(\mathcal{B}).$$

For better readability, we assume all the signing queries are distinct for the following games. If the same (i, m) is asked multiple times, we can take the first response $([\mathbf{t}]_1, [u]_1, [\mathbf{v}]_1)$ and answer the repeated queries with the re-randomization $([\mathbf{t}']_1, [u']_1, [\mathbf{v}']_1)$ as $\mathbf{t}' := \mathbf{t} + \mathbf{B}\mathbf{s}'$ (for $\mathbf{s}' \leftarrow_{\$} \mathbb{Z}_q^k$), $u' := u +$

$\mathsf{G}_0, \mathsf{G}_1, \boxed{\mathsf{G}_2}$:

$\mathcal{PG} \leftarrow_\$ \mathsf{GGen}$; $\mathbf{A} \leftarrow_\$ \mathcal{D}_k$; $\mathbf{B} \leftarrow_\$ \mathcal{U}_{3k,k}$
For $1 \le i \le \lambda$ and $j = 0, 1$:
  $\mathbf{x}_{i,j} \leftarrow_\$ \mathbb{Z}_q^{3k}$; $\mathbf{Y}_{i,j} \leftarrow_\$ \mathbb{Z}_q^{3k\times k}$
  $\mathbf{Z}_{i,j} := (\mathbf{Y}_{i,j} \parallel \mathbf{x}_{i,j}) \cdot \mathbf{A} \in \mathbb{Z}_q^{3k\times k}$
  $\mathbf{P}_{i,j} := \mathbf{B}^\top \cdot (\mathbf{Y}_{i,j} \parallel \mathbf{x}_{i,j}) \in \mathbb{Z}_q^{k\times(k+1)}$
  $\boxed{\mathbf{Z}_{i,j} \leftarrow_\$ \mathbb{Z}_q^{3k\times k}}$
  $\boxed{\mathbf{d}_{i,j} := \mathbf{B}^\top \mathbf{x}_{i,j} \in \mathbb{Z}_q^{k}}$
  $\boxed{\mathbf{E}_{i,j} := (\mathbf{B}^\top \mathbf{Z}_{i,j} - \mathbf{d}_{i,j} \cdot \underline{\mathbf{A}})\overline{\mathbf{A}}^{-1} \in \mathbb{Z}_q^{k\times k}}$
  $\boxed{\mathbf{P}_{i,j} := (\mathbf{E}_{i,j} \parallel \mathbf{d}_{i,j})}$
$\mathsf{pp} := (\mathcal{PG}, [\mathbf{A}]_2, [\mathbf{B}]_1, ([\mathbf{Z}_{i,j}]_2, [\mathbf{P}_{i,j}]_1)_{1\le i\le\lambda, j=0,1})$
For $1 \le i \le \mu$:
  $x'_i \leftarrow_\$ \mathbb{Z}_q$; $\mathbf{y}'_i \leftarrow_\$ \mathbb{Z}_q^{1\times k}$
  $\mathbf{z}'_i := (\mathbf{y}'_i \parallel x'_i)\mathbf{A} \in \mathbb{Z}_q^{1\times k}$
  $\boxed{\mathbf{z}'_i \leftarrow_\$ \mathbb{Z}_q^{1\times k};\ \mathbf{y}'_i = (\mathbf{z}'_i - x'_i \cdot \underline{\mathbf{A}})(\overline{\mathbf{A}})^{-1}}$
  $ssk_i := ([x'_i]_1, [\mathbf{y}'_i]_1)$
  $vk_i := [\mathbf{z}'_i]_2$
$(i^*, \mathsf{m}^*, \sigma^*) \leftarrow_\$ \mathcal{A}^{\mathcal{O}_{\text{SIGN}}(\cdot,\cdot), \mathcal{O}_{\text{CORR}}(\cdot)}(\mathsf{pp}, \{vk_i\}_{1\le i\le\mu})$
If $(i^* \in \mathcal{S}^{\mathsf{corr}}) \vee (m^* \in \mathcal{M}_{i^*}) \vee (\mathsf{Ver}(vk_{i^*}, m^*, \sigma^*) = 0)$:
  Return 0
$\mathsf{hm}^* := H(vk_{i^*}, \mathsf{m}^*)$
If $\exists 1 \le i \le \mu \wedge \mathsf{m} \in \mathcal{M}_i : H(vk_i, \mathsf{m}) = \mathsf{hm}^*$
  Return 0
Parse $\sigma^* := ([\mathbf{t}^*]_1, [u^*]_1, [\mathbf{v}^*]_1)$
If $[u^*]_1 \neq [x'_{i^*}]_1 + [\mathbf{t}^*]_1^\top \cdot \mathbf{x}(\mathsf{hm}^*)$
  Return 0
Return 1

