

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

Shuai Han<sup>1,2</sup> , Tibor Jager<sup>3</sup> , Eike Kiltz<sup>4</sup> , Shengli Liu<sup>1,2,5</sup>  , Jiaxin Pan<sup>6</sup> , Doreen Riepel<sup>4</sup> , and Sven Schäge<sup>4</sup> 

<sup>1</sup> School of Electronic Information and Electrical Engineering,  
Shanghai Jiao Tong University, Shanghai 200240, China  
{dalen17, slliu}@sjtu.edu.cn

<sup>2</sup> State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

<sup>3</sup> Bergische Universität Wuppertal, Germany  
tiber.jager@uni-wuppertal.de

<sup>4</sup> Ruhr-Universität Bochum, Germany  
{eike.kiltz, doreen.riepel, sven.schaege}@rub.de

<sup>5</sup> Westone Cryptologic Research Center, Beijing 100070, China

<sup>6</sup> 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, 22, 31, 25, 15, 13]. As recently pointed out by Jager et al. [25], some of these constructions [2, 22, 31] 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 [25, 15, 13] apply proof techniques that seem to inherently require the random oracle model [4]. For instance, [25] is based on non-committing encryption, which is known to be not instantiable without random oracles [35], whereas [15, 13] 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, 22, 15, 13, 31, 25] 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<sup>corr</sup> security [2]). It is easy to prove that MU-EUF-CMA<sup>corr</sup> 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<sup>corr</sup> 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].

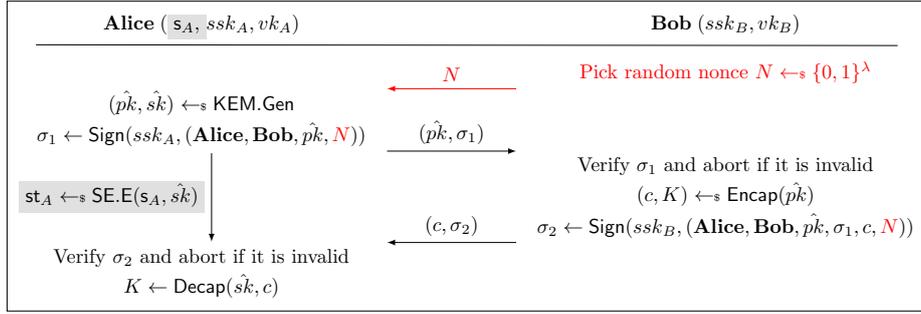
Several previous works have developed techniques to overcome this seeming paradox [1, 2, 22, 14]. Essentially, their approach is to build schemes where secret keys are not uniquely determined by public parameters, along with a reduction that exploits this to evade the paradox. However, all currently 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<sup>corr</sup> security proof of the scheme from [2] with constant-size signatures (namely, SIG<sub>C</sub> 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<sup>corr</sup> security in the standard model. More precisely, SIG<sub>C</sub> 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



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

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 [28, 29] to construct a new MAC 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 [18]. 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<sup>corr</sup>-secure signature in the standard model with practical efficiency.

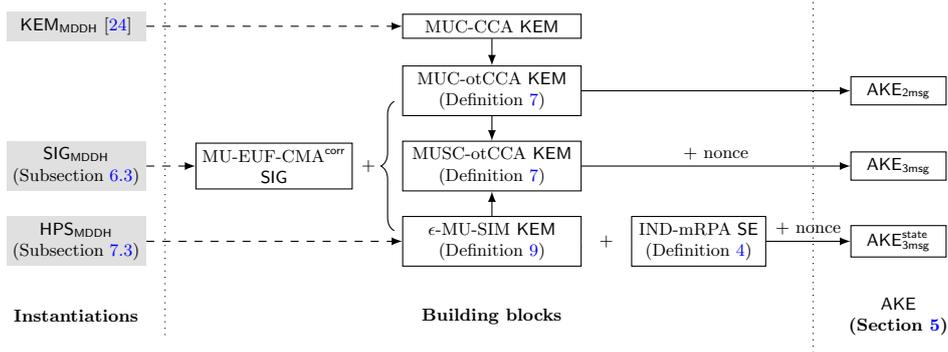
We remark that our new signature scheme circumvents known impossibility results for signatures and MACs [3, 32], since these apply only to schemes with re-randomizable signatures or re-randomizable secret keys [3], or deterministic schemes [32]. Our construction is probabilistic and not efficiently re-randomizable in the sense of [3].<sup>7</sup>

**TIGHTLY-SECURE AKE IN THE STANDARD MODEL.** The classical “*key encapsulation plus digital signatures*” (KEM + 2 × SIG) paradigm to construct AKE protocols gives rise to efficient protocols and is the basis of many constructions, e.g., [7, 11, 22, 15, 13, 31, 25]. To establish a session key, two parties Alice and Bob proceed as follows (cf. Figure 1). Alice generates an ephemeral KEM key pair  $(\hat{p}k, \hat{s}k)$  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 KEM + 2 × SIG paradigm. Given a tightly MU-EUF-CMA<sup>corr</sup> secure signature scheme, it remains to determine suitable security notions for the underlying KEM, which finds a balance between two properties. The security 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* [30]. The model is defined in pseudocode to avoid ambiguity.

- Our first result is a new tight security proof for the two-message protocol  $\text{AKE}_{2\text{msg}}$ , which follows the KEM+2 × SIG paradigm.  $\text{AKE}_{2\text{msg}}$  is exactly the LLGW protocol [31] 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

<sup>7</sup> 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.

version of the standard Multi-User/Challenge CCA (MUC-CCA) security notion for KEMs (allowing for unbounded decapsulation queries [20]), the most efficient instantiations we could find are the MUC-CCA-secure schemes with tight security from [20, 21, 24].<sup>8</sup>

- Our second result is a three-message protocol  $\text{AKE}_{3\text{msg}}$  resisting replay attacks, which extends the  $\text{KEM} + 2 \times \text{SIG}$  protocol  $\text{AKE}_{2\text{msg}}$  with an additional nonce sent at the beginning of the protocol (“Nonce+KEM+ $2 \times \text{SIG}$ ”). For our security proof we require the KEM security notion of Multi-User Single-Challenge one-time CCA (MUSC-otCCA) security, allowing the adversary 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<sub>2</sub> hash proof system [10]. (For example, based on a standard assumption such as Matrix DDH (MDDH) [18] which yields highly efficient KEMs.)
- Our third result is a three-message protocol  $\text{AKE}_{3\text{msg}}^{\text{state}}$ , which extends the Nonce+KEM+ $2 \times \text{SIG}$  protocol  $\text{AKE}_{3\text{msg}}$  by encrypting the state with a symmetric encryption (SE) scheme.  $\text{AKE}_{3\text{msg}}^{\text{state}}$  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” ( $\epsilon$ -MU-SIM) security notion for KEMs, which we also show to be tightly achievable by universal<sub>2</sub> 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.<sup>9</sup>

Note that our  $\text{AKE}_{3\text{msg}}$  and  $\text{AKE}_{3\text{msg}}^{\text{state}}$  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 [31]. Consequently, we can also use counter instead of nonce in  $\text{AKE}_{3\text{msg}}$  and  $\text{AKE}_{3\text{msg}}^{\text{state}}$ , and obtain two two-message counter-based AKE protocols which have the same efficiency and security as  $\text{AKE}_{3\text{msg}}$  and  $\text{AKE}_{3\text{msg}}^{\text{state}}$ , respectively.

**INSTANTIATIONS.** Table 1 gives example instantiations of our protocols from universal<sub>2</sub> hash proof systems from the MDDH assumption and compares them to known protocols. The protocols BHJKL [2] and LLGW [31] 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 [25]. For more details on our instantiations we refer to Section 7. Note that there are other works which study AKE in the standard model (e.g., [19, 26]). 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.

<sup>8</sup> We are aware of the generic constructions of bounded-CCA secure KEMs from CPA-secure KEMs [9], but they do not seem to offer tight security in a multi-challenge setting.

<sup>9</sup> 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  $\text{AKE}_{3\text{msg}}^{\text{state}}$  and  $\text{AKE}_{2\text{msg}}$  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 6.3. 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 [31]	$9 + 10$ $(k^2 + 7k + 1) + (6k + 4)$	2	SXDH $\mathcal{D}_k$ -MDDH	no	$O(\lambda T)$
$\text{AKE}_{3\text{msg}}^{\text{state}}$	$8 + 7$ $(5k + 3) + (5k + 2)$	3	SXDH $\mathcal{D}_k$ -MDDH	yes	$O(\lambda)$
$\text{AKE}_{2\text{msg}}$ (= LLGW)	$9 + 10$ $(k^2 + 7k + 1) + (6k + 4)$	2	SXDH $\mathcal{D}_k$ -MDDH	no	$O(\lambda)$

To obtain an AKE protocol with a tight security reduction in the  $\text{KEM} + 2 \times \text{SIG}$  framework, we rely on the tight  $\text{MU-EUF-CMA}^{\text{corr}}$  security of the signature scheme to guarantee authentication and deal with corruptions, and on the tight  $\text{MUC-CCA}$  security of KEM to deal with session key reveals. To this end, recall that the SBG-style security game for  $\text{MUC-CCA}$  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  $\text{AKE}_{2\text{msg}}$ . We remark that  $\text{AKE}_{2\text{msg}}$  can also be proved secure under an even weaker security notion for KEM, namely  $\text{MUC-otCCA}$ , 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  $pk$ .

To prevent replay attacks we make use of an exchange of nonces resulting in protocol  $\text{AKE}_{3\text{msg}}$ . 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  $pk$ , and thus we can ensure that the simulator only needs to respond to one encapsulation query per  $pk$  in the security game. In this way we can further weaken the security requirement that we need from the KEM to  $\text{MUC-otCCA}$ .

Now, to support state reveals, we use a symmetric encryption scheme  $\text{SE}$  that is used to encrypt the ephemeral secret key  $\hat{s}k$  of each session, similar to [25]. More concretely, we require that the state is computed as  $\text{st} = \text{SE.E}(s, \hat{s}k)$ , where  $s$  is the secret key of  $\text{SE}$  that is made part of the long-term secret key. This modification yields protocol  $\text{AKE}_{3\text{msg}}^{\text{state}}$ . 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  $\text{st}$ . But now the reduction to the  $\text{MUC-otCCA}$  security of the KEM cannot work as before, since the reduction algorithm cannot output  $\text{SE.E}(s, \hat{s}k)$  to the adversary. A natural way to address this problem is to make use of the security of  $\text{SE}$ , and make the reduction change the state to an encryption of some dummy random key  $r$ , i.e.,  $\text{st} = \text{SE.E}(s, r)$ . However, now the  $\text{SE}$  reduction algorithm is faced with a critical decision: If the adversary asks a state reveal query, should the reduction output  $\text{st} = \text{SE.E}(s, \hat{s}k)$  or  $\text{st} = \text{SE.E}(s, r)$ ? It seems that both choices are problematic. If the reduction responds with the encryption of KEM secret key  $\hat{s}k$ , 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 [25].

To obtain a tight security supporting state reveals in the standard model, we enhance the  $\text{MUC-otCCA}$  security of KEM to our new  $\epsilon$ - $\text{MU-SIM}$ -security, so that a symmetric encryption scheme  $\text{SE}$  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  $\epsilon$ -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  $\epsilon$ -MU-SIM-secure KEM in slightly more detail. In a nutshell, an  $\epsilon$ -MU-SIM-secure KEM provides the reduction with access to an additional encapsulation algorithm  $\text{Encap}^*$  that is keyed with the secret key. We have security requirements as follows:

- Computational indistinguishability between  $\text{Encap}$  and  $\text{Encap}^*$ : We require that the reduction can switch to using  $\text{Encap}^*$  without the adversary noticing even given the secret key  $\hat{s}k$  of the KEM. In particular, the resulting indistinguishability notion must tightly reduce to an underlying security assumption.
- Statistical  $\epsilon$ -uniformity: When using the alternative encapsulation mechanism  $\text{Encap}^*$ , we require that the encapsulated key in the challenge ciphertext  $c^*$  will be indistinguishable from random with *statistical distance*  $\epsilon$  (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  $\epsilon$ -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<sub>2</sub> hash proof systems (HPS). In particular, we show that the  $\epsilon$ -MU-SIM-security is implied by the hardness of subset membership problems and the universal<sub>2</sub>-property of HPS.

Our new  $\epsilon$ -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  $\text{Encap}$  to  $\text{Encap}^*$ . By the security properties of our KEM, the adversary cannot notice this, even given  $\hat{s}k$ .
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  $\eta$ -th hybrid ( $\eta = 1, \dots, \mu$  with  $\mu$  being the number of users), we replace the test session keys related to the  $\eta$ -th user with random keys. We can show that this is not recognizable by the adversary since the key  $K^*$  generated by  $\text{Encap}^*$  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  $\eta$ -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  $\text{SE.E}(s, \hat{s}k)$  and thus we can replace the session key by exploiting the  $\epsilon$ -uniformity of  $\text{Encap}^*$ .

**Case 2:** The adversary does not corrupt the  $\eta$ -th user. In this case, we rely on the IND-mRPA security of SE and replace  $\hat{s}k$  in the encrypted state with a random dummy key for this user. Then, we can use  $\epsilon$ -uniformity to replace all tested keys for that user with random keys, as the state does not contain any information about  $\hat{s}k$ . After that, we have to switch back to using the original state encryption mechanism and encrypt the real secret key  $\hat{s}k$ , getting ready for the next hybrid.

After  $\mu$  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  $2\mu$  – 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 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  $[\mu] := \{1, 2, \dots, \mu\}$ . Denote by  $x \leftarrow_s \mathcal{X}$  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  $y \leftarrow_s \mathcal{A}(x)$  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  $\epsilon$ -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  $\mathcal{A}$ ,

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

## 2.1 Digital Signature

**Definition 2 (SIG).** A signature (SIG) scheme  $\text{SIG} = (\text{SIG.Setup}, \text{SIG.Gen}, \text{Sign}, \text{Ver})$  is defined by the following four algorithms.

- **SIG.Setup:** The setup algorithm outputs a public parameter  $\text{pp}_{\text{SIG}}$ , which defines a message space  $\mathcal{M}$ , a signature space  $\Sigma$ , and verification key & signing key spaces  $\mathcal{VK} \times \mathcal{SK}$ .
- **SIG.Gen(pp<sub>SIG</sub>):** The key generation algorithm takes as input  $\text{pp}_{\text{SIG}}$  and outputs a pair of keys  $(vk, ssk) \in \mathcal{VK} \times \mathcal{SK}$ .
- **Sign(ssk, m):** Taking as input a signing key  $ssk$  and a message  $m \in \mathcal{M}$ , the signing algorithm outputs a signature  $\sigma \in \Sigma$ .
- **Ver(vk, m,  $\sigma$ ):** Taking as input a verification key  $vk$ , a message  $m$  and a signature  $\sigma$ , the deterministic verification algorithm outputs a bit indicating whether  $\sigma$  is a valid signature for  $m$  w.r.t.  $vk$ .

We require that for all  $\text{pp}_{\text{SIG}} \in \text{SIG.Setup}$ ,  $(vk, ssk) \in \text{SIG.Gen}(\text{pp}_{\text{SIG}})$ , we have  $\text{Ver}(vk, m, \text{Sign}(ssk, m)) = 1$ .

Below we present the security notion of existential unforgeability with adaptive corruptions in the multi-user setting (MU-EUF-CMA<sup>corr</sup>) for SIG, which was originally defined in [2].

**Definition 3 (MU-EUF-CMA<sup>corr</sup> Security for SIG).** To a signature scheme SIG, the number of users  $\mu \in \mathbb{N}$ , and an adversary  $\mathcal{A}$  we associate the advantage function  $\text{Adv}_{\text{SIG}, \mu}^{\text{mu-corr}}(\mathcal{A}) := \Pr[\text{Exp}_{\text{SIG}, \mu, \mathcal{A}}^{\text{mu-corr}} \Rightarrow 1]$ , where  $\text{Exp}_{\text{SIG}, \mu, \mathcal{A}}^{\text{mu-corr}}$  is defined in Figure 3.

$\text{Exp}_{\text{SIG}, \mu, \mathcal{A}}^{\text{mu-corr}}$ $\text{pp}_{\text{SIG}} \leftarrow_{\mathcal{S}} \text{SIG.Setup}$ For $i \in [\mu]$ : $(vk_i, ssk_i) \leftarrow_{\mathcal{S}} \text{SIG.Gen}(\text{pp}_{\text{SIG}})$ ; $\mathcal{M}_i := \emptyset$ //Record messages from signing queries $\mathcal{S}^{\text{corr}} := \emptyset$ //Record corruption queries $(i^*, m^*, \sigma^*) \leftarrow_{\mathcal{S}} \mathcal{A}^{\mathcal{O}_{\text{SIGN}}(\cdot, \cdot), \mathcal{O}_{\text{CORR}}(\cdot)}(\text{pp}_{\text{SIG}}, \text{VKList} := \{vk_i\}_{i \in [\mu]})$ If $(i^* \notin \mathcal{S}^{\text{corr}}) \wedge (m^* \notin \mathcal{M}_{i^*}) \wedge (\text{Ver}(vk_{i^*}, m^*, \sigma^*) = 1)$ : Return 1 Else: Return 0	$\mathcal{O}_{\text{SIGN}}(i, m)$ : $\sigma \leftarrow_{\mathcal{S}} \text{Sign}(ssk_i, m)$ $\mathcal{M}_i := \mathcal{M}_i \cup \{m\}$ Return $\sigma$  $\mathcal{O}_{\text{CORR}}(i)$ : $\mathcal{S}^{\text{corr}} := \mathcal{S}^{\text{corr}} \cup \{i\}$ Return $ssk_i$
---	---

**Fig. 3.** The MU-EUF-CMA<sup>corr</sup> security experiment  $\text{Exp}_{\text{SIG}, \mu, \mathcal{A}}^{\text{mu-corr}}$  for SIG.

## 2.2 Symmetric Encryption

**Definition 4 (SE).** A symmetric encryption (SE) scheme  $\text{SE} = (\text{E}, \text{D})$  is associated with a key space  $\mathcal{K}$ , a plaintext space  $\mathcal{M}$  and a ciphertext space  $\mathcal{C}$ . It is defined by the following two algorithms.