$\mathcal{O}_{\text{SIGN}}(i, \mathsf{m})$:
$\mathbf{s} \leftarrow_\$ \mathbb{Z}_q^{k}$; $\mathbf{t} := \mathbf{B}\mathbf{s} \in \mathbb{Z}_q^{3k}$
$\mathsf{hm} := H(vk_i, \mathsf{m})$
$u := x'_i + \mathbf{s}^\top \mathbf{B}^\top \mathbf{x}(\mathsf{hm}) \in \mathbb{Z}_q$
$\mathbf{v} := \mathbf{y}'_i + \mathbf{s}^\top \mathbf{B}^\top \mathbf{Y}(\mathsf{hm}) \in \mathbb{Z}_q^{1\times k}$
$\boxed{\mathbf{v} := (\mathbf{z}'_i + \mathbf{t}^\top \mathbf{Z}(\mathsf{hm}) - u \cdot \underline{\mathbf{A}}) \cdot (\overline{\mathbf{A}})^{-1}}$
$\mathcal{M}_i := \mathcal{M}_i \cup \{m\}$
Return $\sigma := ([\mathbf{t}]_1, [u]_1, [\mathbf{v}]_1)$

$\mathcal{O}_{\text{CORR}}(i)$:
$\mathcal{S}^{\mathsf{corr}} := \mathcal{S}^{\mathsf{corr}} \cup \{i\}$
Return ssk_i

Fig. 7. Games used to prove Theorem 4.

$\mathbf{s}'^\top(\mathbf{B}^\top\mathbf{x}(\mathsf{hm}))$ and $\mathbf{v}' := \mathbf{v} + \mathbf{s}'^\top(\mathbf{B}^\top\mathbf{x}(\mathsf{hm}))$ and $\mathsf{hm} := H(vk_i, \mathsf{m})$. Note that this will not change the view of $\mathcal{A}$.

Game G_1: For verifying the forgery, in addition to using Ver, we use the secret $[x'_{i^*}]_1$ and $([\mathbf{x}_{j,b}]_1)_{1\le j\le\lambda}$ to check if $([\mathbf{t}^*]_1, [u^*]_1)$ in the forgery satisfies the following equation:

$$[u^*]_1 = [x'_{i^*}]_1 + [\mathbf{t}^*]_1^\top \cdot \mathbf{x}(\mathsf{hm}^*). \tag{5}$$

We note that

$$\begin{aligned} &\mathsf{Ver}(vk_{i^*}, m^*, \sigma^*) = 1 \\ \Leftrightarrow &(\mathbf{v} \parallel u) \cdot \mathbf{A} = (\mathbf{y}'_{i^*} \parallel x'_{i^*})\mathbf{A} + {\mathbf{t}^*}^\top \cdot (\mathbf{Y}(\mathsf{hm}) \parallel \mathbf{x}(\mathsf{hm})) \cdot \mathbf{A}. \end{aligned}$$

Thus, if Equation (5) does not hold, then the vector $[(\mathbf{v} \parallel u)]_1 - ([\mathbf{y}'_{i^*} \parallel x'_{i^*}]_1 + [{\mathbf{t}^*}^\top]_1 \cdot \mathbf{x}(\mathsf{hm}^*)) \in \mathbb{G}_1^{1\times(k+1)}$ is non-zero and orthogonal to $[\mathbf{A}]_2$. Therefore, we bound the difference between G_0 and G_1 with the $\mathcal{D}_k$-KMDH assumption as

$$|\Pr[\mathsf{Win}_0] - \Pr[\mathsf{Win}_1]| \le \mathsf{Adv}^{\text{KMDH}}_{\mathsf{GGen},\mathcal{D}_k,\mathbb{G}_2}(\mathcal{B}).$$