- **E(k, m):** Taking as input a symmetric key  $k \in \mathcal{K}$  and a plaintext  $m \in \mathcal{M}$ , the encryption algorithm outputs a ciphertext  $c \in \mathcal{C}$ .
- **D(k, c):** Taking as input a symmetric key  $k \in \mathcal{K}$  and a ciphertext  $c \in \mathcal{C}$ , the decryption algorithm outputs a plaintext  $m \in \mathcal{M}$ .

We require that for all  $k \in \mathcal{K}$ , all  $m \in \mathcal{M}$ , we have  $D(k, E(k, m)) = m$ .

Below we define the indistinguishability against multi-challenge Random Plaintext Attacks (IND-mRPA security) for SE, which asks indistinguishability of encryptions of random plaintexts and encryptions of dummy (random) plaintexts.

**Definition 5 (IND-mRPA Security for SE).** To a symmetric encryption scheme SE, the number of users  $\mu \in \mathbb{N}$ , and an adversary  $\mathcal{A}$  we associate the advantage function

$$\text{Adv}_{\text{SE}, \mu}^{\text{mrpa}}(\mathcal{A}) = \left| \Pr \left[ \mathcal{A}(\{m_i, c_i^{(0)}\}_{i \in [\mu]}) \Rightarrow 1 \right] - \Pr \left[ \mathcal{A}(\{m_i, c_i^{(1)}\}_{i \in [\mu]}) \Rightarrow 1 \right] \right|,$$

where  $k \leftarrow_s \mathcal{K}$ ,  $m_i, r_i \leftarrow_s \mathcal{M}$ ,  $c_i^{(0)} \leftarrow_s E(k, m_i)$  and  $c_i^{(1)} \leftarrow_s E(k, r_i)$  for  $\forall i \in [\mu]$ .

*Remark 1.* We note that IND-mRPA is weaker than the traditional IND-CPA (indistinguishability against Chosen Plaintext attacks) security notion. In particular, IND-mRPA is achievable by deterministic SEs, while IND-CPA necessarily requires a probabilistic encryption. Consequently, IND-mRPA secure SE admits more practical instantiations. For example, a PRF or even a weak PRF [34] itself is an IND-mRPA secure SE.

### 3 Security Notions for KEMs

In the section, we present definitions of Key Encapsulation Mechanism (KEM) and its security notions.

**Definition 6 (KEM).** A key encapsulation mechanism (KEM) scheme  $\text{KEM} = (\text{KEM.Setup}, \text{KEM.Gen}, \text{Encap}, \text{Decap})$  consists of four algorithms:

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

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 7 (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  $\text{Adv}_{\text{KEM}, \mu}^{\text{muc-otcca}}(\mathcal{A}) := \left| \Pr[\text{Exp}_{\text{KEM}, \mu, \mathcal{A}}^{\text{muc-otcca}} \Rightarrow 1] - \frac{1}{2} \right|$  and  $\text{Adv}_{\text{KEM}, \mu}^{\text{musc-otcca}}(\mathcal{A}) := \left| \Pr[\text{Exp}_{\text{KEM}, \mu, \mathcal{A}}^{\text{musc-otcca}} \Rightarrow 1] - \frac{1}{2} \right|$ , where the experiments are defined in Figure 4.

Below we recall the definition of the *diversity property* from [31].

**Definition 8 ( $\gamma$ -Diversity of KEM).** A KEM scheme  $\text{KEM}$  is called  $\gamma$ -diverse if for all  $\text{pp}_{\text{KEM}} \in \text{KEM.Setup}$ , it holds that

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

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

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

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

**Fig. 4.** The MUC-otCCA security experiment  $\text{Exp}_{\text{KEM}, \mu, \mathcal{A}}^{\text{muc-otcca}}$  and the MUSC-otCCA security experiment  $\text{Exp}_{\text{KEM}, \mu, \mathcal{A}}^{\text{musc-otcca}}$  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  $\epsilon$ -MU-SIM ( $\epsilon$ -multi-user simulatable) security. Jumping ahead,  $\epsilon$ -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  $\epsilon$ -MU-SIM security (Definition 9) and in Subsection 7.2, we present simple constructions of  $\epsilon$ -MU-SIM secure KEMs from universal<sub>2</sub>-HPS.

Informally,  $\epsilon$ -MU-SIM security requires that there exists a simulated encapsulation algorithm  $\text{Encap}^*(sk)$  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  $\epsilon$ -close to uniform.

**Definition 9 ( $\epsilon$ -MU-SIM Security for KEM).** *We require that there exists a simulated encapsulation algorithm  $\text{Encap}^*(sk)$  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  $\epsilon$ -uniformity we require that for any (unbounded) adversary  $\mathcal{A}$ , it holds that*

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

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

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

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

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

Note that  $\epsilon$ -MU-SIM security tightly implies MUSC-otCCA<sup>rev&corr</sup> 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  $sk_1, \dots, sk_\mu$ . By (1) one can see that one single decapsulation query is supported. In particular,  $\epsilon$ -MU-SIM security tightly implies MUSC-otCCA security. In Section 7 we will define universal<sub>2</sub> hash proof systems and show how they imply  $\epsilon$ -MU-SIM secure KEMs.

## 4 Authenticated Key Exchange

### 4.1 Definition of Authenticated Key Exchange

**Definition 10 (AKE).** *An authenticated key exchange (AKE) scheme  $\text{AKE} = (\text{AKE.Setup}, \text{AKE.Gen}, \text{AKE.Protocol})$  consists of two probabilistic algorithms and an interactive protocol.*

- **AKE.Setup:** The setup algorithm outputs the public parameter  $\text{pp}_{\text{AKE}}$ .

- $\text{AKE.Gen}(\text{pp}_{\text{AKE}}, P_i)$ : The generation algorithm takes as input  $\text{pp}_{\text{AKE}}$  and a party  $P_i$ , and outputs a key pair  $(pk_i, sk_i)$ .
- $\text{AKE.Protocol}(P_i(\text{res}_i) \rightleftharpoons P_j(\text{res}_j))$ : The protocol involves two parties  $P_i$  and  $P_j$ , who have access to their own resources,  $\text{res}_i := (sk_i, \text{pp}_{\text{AKE}}, \{pk_u\}_{u \in [\mu]})$  and  $\text{res}_j := (sk_j, \text{pp}_{\text{AKE}}, \{pk_u\}_{u \in [\mu]})$ , respectively. Here  $\mu$  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  $(\Psi_j, k_j)$  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  $\text{AKE.Protocol}(P_i(\text{res}_i) \rightleftharpoons P_j(\text{res}_j))$ , i.e.,  $\Psi_i = \Psi_j = \mathbf{accept}$ ,  $k_i = k_j \neq \emptyset$ .

## 4.2 Security Model of AKE

We will adapt the security model formalized by [2, 30, 22], which in turn followed the model proposed by Bellare and Rogaway [5]. We also include replay attacks [31] and multiple test queries with respect to the same random bit [25].

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

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

- $\text{st}_i^s$ : State information that has to be stored between two rounds in order to execute the protocol.
- $\text{Pid}_i^s$ : The intended communication peer's identity.
- $k_i^s \in \mathcal{K}$ : The session key computed by  $\pi_i^s$ . Here  $\mathcal{K}$  is the session key space. We assume that  $\emptyset \in \mathcal{K}$ .
- $\Psi_i^s \in \{\emptyset, \mathbf{accept}, \mathbf{reject}\}$ :  $\Psi_i^s$  indicates whether  $\pi_i^s$  has completed the protocol execution and accepted  $k_i^s$ .

At the beginning,  $(\text{st}_i^s, \text{Pid}_i^s, k_i^s, \Psi_i^s)$  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  $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}$  with the help of the oracles defined above.  $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}$  is a game played between an AKE challenger  $\mathcal{C}$  and an adversary  $\mathcal{A}$ .  $\mathcal{C}$  will simulate the executions of the  $\ell$  protocol instances for each of the  $\mu$  users with oracles  $\pi_i^s$ . We give a formal description in Figure 5.

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

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

A  $\text{Corrupt}$  query allows  $\mathcal{A}$  to corrupt a party  $P_i$  and get its long-term secret key  $sk_i$ . With a  $\text{RegisterCorrupt}$  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  $\text{Test}$  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  $\text{Test}$  query for  $\pi_i^s$ , the oracle will return  $\perp$  if the session key  $k_i^s$  is not generated yet. Otherwise,  $\pi_i^s$  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.

- $\text{Send}(i, s, j, \text{msg})$ : If  $\text{msg} = \top$ , it means that  $\mathcal{A}$  asks oracle  $\pi_i^s$  to send the first protocol message to  $P_j$ . Otherwise,  $\mathcal{A}$  impersonates  $P_j$  to send message  $\text{msg}$  to  $\pi_i^s$ . Then  $\pi_i^s$  executes the AKE protocol with  $\text{msg}$  as  $P_i$  does, computes a message  $\text{msg}'$ , and updates its own variables  $\text{var}_i^s = (\text{st}_i^s, \text{Pid}_i^s, k_i^s, \Psi_i^s)$ . The output message  $\text{msg}'$  is returned to  $\mathcal{A}$ . If  $\text{Send}(i, s, j, \text{msg})$  is the  $\tau$ -th query asked by  $\mathcal{A}$  and  $\pi_i^s$  changes  $\Psi_i^s$  to  $\mathbf{accept}$  after that, then we say that  $\pi_i^s$  is  $\tau$ -accepted.