Game G_2: We do not use the values $\mathbf{Y}_{j,b}$ (for $1 \le j \le \lambda$ and $b = 0, 1$) and $\mathbf{y}'_i$ (for $1 \le i \le \mu$) to simulate G_2. We make this change by substituting all $\mathbf{Y}_{j,b}$

and $\mathbf{y}'_i$ using the formulas

$$\mathbf{Y}_{j,b}^\top = (\mathbf{Z}_{j,b} - \mathbf{x}_{j,b} \cdot \underline{\mathbf{A}})(\overline{\mathbf{A}})^{-1} \text{ and } \mathbf{y}'_i = (\mathbf{z}'_i - x'_i \cdot \underline{\mathbf{A}})(\overline{\mathbf{A}})^{-1}, \tag{6}$$

respectively. More precisely, the public parameters pp are computed by picking $\mathbf{Z}_{j,b}$ and $\mathbf{x}_{j,b}$ at random and then defining $\mathbf{Y}_{j,b}$ using Equation (6). The verification keys vk_i for user i $(1 \leq i \leq \mu)$ are computed by picking $\mathbf{z}'_i$ and x'_i at random. For $\mathcal{O}_{\text{SIGN}}(i, \mathsf{m})$, we now compute

$$\begin{aligned}\mathbf{v} &:= \mathbf{y}'_i + \mathbf{t}^\top \mathbf{Y}(\mathsf{hm}) \in \mathbb{Z}_q^{1\times k}\\ &= (\mathbf{z}'_i - x'_i \cdot \underline{\mathbf{A}})(\overline{\mathbf{A}})^{-1} + \mathbf{t}^\top(\mathbf{Z}(\mathsf{hm}) - \mathbf{x}(\mathsf{hm}) \cdot \underline{\mathbf{A}})(\overline{\mathbf{A}})^{-1}\\ &= (\mathbf{z}'_i + \mathbf{t}^\top \mathbf{Z}(\mathsf{hm}) - \underbrace{(x'_i + \mathbf{t}^\top \mathbf{x}(\mathsf{hm}))}_{=u} \cdot \underline{\mathbf{A}})(\overline{\mathbf{A}})^{-1}.\end{aligned}$$

The secret verification of the forgery can be done by knowing x'_{i^*} and $\mathbf{x}_{j,b}$.

The changes in G_2 are only conceptual, since Equations (6) are equivalent to $\mathbf{Z}_{j,b} = (\mathbf{Y}_{j,b} \parallel \mathbf{x}_{j,b})\mathbf{A}$ and $\mathbf{z}'_i = (\mathbf{y}'_i \parallel x'_i)\mathbf{A}$. Thus, we have

$$\Pr[\mathsf{Win}_1] = \Pr[\mathsf{Win}_2].$$

In order to bound $\Pr[\mathsf{Win}_2]$, consider a "message authentication code" MAC which is defined as follows.

- The public parameters consist of $\mathsf{pp}_{\mathsf{MAC}} := (\mathcal{PG}, [\mathbf{B}]_1, ([\mathbf{d}_{i,j}]_1)_{1\leq i\leq\lambda, j=0,1})$, where $\mathbf{d}_{i,j} := \mathbf{B}^\top \mathbf{x}_{i,j} \in \mathbb{Z}_q^k$ for $\mathbf{x}_{i,j} \leftarrow_\$ \mathbb{Z}_q^{3k}$ and $\mathbf{B} \leftarrow_\$ \mathcal{U}_{3k,k}$.
- The secret key is $[x']_1$.
- The MAC tag on hm is $([\mathbf{t}]_1, [u]_1)$, where $\mathbf{t} := \mathbf{B}\mathbf{s}$ and $u := x' + \mathbf{t}^\top \mathbf{x}(\mathsf{hm})$, for $\mathbf{s} \leftarrow_\$ \mathbb{Z}_q^k$.

Note that strictly speaking MAC is not a MAC since verification cannot only be done efficiently by knowing the values $\mathbf{x}_{i,j}$.

The following lemma states MU-EUF-CMA$^{\mathsf{corr}}$ security of MAC, with proof in the full version [21].