<p><math>\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}</math>:</p> <p><math>\text{pp}_{\text{AKE}} \leftarrow \text{AKE.Setup}</math>  For <math>i \in [\mu]</math>:  <math>(pk_i, sk_i) \leftarrow \text{AKE.Gen}(\text{pp}_{\text{AKE}}, P_i)</math>;  <math>crp_i := \text{false}</math> //Corruption variable  <math>\text{PKList} := \{pk_i\}_{i \in [\mu]}</math>; <math>b \leftarrow \mathfrak{s} \{0, 1\}</math>  For <math>(i, s) \in [\mu] \times [\ell]</math>:  <math>\text{var}_i^s := (\text{st}_i^s, \text{Pid}_i^s, k_i^s, \Psi_i^s) := (\emptyset, \emptyset, \emptyset, \emptyset)</math>;  <math>\text{Aflag}_i^s := \text{false}</math> //Whether <math>\text{Pid}_i^s</math> is corrupted when <math>\pi_i^s</math> accepts  <math>T_i^s := \text{false}</math>; <math>k\text{Rev}_i^s := \text{false}</math> // Test, Key Reveal variables  <math>st\text{Rev}_i^s := \text{false}</math>; <math>First\text{Acc}_i^s := \emptyset</math>  // State Reveal &amp; First Acceptance variables  <math>b^* \leftarrow \mathcal{A}^{\text{AKE}(\cdot)}(\text{pp}_{\text{AKE}}, \text{PKList})</math></p> <p><math>\text{Win}_{\text{Auth}} := \text{false}</math>  <math>\text{Win}_{\text{Auth}} := \text{true}</math>, If <math>\exists (i, s) \in [\mu] \times [\ell]</math> s.t.  (1) <math>\Psi_i^s = \text{accept}</math> // <math>\pi_i^s</math> is <math>\tau</math>-accepted  (2) <math>\text{Aflag}_i^s = \text{false}</math> // <math>P_j</math> is <math>\hat{\tau}</math>-corrupted with <math>j := \text{Pid}_i^s</math> and <math>\hat{\tau} &gt; \tau</math>  (3) (3.1) <math>\vee</math> (3.2) <math>\vee</math> (3.3). Let <math>j := \text{Pid}_i^s</math>  (3.1) <math>\nexists t \in [\ell]</math> s.t. <math>\text{Partner}(\pi_i^s \leftarrow \pi_j^t)</math>  (3.2) <math>\exists t \in [\ell], (j', t') \in [\mu] \times [\ell]</math> with <math>(j, t) \neq (j', t')</math> s.t.  <math>\text{Partner}(\pi_i^s \leftarrow \pi_j^t) \wedge \text{Partner}(\pi_i^{s'} \leftarrow \pi_{j'}^{t'})</math>  (3.3) <math>\exists t \in [\ell], (i', s') \in [\mu] \times [\ell]</math> with <math>(i, s) \neq (i', s')</math> s.t.  <math>\text{Partner}(\pi_i^s \leftarrow \pi_j^t) \wedge \text{Partner}(\pi_{i'}^{s'} \leftarrow \pi_j^t)</math> //Replay attacks</p> <p><math>\text{Win}_{\text{Ind}} := \text{false}</math>  If <math>b^* = b</math>:  <math>\text{Win}_{\text{Ind}} := \text{true}</math>; Return 1  Else: Return 0</p> <p><math>\text{Partner}(\pi_i^s \leftarrow \pi_j^t)</math>: //Checking whether <math>\text{Partner}(\pi_i^s \leftarrow \pi_j^t)</math>  If <math>\pi_i^s</math> sent the first message and <math>k_i^s = \text{K}(\pi_i^s, \pi_j^t) \neq \emptyset</math>: Return 1  If <math>\pi_i^s</math> received the first message and <math>k_i^s = \text{K}(\pi_j^t, \pi_i^s) \neq \emptyset</math>: Return 1  Return 0</p> <p><math>\pi_i^s(\text{msg}, j)</math>:  // <math>\pi_i^s</math> executes AKE according to the protocol specification  If <math>\text{Pid}_i^s = \emptyset</math>: <math>\text{Pid}_i^s := j</math>  If <math>\text{Pid}_i^s = j</math>:  <math>\pi_i^s</math> receives <math>\text{msg}</math> and uses <math>\text{res}_i, \text{var}_i^s</math> to generate the next message <math>\text{msg}'</math> of AKE, and updates <math>(\text{st}_i^s, \text{Pid}_i^s, k_i^s, \Psi_i^s)</math>;  If <math>\text{msg} = \top</math>: <math>\pi_i^s</math> generates the first message <math>\text{msg}'</math> as initiator;  If <math>\text{msg}</math> is the last message of AKE: <math>\text{msg}' := \emptyset</math>;  Return <math>\text{msg}'</math>  If <math>\text{Pid}_i^s \neq j</math>: Return <math>\perp</math></p> <p><math>\mathcal{O}_{\text{AKE}}(\text{query})</math>:  If <math>\text{query} = \text{RegisterCorrupt}(u, pk_u)</math>:  If <math>u \in [\mu]</math>: Return <math>\perp</math>  <math>\text{PKList} := \text{PKList} \cup \{pk_u\}</math>  <math>crp_u := \text{true}</math>  Return <math>\text{PKList}</math></p>	<p><math>\mathcal{O}_{\text{AKE}}(\text{query})</math>:</p> <p>If <math>\text{query} = \text{Send}(i, s, j, \text{msg})</math>:  If <math>\Psi_i^s = \text{accept}</math>: Return <math>\perp</math>  <math>\text{msg}' \leftarrow \pi_i^s(\text{msg}, j)</math>  If <math>\Psi_i^s = \text{accept}</math>:  If <math>crp_j = \text{true}</math>: <math>\text{Aflag}_i^s := \text{true}</math>;  // Determine whether <math>\pi_i^s</math> accepts before its partner  If <math>crp_j = \text{false} \wedge \exists t \in [\ell]</math> s.t. <math>\text{Partner}(\pi_i^s \leftarrow \pi_j^t)</math>:  If <math>\Psi_j^t \neq \text{accept}</math>:  <math>First\text{Acc}_i^s := \text{true}</math>; <math>First\text{Acc}_j^t := \text{false}</math>  If <math>\Psi_j^t = \text{accept}</math>:  <math>First\text{Acc}_i^s := \text{false}</math>; <math>First\text{Acc}_j^t := \text{true}</math>  Return <math>\text{msg}'</math></p> <p>If <math>\text{query} = \text{Corrupt}(i)</math>:  If <math>i \notin [\mu]</math>: Return <math>\perp</math>  For <math>s \in [\ell]</math>  If <math>First\text{Acc}_i^s = \text{false} \wedge st\text{Rev}_i^s = \text{true}</math>:  If <math>T_i^s = \text{true}</math>: Return <math>\perp</math>; //avoid TA6  If <math>\exists t \in [\ell]</math> s.t. <math>\text{Partner}(\pi_i^s \leftarrow \pi_j^t)</math>:  If <math>T_j^t = \text{true}</math>: Return <math>\perp</math>; //avoid TA7  <math>crp_i := \text{true}</math>  Return <math>sk_i</math></p> <p>If <math>\text{query} = \text{SessionKeyReveal}(i, s)</math>:  If <math>\Psi_i^s \neq \text{accept}</math>: Return <math>\perp</math>  If <math>T_i^s = \text{true}</math>: Return <math>\perp</math> //avoid TA2  Let <math>j := \text{Pid}_i^s</math>  If <math>\exists t \in [\ell]</math> s.t. <math>\text{Partner}(\pi_i^s \leftrightarrow \pi_j^t)</math>:  If <math>T_j^t = \text{true}</math>: Return <math>\perp</math> //avoid TA4  <math>k\text{Rev}_i^s := \text{true}</math>; Return <math>k_i^s</math></p> <p>If <math>\text{query} = \text{StateReveal}(i, s)</math>:  If <math>First\text{Acc}_i^s = \text{false} \wedge crp_i = \text{true}</math>:  If <math>T_i^s = \text{true}</math>: Return <math>\perp</math>; //avoid TA6  Let <math>j := \text{Pid}_i^s</math>  If <math>\exists t \in [\ell]</math> s.t. <math>\text{Partner}(\pi_i^s \leftrightarrow \pi_j^t)</math>:  If <math>T_j^t = \text{true}</math>: Return <math>\perp</math>; //avoid TA7  <math>st\text{Rev}_i^s := \text{true}</math>; Return <math>st_i^s</math></p> <p>If <math>\text{query} = \text{Test}(i, s)</math>:  If <math>\Psi_i^s \neq \text{accept} \vee \text{Aflag}_i^s = \text{true} \vee k\text{Rev}_i^s = \text{true}</math>  <math>\vee T_i^s = \text{true}</math>: Return <math>\perp</math> //avoid TA1, TA2, TA3  If <math>First\text{Acc}_i^s = \text{false}</math>:  If <math>crp_i = \text{true} \wedge st\text{Rev}_i^s = \text{true}</math>:  Return <math>\perp</math> //avoid TA6  Let <math>j := \text{Pid}_i^s</math>  If <math>\exists t \in [\ell]</math> s.t. <math>\text{Partner}(\pi_i^s \leftrightarrow \pi_j^t)</math>:  If <math>k\text{Rev}_j^t = \text{true} \vee T_j^t = \text{true}</math>:  Return <math>\perp</math> //avoid TA4, TA5  If <math>\exists t \in [\ell]</math> s.t. <math>\text{Partner}(\pi_i^s \leftarrow \pi_j^t)</math>:  If <math>First\text{Acc}_j^t = \text{false} \wedge crp_j = \text{true}</math>  <math>\wedge st\text{Rev}_j^t = \text{true}</math>: Return <math>\perp</math> //avoid TA7  <math>T_i^s := \text{true}</math>; <math>k_0 := k_i^s</math>; <math>k_1 \leftarrow \mathfrak{s} \mathcal{K}</math>; Return <math>k_b</math></p>
--	---

**Fig. 5.** The security experiments  $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}$ ,  $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}^{\text{replay}}$  (both without red text) and  $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}^{\text{replay, state}}$  (with red text). The list of trivial attacks is given in Table 2.

- **Corrupt**( $i$ ):  $\mathcal{C}$  reveals party  $P_i$ 's long-term secret key  $sk_i$  to  $\mathcal{A}$ . After corruption,  $\pi_i^1, \dots, \pi_i^\ell$  will stop answering any query from  $\mathcal{A}$ .  
If **Corrupt**( $i$ ) is the  $\tau$ -th query asked by  $\mathcal{A}$ , we say that  $P_i$  is  $\tau$ -corrupted.  
If  $\mathcal{A}$  has never asked **Corrupt**( $i$ ), we say that  $P_i$  is  $\infty$ -corrupted.
- **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.
- **StateReveal**( $i, s$ ): The query means that  $\mathcal{A}$  asks  $\mathcal{C}$  to reveal  $\pi_i^s$ 's session state.  $\mathcal{C}$  returns  $st_i^s$  to  $\mathcal{A}$ .
- **SessionKeyReveal**( $i, s$ ): The query means that  $\mathcal{A}$  asks  $\mathcal{C}$  to reveal  $\pi_i^s$ '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  $\pi_i^s$ .
- **Test**( $i, s$ ): If  $\Psi_i^s \neq \mathbf{accept}$ ,  $\mathcal{C}$  returns  $\perp$ . Otherwise,  $\mathcal{C}$  sets  $k_0 = k_i^s$ , samples  $k_1 \leftarrow_s \mathcal{K}$ , and returns  $k_b$  to  $\mathcal{A}$ . We require that  $\mathcal{A}$  can ask **Test**( $i, s$ ) to each oracle  $\pi_i^s$  only once.

Informally, the pseudorandomness of  $k_i^s$  asks that any PPT adversary  $\mathcal{A}$  with access to **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 11 (Original Key [30]).** For two oracles  $\pi_i^s$  and  $\pi_j^t$ , 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  $\pi_i^s$  is the initiator.

*Remark 2.* 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 12 (Partner [30]).** Let  $\mathsf{K}(\cdot, \cdot)$  denote the original key function. We say that an oracle  $\pi_i^s$  is partnered to  $\pi_j^t$ , denoted as  $\mathbf{Partner}(\pi_i^s \leftarrow \pi_j^t)$ <sup>3</sup>, if one of the following requirements holds:

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

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

**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  $\pi_i^s$ :

- $crp_i$ : whether  $P_i$  is corrupted.
- $\mathbf{Aflag}_i^s$ : whether the intended partner is corrupted when  $\pi_i^s$  accepts.
- $T_i^s$ : whether  $\pi_i^s$  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.
- $\mathbf{FirstAcc}_i^s$ : 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 3.* We introduced variable  $\mathbf{FirstAcc}$  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.

- In general, the session key of the party which accepts the key after its partner (i.e.,  $\mathbf{FirstAcc} = \mathbf{false}$ ), 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.,  $\mathbf{FirstAcc} = \mathbf{true}$ ) 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.,  $\mathbf{FirstAcc} = \mathbf{false}$ ). This is a minimal trivial attack regarding state reveal<sup>10</sup>, and it is formalized as **TA6** and **TA7** in Table 2.

<sup>3</sup> The arrow notion  $\pi_i^s \leftarrow \pi_j^t$  means  $\pi_i^s$  (not necessarily  $\pi_j^t$ ) has computed and accepted the original key.

<sup>10</sup> It is also possible to define the trivial attack regardless of  $\mathbf{FirstAcc}$ , but our definition of **TA6** and **TA7** is minimal and makes our security model stronger.

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

Types	Trivial attacks	Explanation
<b>TA1</b>	$T_i^s = \text{true} \wedge \text{Aflag}_i^s = \text{true}$	$\pi_i^s$ is tested but $\pi_i^s$ 's partner is corrupted when $\pi_i^s$ accepts session key $k_i^s$
<b>TA2</b>	$T_i^s = \text{true} \wedge k\text{Rev}_i^s = \text{true}$	$\pi_i^s$ is tested and its session key $k_i^s$ is revealed
<b>TA3</b>	$T_i^s = \text{true}$ when $\text{Test}(i, s)$ is queried	$\text{Test}(i, s)$ is queried at least twice
<b>TA4</b>	$T_i^s = \text{true} \wedge \text{Partner}(\pi_i^s \leftrightarrow \pi_j^t) \wedge k\text{Rev}_j^t = \text{true}$	$\pi_i^s$ is tested, $\pi_i^s$ and $\pi_j^t$ are partnered to each other, and $\pi_j^t$ 's session key $k_j^t$ is revealed
<b>TA5</b>	$T_i^s = \text{true} \wedge \text{Partner}(\pi_i^s \leftrightarrow \pi_j^t) \wedge T_j^t = \text{true}$	$\pi_i^s$ is tested, $\pi_i^s$ and $\pi_j^t$ are partnered to each other, and $\pi_j^t$ is tested
<b>TA6</b>	$T_i^s = \text{true} \wedge \text{FirstAcc}_i^s = \text{false}$ $\wedge \text{stRev}_i^s = \text{true} \wedge \text{crp}_i = \text{true}$	$\pi_i^s$ is tested, $\pi_i^s$ accepts its key after its partner, and $\pi_i^s$ is both corrupted and has its state $\text{st}_i^s$ revealed
<b>TA7</b>	$T_i^s = \text{true} \wedge \text{Partner}(\pi_i^s \leftarrow \pi_j^t)$ $\wedge \text{FirstAcc}_j^t = \text{false} \wedge \text{stRev}_j^t = \text{true} \wedge \text{crp}_j = \text{true}$	$\pi_i^s$ is tested, $\pi_i^s$ accepts its session key before its partner, but its partner $\pi_j^t$ is both corrupted and state revealed

The following definition also captures replay attacks. Given  $\text{Partner}(\pi_i^{s'} \leftarrow \pi_j^t)$ , a successful replay attack means that  $\mathcal{A}$  resends to  $\pi_i^s$  the messages, which were sent from  $\pi_j^t$  to  $\pi_i^{s'}$ , and  $\pi_i^s$  is fooled to compute a session key, i.e.,  $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$ . Note that a stateless 2-pass AKE protocol cannot be secure against replay attacks [31]. Therefore, we also define security without replay attacks in Definition 15.

Furthermore, we distinguish between security with state reveals (Definition 13) and without state reveals (Definition 14), where in the latter the adversary cannot query the session state of an oracle.

**Definition 13 (Security of AKE with Replay Attacks and State Reveal).** Let  $\mu$  be the number of users and  $\ell$  the maximum number of protocol executions per user. The security experiment  $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay, state}}$  (see Fig. 5) is played between the challenger  $\mathcal{C}$  and the adversary  $\mathcal{A}$ .

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

• **Strong Authentication.** Let  $\text{Win}_{\text{Auth}}$  denote the event that  $\mathcal{A}$  breaks authentication in the security experiment.  $\text{Win}_{\text{Auth}}$  happens iff  $\exists (i, s) \in [\mu] \times [\ell]$  s.t.

- (1)  $\pi_i^s$  is  $\tau$ -accepted.
- (2)  $P_j$  is  $\hat{\tau}$ -corrupted with  $j := \text{Pid}_i^s$  and  $\hat{\tau} > \tau$ .
- (3) Either (3.1) or (3.2) or (3.3) happens<sup>11</sup>. Let  $j := \text{Pid}_i^s$ .
  - (3.1) There is no oracle  $\pi_j^t$  that  $\pi_i^s$  is partnered to.
  - (3.2) There exist two distinct oracles  $\pi_j^t$  and  $\pi_{j'}^{t'}$ , to which  $\pi_i^s$  is partnered.
  - (3.3) There exist two oracles  $\pi_{i'}^{s'}$  and  $\pi_j^t$  with  $(i', s') \neq (i, s)$ , such that both  $\pi_i^s$  and  $\pi_{i'}^{s'}$  are partnered to  $\pi_j^t$ .