Lemma 2 (Core Lemma). *For every adversaries $\mathcal{A}$ interacting with $\mathsf{UF\text{-}CMA}^{\mathsf{corr}}$, there exists an adversary $\mathcal{B}$ against the $\mathcal{U}_{3k,k}$-MDDH assumption in $\mathbb{G}_1$ with*

$$\Pr[\mathsf{UF\text{-}CMA}_{\mathcal{A}}^{\mathsf{corr}} \Rightarrow 1] \leq (8k\lambda + 2k) \cdot \mathsf{Adv}_{\mathsf{GGen}, \mathcal{U}_{3k,k}, \mathbb{G}_1}^{\mathrm{MDDH}}(\mathcal{B}_1) + \frac{4\lambda + 2k + 2}{q-1},$$

and $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$, where Q_e is the number of $\mathcal{A}$'s queries to $\mathcal{O}_{\text{MAC}}$.

Finally, we bound the probability that the adversary wins in G_2 using our Core Lemma (Lemma 2) by constructing an adversary $\mathcal{B}_{\mathsf{MAC}}$ as in Figure 9.

$$\Pr[\mathsf{Win}_2] = \Pr[\mathsf{UF\text{-}CMA}_{\mathcal{B}_{\mathsf{MAC}}}^{\mathsf{corr}} \Rightarrow 1].$$

In order to analyze $\Pr[\mathsf{Win}_2]$ we argue as follows. The simulated pp and $(vk_i)_{1\leq i\leq\mu}$ are distributed as in G_2. Further, queries to $\mathcal{O}_{\text{SIGN}}$ and $\mathcal{O}_{\text{CORR}}$ from ssk_i can be

$\underline{\mathsf{UF\text{-}CMA}^{\mathsf{corr}}_{\mathcal{A}}:}$
$\beta = 0$
$\mathcal{PG} \leftarrow_{\$} \mathsf{GGen}$
$\mathbf{B} \leftarrow_{\$} \mathcal{U}_{3k,k}$
For $1 \leq i \leq \lambda$ and $j = 0, 1$:
  $\mathbf{x}_{i,j} \leftarrow_{\$} \mathbb{Z}_q^{3k}$
$\mathsf{pp}_{\mathsf{MAC}} := (\mathcal{PG}, [\mathbf{B}]_1, ([\mathbf{B}^\top \mathbf{x}_{i,j}]_1)_{1 \leq i \leq \lambda, j=0,1})$
For $1 \leq i \leq \mu$:
  $x'_i \leftarrow_{\$} \mathbb{Z}_q$
$\mathcal{A}^{\mathcal{O}_{\mathrm{MAC}}(\cdot), \mathcal{O}_{\mathrm{VER}}(\cdot,\cdot), \mathcal{O}'_{\mathrm{CORR}}(\cdot)}(\mathsf{pp}_{\mathsf{MAC}})$
Return β

$\underline{\mathcal{O}_{\mathrm{MAC}}(i, \mathsf{hm}):}$
$\mathcal{Q} := \mathcal{Q} \cup \{(i, \mathsf{hm})\}$
$\mathbf{s} \leftarrow_{\$} \mathbb{Z}_q^k$; $\mathbf{t} := \mathbf{Bs} \in \mathbb{Z}_q^{3k}$
$u := x'_i + \mathbf{t}^\top \mathbf{x}(\mathsf{hm}) \in \mathbb{Z}_q$
Return $\sigma := ([\mathbf{t}]_1, [u]_1)$

$\underline{\mathcal{O}_{\mathrm{VER}}(i^*, \mathsf{hm}^*, ([\mathbf{t}^*]_1, [u^*]_1)):}$ //at most once
If $(i^*, \mathsf{hm}^*) \in \mathcal{Q} \vee (i^* \in \mathcal{L})$:
  Return 0
If $[u^*]_1 := [x'_{i^*}]_1 + [\mathbf{t}^{*\top}]_1 \cdot \mathbf{x}(\mathsf{hm}^*)$:
  $\beta := 1$
  Return 1
Else: Return 0

$\underline{\mathcal{O}'_{\mathrm{CORR}}(i)}$
$\mathcal{L} := \mathcal{L} \cup \{i\}$
Return $[x'_i]_1$

Fig. 8. Game $\mathsf{UF\text{-}CMA}^{\mathsf{corr}}$ for Lemma 2.