• **Indistinguishability.** Let  $\text{Win}_{\text{Ind}}$  denote the event that  $\mathcal{A}$  breaks indistinguishability in the experiment  $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay, state}}$  above. Let  $b^*$  be  $\mathcal{A}$ 's output. Then  $\text{Win}_{\text{Ind}}$  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  $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay, state}} \Rightarrow 1$  iff  $\text{Win}_{\text{Ind}}$  happens. Hence, the advantage of  $\mathcal{A}$  is defined as

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

<sup>11</sup> 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.

**Definition 14 (Security of AKE with Replay Attacks and without State Reveal).** The security experiment  $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay}}$  (see Fig. 5) is defined like  $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay, state}}$  except that we disallow state reveal queries. Similarly, the advantage of an adversary  $\mathcal{A}$  in  $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay}}$  is defined as

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

**Definition 15 (Security of AKE without Replay Attack and State Reveal).** The security experiment  $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}$  (see Fig. 5) is defined like  $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}^{\text{replay, state}}$  except that we disallow replay attacks and state reveal queries. Similarly, the advantage of an adversary  $\mathcal{A}$  in  $\text{Exp}_{\text{AKE},\mu,\ell,\mathcal{A}}$  is defined as

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

*Remark 4 (Perfect Forward Security and KCI Resistance).* The security model of AKE supports (perfect) forward security (a.k.a. forward secrecy [23]). 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,  $\pi_i^s$  may be corrupted before  $\text{Test}(i, s)$ , which provides resistance to key-compromise impersonation (KCI) attacks [27].

## 5 AKE Protocols

We construct AKE protocols  $\text{AKE}_{2\text{msg}}$ ,  $\text{AKE}_{3\text{msg}}$  and  $\text{AKE}_{3\text{msg}}^{\text{state}}$  from a signature scheme SIG and a key encapsulation mechanism KEM. Additionally, we use a symmetric encryption scheme SE with key space  $\mathcal{K}_{\text{SE}}$  to encrypt the state in protocol  $\text{AKE}_{3\text{msg}}^{\text{state}}$ . Apart from that,  $\text{AKE}_{3\text{msg}}^{\text{state}}$  and  $\text{AKE}_{3\text{msg}}$  are the same. The protocols are given in Figure 6.

The setup algorithm generates the public parameter  $\text{pp}_{\text{AKE}} := (\text{pp}_{\text{SIG}}, \text{pp}_{\text{KEM}})$  by running  $\text{SIG.Setup}$  and  $\text{KEM.Setup}$ . The key generation algorithm inputs the public parameter and a party  $P_i$  and generates a signature key pair  $(vk_i, ssk_i)$ . In  $\text{AKE}_{3\text{msg}}^{\text{state}}$ , it also chooses a symmetric key  $s_i$  uniformly from the key space  $\mathcal{K}_{\text{SE}}$ . 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  $\text{res}_i$  and  $\text{res}_j$  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  $\Psi_i$ ,  $k_i$  and  $\text{st}_i$  with the empty string. To initiate a session in  $\text{AKE}_{3\text{msg}}$  and  $\text{AKE}_{3\text{msg}}^{\text{state}}$ , 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  $\text{AKE}_{2\text{msg}}$  is sent by  $P_i$ . It generates an ephemeral key pair  $(\hat{p}k, \hat{s}k)$  by running  $\text{KEM.Gen}(\text{pp}_{\text{KEM}})$  and computes a signature  $\sigma_1$  over the identities of  $P_i$  and  $P_j$ , the ephemeral public key and the nonce (only in  $\text{AKE}_{3\text{msg}}$  and  $\text{AKE}_{3\text{msg}}^{\text{state}}$ ). When using state encryption, it also encrypts the ephemeral secret key using its symmetric key  $s_i$  and stores the ciphertext in  $\text{st}_i$ . It then sends  $(\hat{p}k, \sigma_1)$  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  $(c, K) \leftarrow_s \text{Encap}(\hat{p}k)$ . It computes a signature  $\sigma_2$  over the identities as well as the previous message,  $c$  and the nonce (only in  $\text{AKE}_{3\text{msg}}$  and  $\text{AKE}_{3\text{msg}}^{\text{state}}$ ).  $P_j$  accepts the session key and sets  $k_j$  to  $K$ . It sends  $(c, \sigma_2)$  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.

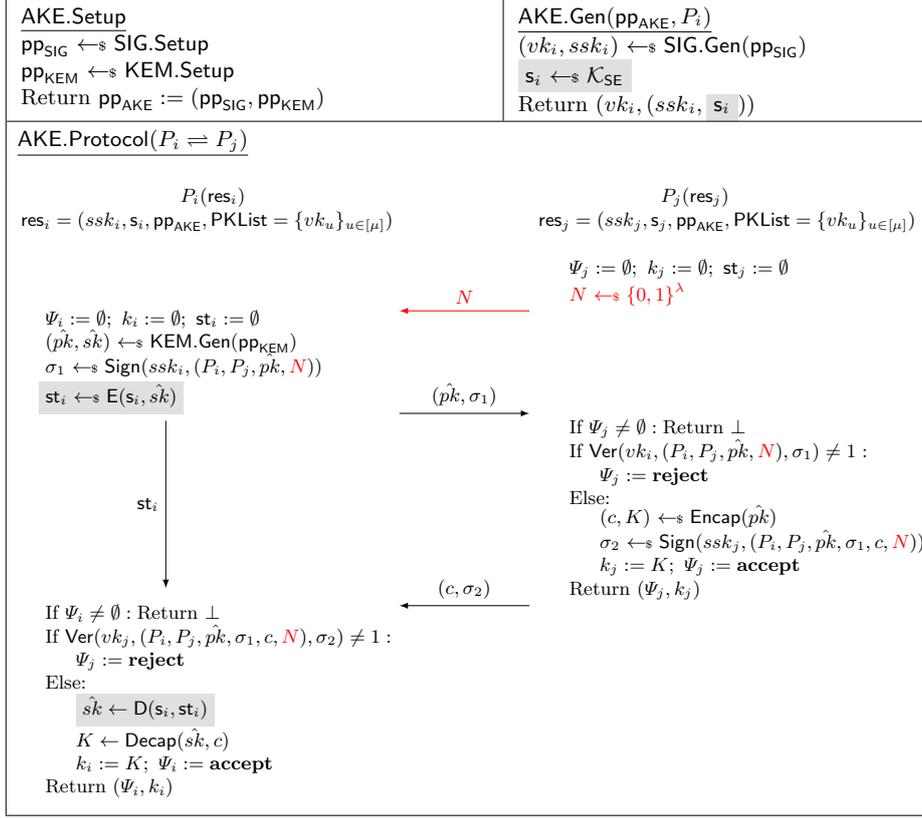
**Correctness.** Correctness of  $\text{AKE}_{2\text{msg}}$ ,  $\text{AKE}_{3\text{msg}}$  and  $\text{AKE}_{3\text{msg}}^{\text{state}}$  follows directly from the correctness of SIG, KEM and SE.

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

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

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

We first give a proof sketch, then present the formal proof of Theorem 1.



**Fig. 6.** Generic construction of  $\text{AKE}_{2\text{msg}}$  (without red and gray parts),  $\text{AKE}_{3\text{msg}}$  (with red and without gray parts) and  $\text{AKE}_{3\text{msg}}^{\text{state}}$  (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.

*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  $pk$  and ciphertexts  $c$  that are used for test queries. Due to the nonce, the adversary can also not replay a message containing a particular  $pk$ . Thus, each  $pk$  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  $\text{Encap}$  and  $\text{Encap}^*$  are indistinguishable, even given  $\hat{sk}$ . So, we first switch from  $\text{Encap}$  to  $\text{Encap}^*$ . 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  $\eta$ -th hybrid, we replace the test session keys for party  $P_\eta$ . We can show that this is unnoticeable using that the key  $K^*$  generated by  $\text{Encap}^*$  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  $P_\eta$ . 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  $\text{E}(s_\eta, \hat{sk})$  and thus we can replace the session key by  $\epsilon$ -uniformity of  $\text{Encap}^*$ .

**Case 2:** The adversary does not corrupt  $P_\eta$ . 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  $\epsilon$ -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  $2\mu$  only in the IND-mRPA security of the symmetric encryption scheme. All other parts are tight.

**Proof of Theorem 1.** For the proof, we will first define two further variables  $\text{Sent}_i^s$  and  $\text{Recv}_i^s$  for an oracle  $\pi_i^s$ . The set  $\text{Sent}_i^s$  will store outgoing messages of the oracle and the set  $\text{Recv}_i^s$  will store incoming messages, respectively. We stress that  $\text{Recv}_i^s$  will only store *valid* messages, e.g., the signature needs to be valid.