perfectly simulated using $\mathcal{O}_{\mathrm{MAC}}$ and $\mathcal{O}'_{\mathrm{CORR}}$, respectively. The additional group elements $[\mathbf{v}]_1$ from σ and $[\mathbf{y}'_i]_1$ can be simulated as in G_2. Finally, using a valid forgery $(i^*, \mathsf{m}^*, \sigma^*)$ output by $\mathcal{A}$, $\mathcal{B}_{\mathsf{MAC}}$ wins its own game by calling $\mathcal{O}_{\mathrm{VER}}(i^*, \mathsf{hm}^*, ([\mathbf{t}^*]_1, [u^*]_1)$, where $([\mathbf{t}^*]_1, [u^*]_1)$ is a valid MAC tag on hm^* for user i^*. □

$\underline{\mathcal{B}_{\mathsf{MAC}}^{\mathcal{O}_{\mathrm{MAC}}(\cdot), \mathcal{O}_{\mathrm{VER}}(\cdot), \mathcal{O}'_{\mathrm{CORR}}(\cdot)}(\mathsf{pp}_{\mathsf{MAC}}):}$
Parse $\mathsf{pp}_{\mathsf{MAC}} =: (\mathcal{PG}, [\mathbf{B}]_1, ([\mathbf{d}_{i,j}]_1)_{1 \leq i \lambda, j=0,1})$
$\mathbf{A} \leftarrow_{\$} \mathcal{D}_k$
For $1 \leq i \leq \lambda$ and $j = 0, 1$:
  $\mathbf{Z}_{i,j} \leftarrow_{\$} \mathbb{Z}_q^{3k \times k}$
  $\mathbf{E}_{i,j} := (\mathbf{B}^\top \mathbf{Z}_{i,j} - \mathbf{d}_{i,j} \cdot \underline{\mathbf{A}})\overline{\mathbf{A}}^{-1} \in \mathbb{Z}_q^{k \times k}$
  $\mathbf{P}_{i,j} := (\mathbf{E}_{i,j} \parallel \mathbf{d}_{i,j})$
$\mathsf{pp} := (\mathcal{PG}, [\mathbf{A}]_2, [\mathbf{B}]_1, ([\mathbf{Z}_{i,j}]_2, [\mathbf{P}_{i,j}]_1)_{1 \leq i \leq \lambda, j=0,1})$
For $1 \leq i \leq \mu$:
  $\mathbf{z}'_i \leftarrow_{\$} \mathbb{Z}_q^{1 \times k}$
  $vk_i := [\mathbf{z}'_i]_2$ //ssk_i is undefined
$(i^*, \mathsf{m}^*, \sigma^*) \leftarrow_{\$} \mathcal{A}^{\mathcal{O}_{\mathrm{SIGN}}(\cdot,\cdot), \mathcal{O}_{\mathrm{CORR}}(\cdot)}(\mathsf{pp}, \{vk_i\}_{1 \leq i \leq \mu})$
If $(i^* \in \mathcal{S}^{\mathsf{corr}}) \vee (m^* \in \mathcal{M}_{i^*}) \vee (\mathsf{Ver}(vk_{i^*}, m^*, \sigma^*) = 0)$:
  Return 0
$\mathsf{hm}^* := H(vk_{i^*}, \mathsf{m}^*)$
If $\exists 1 \leq i \leq \mu \wedge \mathsf{m} \in \mathcal{M}_i : H(vk_i, \mathsf{m}) = \mathsf{hm}^*$
  Return 0
Parse $\sigma^* := ([\mathbf{t}^*]_1, [u^*]_1, [\mathbf{v}^*]_1)$
$\mathcal{O}_{\mathrm{VER}}(i^*, \mathsf{hm}^*, [\mathbf{t}^*]_1, [u^*]_1)$
Return 1

$\underline{\mathcal{O}_{\mathrm{SIGN}}(i, \mathsf{m}):}$
$\mathsf{hm} := H(vk_i, \mathsf{m})$
$([\mathbf{t}]_1, [u]_1) \leftarrow_{\$} \mathcal{O}_{\mathrm{MAC}}(\mathsf{hm})$
$\mathbf{v} := (\mathbf{z}'_i + \mathbf{t}^\top \mathbf{Z}(\mathsf{hm}) - u \cdot \underline{\mathbf{A}}) \cdot (\overline{\mathbf{A}})^{-1}$
$\mathcal{M}_i := \mathcal{M}_i \cup \{m\}$
Return $\sigma := ([\mathbf{t}]_1, [u]_1, [\mathbf{v}]_1)$

$\underline{\mathcal{O}_{\mathrm{CORR}}(i):}$
$\mathcal{S}^{\mathsf{corr}} := \mathcal{S}^{\mathsf{corr}} \cup \{i\}$
$[x'_i]_1 \leftarrow \mathcal{O}'_{\mathrm{CORR}}(i)$
$\mathbf{y}'_i = (\mathbf{z}'_i - x'_i \cdot \underline{\mathbf{A}})(\overline{\mathbf{A}})^{-1}$
Return $ssk_i := ([x'_i]_1, [\mathbf{y}'_i]_1)$

Fig. 9. Reduction $\mathcal{B}_{\mathsf{MAC}}$ to bound the winning probability in G_2. $\mathcal{B}_{\mathsf{MAC}}$ receives $\mathsf{pp}_{\mathsf{MAC}}$ and gets oracle access to $\mathcal{O}_{\mathrm{MAC}}$ and $\mathcal{O}_{\mathrm{VER}}$, and $\mathcal{O}'_{\mathrm{CORR}}$ as in Figure 8.

6 Concrete Instantiation of Our AKE Protocols

For $\mathsf{AKE}_{\mathsf{3msg}}$, we use our new signature scheme $\mathsf{SIG}_{\mathsf{MDDH}}$ (Figure 6) and the ϵ-MU-SIM KEM constructed from the MDDH-based hash proof system $\mathsf{HPS}_{\mathsf{MDDH}}$ (cf. the full version [21]). For $\mathsf{AKE}^{\mathsf{state}}_{\mathsf{3msg}}$, the symmetric encryption scheme to protect against state reveals can be instantiated using any weakly secure (deterministic) encryption scheme such as AES or even a weak PRF.

For the KEM constructed in the full version [21], the KEM public key consists of $2k$ group elements and the ciphertext of $k+1$ group elements. A signature consists of $4k+1$ group elements, cf. Figure 6. Therefore, the first message is a bitstring of length λ, the second message consists of $6k+1$ group elements and the third message consists of $5k+2$ group elements. For $k=1$, we get an efficient SXDH-based scheme with 15 elements in total.

We instantiate protocol $\mathsf{AKE}_{\mathsf{2msg}}$ using our signature scheme from Figure 6 and the MUC-otCCA secure KEM from Han *et al.* [22]. γ-diversity of the KEM is proven in [29, Appendix D.2]. We analyze the communication complexity of $\mathsf{AKE}_{\mathsf{2msg}}$ as follows. The KEM public key consists of k^2+3k group elements and the ciphertext of $2k+3$ group elements. A signature consists of $4k+1$ group elements. Therefore, the first message consists of k^2+7k+1 group elements and the second message consists of $6k+4$ group elements. For $k=1$, we get an efficient SXDH-based scheme with $9+10=19$ group elements in total.

For an overview we refer to Table 1 of the introduction.

Acknowledgments. We would like to thank the reviewers for their helpful comments. Shuai Han and Shengli Liu were partially supported by National Natural Science Foundation of China (Grant Nos. 61925207, 62002223), Guangdong Major Project of Basic and Applied Basic Research (2019B030302008), Shanghai Sailing Program (20YF1421100), Young Elite Scientists Sponsorship Program by China Association for Science and Technology, and the National Key Research and Development Project 2020YFA0712300. Tibor Jager was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement 802823. Eike Kiltz was supported by the BMBF iBlockchain project, the EU H2020 PROMETHEUS project 780701, DFG SPP 1736 Big Data, and the DFG Cluster of Excellence 2092 CASA. Doreen Riepel was supported by the Deutsche Forschungsgemeinschaft (DFG) Cluster of Excellence 2092 CASA. Sven Schäge was supported by the German Federal Ministry of Education and Research (BMBF), Project DigiSeal (16KIS0695) and Huawei Technologies Düsseldorf, Project vHSM.