**Message Consistency.** For our 3-move protocol given in Figure 6, we say that an oracle  $\pi_i^s$  is *message-consistent* with another oracle  $\pi_j^t$ , denoted by  $\text{MsgCon}(\pi_i^s \leftarrow \pi_j^t)$ , if  $\text{Pid}_i^s := j$  and  $\text{Pid}_j^t := i$  and either

- (1)  $\pi_i^s$  has sent the first message, the same nonce  $N$  is contained in  $\text{Sent}_i^s$  and  $\text{Recv}_j^t$  and the same ephemeral key  $\hat{pk}$  is contained in  $\text{Recv}_i^s$  and  $\text{Sent}_j^t$ , or
- (2)  $\pi_i^s$  has received the first message, the same nonce  $N$  and ciphertext  $c$  are contained in  $\text{Recv}_i^s$  and  $\text{Sent}_j^t$  and the same ephemeral key  $\hat{pk}$  is contained in  $\text{Sent}_i^s$  and  $\text{Recv}_j^t$ .

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

To prove the theorem, we now consider the sequence of games  $G_0$ - $G_5$ . In the following, we describe the games and show that adjacent games are indistinguishable. Let  $\text{Win}_i$  denote the probability that  $G_i$  returns 1.

**Game  $G_0$ :**  $G_0$  is the original experiment  $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}^{\text{replay, state}}$ . In addition to the original game, we add the sets  $\text{Sent}_i^s$  and  $\text{Recv}_i^s$  which is only a conceptual change. We have

$$\Pr[\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}^{\text{replay, state}} \Rightarrow 1] = \Pr[\text{Win}_0] .$$

**Game  $G_1$ :** In  $G_1$ , we define the event **Repeat** which happens if a nonce repeats for any two oracles of the same party. If **Repeat** happens, the game aborts (see also Figure 7). Due to the difference lemma,

$$|\Pr[\text{Win}_0] - \Pr[\text{Win}_1]| \leq \Pr[\text{Repeat}] .$$

Using the birthday paradox and union bound over the number of parties, we have  $\Pr[\text{Repeat}] \leq \mu\ell^2 \cdot 2^{-\lambda}$ , where  $\lambda$  is the length of the nonce in bits.

**Game  $G_2$ :** In  $G_2$ , we define the event **NoMsgCon** which happens if there exists some  $(i, s)$  such that  $\pi_i^s$  accepts, the intended partner  $j := \text{Pid}_i^s$  is uncorrupted when  $\pi_i^s$  accepts, and there does not exist  $t \in [\ell]$  such that  $\pi_i^s$  is message-consistent with  $\pi_j^t$ . If event **NoMsgCon** happens, the game will abort (see also Figure 7). Due to the difference lemma,

$$|\Pr[\text{Win}_1] - \Pr[\text{Win}_2]| \leq \Pr[\text{NoMsgCon}] .$$

We will prove the following lemma.

**Lemma 1.** *There exists an adversary  $\mathcal{B}_{\text{SIG}}$  against SIG such that*

$$\Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.1)] \leq \Pr[\text{NoMsgCon}] \leq \text{Adv}_{\text{SIG}, \mu}^{\text{mu-corr}}(\mathcal{B}_{\text{SIG}}) .$$

*Proof.* If there exists an oracle  $\pi_j^t$  such that  $\pi_i^s$  is message-consistent with  $\pi_j^t$ , then due to correctness of KEM,  $\pi_i^s$  is also partnered to  $\pi_j^t$ . It follows that  $\Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.1)] \leq \Pr[\text{NoMsgCon}]$ .

To prove that  $\Pr[\text{NoMsgCon}] \leq \text{Adv}_{\text{SIG}, \mu}^{\text{mu-corr}}(\mathcal{B}_{\text{SIG}})$ , we construct adversary  $\mathcal{B}_{\text{SIG}}$  against MU-EUF-CMA<sup>corr</sup> security of SIG.  $\mathcal{B}_{\text{SIG}}$  inputs the public parameter  $\text{pp}_{\text{SIG}}$  and a list of verification keys  $\{vk_i\}_{i \in [\mu]}$  and has access to a signing oracle  $\mathcal{O}_{\text{SIGN}}(\cdot, \cdot)$  and a corrupt oracle  $\mathcal{O}_{\text{CORR}}(\cdot)$ .  $\mathcal{B}_{\text{SIG}}$  then runs  $\text{pp}_{\text{KEM}} \leftarrow \text{KEM.Setup}$  and sets  $\text{pp}_{\text{AKE}} := (\text{pp}_{\text{SIG}}, \text{pp}_{\text{KEM}})$  and  $\text{PKList} := \{vk_i\}_{i \in [\mu]}$ . It chooses symmetric keys  $s_i$  for each user  $i \in [\mu]$ , initializes all variables and then runs  $\mathcal{A}$  on  $\text{pp}_{\text{AKE}}$  and  $\text{PKList}$ . If  $\mathcal{A}$  queries  $\mathcal{O}_{\text{AKE}}$ ,  $\mathcal{B}_{\text{SIG}}$  responds as follows.

- **Send** $(i, s, j, \text{msg} = N)$ : In order to get  $\sigma_1$ ,  $\mathcal{B}_{\text{SIG}}$  queries its signing oracle  $\mathcal{O}_{\text{SIGN}}(i, (P_i, P_j, \hat{pk}, N))$ .
- **Send** $(i, s, j, \text{msg} = (pk, \sigma_1))$ : In order to get  $\sigma_2$ ,  $\mathcal{B}_{\text{SIG}}$  queries its signing oracle  $\mathcal{O}_{\text{SIGN}}(i, (P_j, P_i, pk, \sigma_1, c, N))$ .

- **Corrupt**( $i$ ):  $\mathcal{B}_{\text{SIG}}$  queries its own oracle  $\mathcal{O}_{\text{CORR}}(i)$  and receives the signing key  $ssk_i$ . It returns  $(ssk_i, s_i)$  to  $\mathcal{A}$ .
- Queries **Send**( $i, s, j, \top$ ), **Send**( $i, s, j, (c, \sigma_2)$ ), **RegisterCorrupt**, **StateReveal**, **SessionKeyReveal** and **Test** can be simulated as in the original experiment  $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}^{\text{replay, state}}$ .

During the simulation,  $\mathcal{B}_{\text{SIG}}$  checks if **NoMsgCon** happens. If this is the case, there exists an oracle  $\pi_i^s$  such that  $\pi_i^s$  has accepted and  $j := \text{Pid}_i^s$  is uncorrupted at that point in time.

Now we show that then there is a valid message-signature pair  $(m^*, \sigma^*)$  in  $\text{Sent}_i^s$  and  $\text{Recv}_i^s$  such that  $\text{Ver}(vk_j, m^*, \sigma^*) = 1$  and  $m^*$  is different from any message  $m$  signed by  $\pi_j^t$  for all  $t \in [\ell]$ . Since  $\pi_i^s$  is accepted,  $\text{Sent}_i^s \neq \emptyset$  and  $\text{Recv}_i^s \neq \emptyset$ .

**Case 1:**  $\pi_i^s$  sent the first message. Let  $\text{Sent}_i^s = \{N, (c, \sigma_2)\}$  and  $\text{Recv}_i^s = \{(\hat{p}k, \sigma_1)\}$ . We have  $\text{Ver}(vk_j, (P_j, P_i, \hat{p}k, N), \sigma_1) = 1$ , since  $\text{Recv}_i^s \neq \emptyset$ . For any oracle  $\pi_j^t$  with  $\text{Recv}_j^t = \{N', \cdot\}$  and  $\text{Sent}_j^t = \{(\hat{p}k', \sigma_1')\} \neq \emptyset$ , **NoMsgCon** implies that  $(\hat{p}k, N) \neq (\hat{p}k', N')$ . In this case,  $\mathcal{B}_{\text{SIG}}$  sets  $(m^*, \sigma^*) := ((P_j, P_i, \hat{p}k, N), \sigma_1)$ .

**Case 2:**  $\pi_i^s$  received the first message. Let  $\text{Recv}_i^s = \{N, (c, \sigma_2)\}$  and  $\text{Sent}_i^s = \{(\hat{p}k, \sigma_1)\}$ . We have  $\text{Ver}(vk_j, (P_i, P_j, \hat{p}k, \sigma_1, c, N), \sigma_2) = 1$ , since  $(c, \sigma_2) \in \text{Recv}_i^s$ . For any oracle  $\pi_j^t$  with  $\text{Recv}_j^t = \{(\hat{p}k', \sigma_1')\} \neq \emptyset$  and  $\text{Sent}_j^t = \{N', (c', \sigma_2')\} \neq \emptyset$ , **NoMsgCon** implies that  $(\hat{p}k, c, N) \neq (\hat{p}k', c', N')$ . In this case,  $\mathcal{B}_{\text{SIG}}$  sets  $(m^*, \sigma^*) := ((P_i, P_j, \hat{p}k, \sigma_1, c, N), \sigma_2)$ .

As soon as event **NoMsgCon** happens,  $\mathcal{B}_{\text{SIG}}$  retrieves the message-signature  $(m^*, \sigma^*)$  pair as just described and outputs  $(j, m^*, \sigma^*)$ . As  $P_j$  is uncorrupted,  $\mathcal{B}_{\text{SIG}}$  has not queried  $\mathcal{O}_{\text{CORR}}(j)$  and  $m^*$  is different from all signing queries for  $j$ , which concludes the proof of Lemma 1.  $\square$

Before moving to  $\mathsf{G}_3$ , let us bound (1)  $\wedge$  (2)  $\wedge$  (3.2) and (1)  $\wedge$  (2)  $\wedge$  (3.3).