References

[1] Bader, C.: Efficient signatures with tight real world security in the random-oracle model. In: Gritzalis, D., Kiayias, A., Askoxylakis, I.G. (eds.) CANS 14. LNCS, vol. 8813, pp. 370–383. Springer, Heidelberg (Oct 2014)

[2] Bader, C., Hofheinz, D., Jager, T., Kiltz, E., Li, Y.: Tightly-secure authenticated key exchange. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 629–658. Springer, Heidelberg (Mar 2015)

[3] Bader, C., Jager, T., Li, Y., Schäge, S.: On the impossibility of tight cryptographic reductions. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 273–304. Springer, Heidelberg (May 2016)

[4] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993)

[5] Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (Aug 1994)

[6] Blazy, O., Kiltz, E., Pan, J.: (Hierarchical) identity-based encryption from affine message authentication. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 408–425. Springer, Heidelberg (Aug 2014)

[7] Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (May 2001)

[8] Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., shelat, a., Vaikuntanathan, V.: Bounded CCA2-secure encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 502–518. Springer, Heidelberg (Dec 2007)

[9] Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (Apr / May 2002)

[10] Cremers, C.J.F., Feltz, M.: Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 734–751. Springer, Heidelberg (Sep 2012)

[11] Davis, H., Günther, F.: Tighter proofs for the SIGMA and TLS 1.3 key exchange protocols. ACNS 2021 (2021), https://eprint.iacr.org/2020/1029

[12] Diemert, D., Gellert, K., Jager, T., Lyu, L.: More efficient digital signatures with tight multi-user security. 24th International Conference on Practice and Theory of Public-Key Cryptography, PKC 2021 (2021)

[13] Diemert, D., Jager, T.: On the tight security of TLS 1.3: Theoretically-sound cryptographic parameters for real-world deployments. Cryptology ePrint Archive, Report 2020/726 (2020), https://eprint.iacr.org/2020/726

[14] Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (Aug 2013)

[15] Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.L.: An algebraic framework for Diffie-Hellman assumptions. Journal of Cryptology 30(1), 242–288 (Jan 2017)

[16] Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 467–484. Springer, Heidelberg (May 2012)

[17] Gay, R., Hofheinz, D., Kiltz, E., Wee, H.: Tightly CCA-secure encryption without pairings. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 1–27. Springer, Heidelberg (May 2016)

[18] Gay, R., Hofheinz, D., Kohl, L.: Kurosawa-desmedt meets tight security. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 133–160. Springer, Heidelberg (Aug 2017)

[19] Gjøsteen, K., Jager, T.: Practical and tightly-secure digital signatures and authenticated key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 95–125. Springer, Heidelberg (Aug 2018)

[20] Günther, C.G.: An identity-based key-exchange protocol. In: Quisquater, J.J., Vandewalle, J. (eds.) EUROCRYPT'89. LNCS, vol. 434, pp. 29–37. Springer, Heidelberg (Apr 1990)

[21] Han, S., Jager, T., Kiltz, E., Liu, S., Pan, J., Riepel, D., Schäge, S.: Authenticated key exchange and signatures with tight security in the standard model. Cryptology ePrint Archive, Report 2021/863 (2021), https://eprint.iacr.org/2021/863

[22] Han, S., Liu, S., Lyu, L., Gu, D.: Tight leakage-resilient CCA-security from quasi-adaptive hash proof system. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 417–447. Springer, Heidelberg (Aug 2019)

[23] Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. In: Canteaut, A., Standaert, F. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 117–146. Springer (2021)

[24] Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 273–293. Springer, Heidelberg (Aug 2012)

[25] Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (Aug 2005)

[26] Langrehr, R., Pan, J.: Tightly secure hierarchical identity-based encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part I. LNCS, vol. 11442, pp. 436–465. Springer, Heidelberg (Apr 2019)

[27] Langrehr, R., Pan, J.: Unbounded HIBE with tight security. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 129–159. Springer, Heidelberg (Dec 2020)

[28] Li, Y., Schäge, S.: No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1343–1360. ACM Press (Oct / Nov 2017)

[29] Liu, X., Liu, S., Gu, D., Weng, J.: Two-pass authenticated key exchange with explicit authentication and tight security. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 785–814. Springer, Heidelberg (Dec 2020)

[30] Morgan, A., Pass, R., Shi, E.: On the adaptive security of MACs and PRFs. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 724–753. Springer, Heidelberg (Dec 2020)

[31] Morillo, P., Ràfols, C., Villar, J.L.: The kernel matrix Diffie-Hellman assumption. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 729–758. Springer, Heidelberg (Dec 2016)

[32] Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (Aug 2002)