**Multiple Partners.** Event (1)  $\wedge$  (2)  $\wedge$  (3.2) happens if there exists any oracle  $\pi_i^s$  that has accepted with  $\text{Aflag}_i^s = \text{false}$  and has more than one partner oracle. We can show that

$$\Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.2)] \leq (\mu\ell)^2 \cdot 2^{-\gamma}.$$

The session key only depends on the ephemeral public key  $\hat{p}k$  and the ciphertext  $c$ . In the following, we assume that there are two oracles  $\pi_j^t$  and  $\pi_{j'}^{t'}$ , such that  $\pi_i^s$  is partnered to both  $\pi_j^t$  and  $\pi_{j'}^{t'}$ . We distinguish two cases:

**Case 1:**  $\pi_i^s$  sent the first message. Let  $\hat{p}k$  and  $\hat{p}k'$  be the public keys determined by the internal randomness of  $\pi_j^t$  and  $\pi_{j'}^{t'}$ , respectively. Let  $r$  be the internal randomness of  $\pi_i^s$  which is used by **Encap**. The original keys are derived from  $(c, K) \leftarrow \text{Encap}(\hat{p}k; r)$  and  $(c', K') \leftarrow \text{Encap}(\hat{p}k'; r)$ . As  $\pi_i^s$  is partnered to both oracles,  $k_i^s = K = K'$ . Due to  $\gamma$ -diversity of KEM, this will happen only with probability at most  $2^{-\gamma}$ .

**Case 2:**  $\pi_i^s$  received the first message. Let  $\hat{p}k$  be the ephemeral public key determined by the internal randomness of  $\pi_i^s$ . Let  $(c, K) \leftarrow \text{Encap}(\hat{p}k; r)$  and  $(c', K') \leftarrow \text{Encap}(\hat{p}k; r')$ , where  $r, r'$  is the internal randomness of  $\pi_j^t$  and  $\pi_{j'}^{t'}$ , respectively. As  $\pi_i^s$  is partnered to both oracles, this implies that  $k_i^s = \text{Decap}(\hat{sk}, c) = \text{Decap}(\hat{sk}, c')$ . By the correctness and  $\gamma$ -diversity of KEM, we have  $k_i^s = K = K'$  which will happen with probability at most  $2^{-\gamma}$ .

As there are  $\mu\ell$  oracles, we can upper bound the probability for event (1)  $\wedge$  (2)  $\wedge$  (3.2) by  $(\mu\ell)^2 \cdot 2^{-\gamma}$ .

**Replay Attacks.** Event (1)  $\wedge$  (2)  $\wedge$  (3.3) covers replay attacks and happens if there exists any oracle  $\pi_i^s$  that has accepted with  $\text{Aflag}_i^s = \text{false}$ , is partnered to an oracle  $\pi_j^t$ , and there exists another oracle  $\pi_{j'}^{s'}$  such that this oracle is also partnered to  $\pi_j^t$ . We will show

$$\begin{aligned} \Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.3)] &\leq \Pr[\text{Repeat}] + \Pr[\text{NoMsgCon}] + \Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.2)] \\ &\leq \text{Adv}_{\text{SIG}, \mu}^{\text{mu-corr}}(\mathcal{B}_{\text{SIG}}) + (\mu\ell)^2 \cdot 2^{-\gamma} + \mu\ell^2 \cdot 2^{-\lambda}. \end{aligned}$$

We bound (1)  $\wedge$  (2)  $\wedge$  (3.3) by using previous observations. Assume that **NoMsgCon** and (1)  $\wedge$  (2)  $\wedge$  (3.2) do not occur. Then for each oracle  $\pi_i^s$  there exists a unique oracle  $\pi_j^t$  such that  $\pi_i^s$  is partnered to and message-consistent with  $\pi_j^t$ . Now assume there exists another oracle  $\pi_{j'}^{s'}$  that is also partnered to and message-consistent with  $\pi_j^t$ . Message-consistency implies that  $i = i'$ . Now let  $s \neq s'$ .

**Case 1:**  $\pi_i^s$  sent the first message. Then, the three sets  $\text{Recv}_i^s, \text{Recv}_{i'}^{s'}, \text{Sent}_j^t$  all contain the same ephemeral public key  $\hat{pk}$  and no other oracle than  $\pi_j^t$  has output  $\hat{pk}$ . Let  $\text{Sent}_i^s = \{N, (c, \sigma_2)\}$ , then  $\text{Sent}_{i'}^{s'} = \{N, (c', \sigma_2')\}$  shares the same nonce  $N$ . However, this will only happen if Repeat happens.

**Case 2:**  $\pi_i^s$  received the first message. Then, the three sets  $\text{Sent}_i^s, \text{Sent}_{i'}^{s'}, \text{Recv}_j^t$  all contain the same ephemeral public key  $\hat{pk}$  and the sets  $\text{Recv}_i^s, \text{Recv}_{i'}^{s'}, \text{Sent}_j^t$  contain the same nonce  $N$  and ciphertext  $c$ . This means that  $\pi_j^t$  is partnered to both  $\pi_i^s$  and  $\pi_{i'}^{s'}$  and thus contradicts to the fact that each oracle has a unique partner.

At this point note that

$$\begin{aligned} \Pr[\text{Win}_{\text{Auth}}] &= \Pr_{\exists(i,s)} [(1) \wedge (2) \wedge ((3.1) \vee (3.2) \vee (3.3))] \\ &\leq \Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.1)] + \Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.2)] + \Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.3)] \\ &\leq 2 \cdot \text{Adv}_{\text{SIG}, \mu}^{\text{mu-corr}}(\mathcal{B}_{\text{SIG}}) + 2(\mu\ell)^2 \cdot 2^{-\gamma} + \mu\ell^2 \cdot 2^{-\lambda}. \end{aligned}$$

The analysis of  $\text{Win}_{\text{Auth}}$  will be helpful for the next game hop. The following games  $\text{G}_3$ - $\text{G}_5$  are also given in Figure 7.

**Game  $\text{G}_3$ :** In  $\text{G}_3$ , we check the partnership  $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$  by message-consistency  $\text{MsgCon}(\pi_i^s \leftarrow \pi_j^t)$  if  $\Psi_i^s = \text{accept}$  and  $\text{Aflag}_i^s = \text{false}$ . We claim that

$$|\Pr[\text{Win}_2] - \Pr[\text{Win}_3]| \leq \Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.2)] \leq (\mu\ell)^2 \cdot 2^{-\gamma}.$$

Recall that if  $\text{NoMsgCon}$  does not happen, we know that each oracle  $\pi_i^s$  that has accepted with  $\text{Aflag}_i^s = \text{false}$  is partnered to and message-consistent with an oracle  $\pi_j^t$ . If any such oracle  $\pi_i^s$  has a unique partner, then  $\text{G}_2$  is identical to  $\text{G}_3$ . On the other hand, the probability that there exists an oracle  $\pi_i^s$  that has accepted with  $\text{Aflag}_i^s = \text{false}$  and has multiple partners is  $\Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.2)]$ , which is bounded by  $(\mu\ell)^2 \cdot 2^{-\gamma}$ . Thus, the claims follows by the difference lemma.

**Game  $\text{G}_4$ :** In  $\text{G}_4$ , we use the  $\text{Encap}^*$  algorithm (instead of  $\text{Encap}$ ) whenever  $\mathcal{A}$  issues a  $\text{Send}$  query  $(i, s, j, \text{msg})$  with a second protocol message  $\text{msg} = (\hat{pk}, \sigma_1)$  and the intended partner  $P_j$  is not corrupted. We construct adversary  $\mathcal{B}_{\text{KEM}}$  against indistinguishability of  $\text{Encap}$  and  $\text{Encap}^*$ .

$\mathcal{B}_{\text{KEM}}$  inputs the public parameter  $\text{pp}_{\text{KEM}}$  and  $\{pk_n, sk_n, c_n, K_n\}_{n \in [\mu\ell]}$ , where  $(c_n, K_n)$  are either computed by  $\text{Encap}(pk_n)$  or by  $\text{Encap}^*(sk_n)$ .  $\mathcal{B}_{\text{KEM}}$  generates the public parameter for  $\text{SIG}$  and signature key pairs  $(vk_i, ssk_i)$  for  $i \in [\mu]$ , as well as symmetric keys  $s_i$ . It sets  $\text{PKList} := \{vk_i\}_{i \in [\mu]}$ , initializes all variables, chooses  $b \leftarrow_{\$} \{0, 1\}$  and runs  $\mathcal{A}$ . If  $\mathcal{A}$  makes a query to  $\mathcal{O}_{\text{AKE}}$ ,  $\mathcal{B}_{\text{KEM}}$  simulates the response as follows:

- $\text{Send}(i, s, j, \text{msg} = N)$ :  $\mathcal{B}_{\text{KEM}}$  uses the key pair with index  $(i-1)\mu + s$  as ephemeral key pair, i.e.  $(pk, sk) := (pk_{(i-1)\mu+s}, sk_{(i-1)\mu+s})$ .
- $\text{Send}(i, s, j, \text{msg} = (pk, \sigma_1))$ : If  $P_j$  is uncorrupted, then due to the fact that event  $\text{NoMsgCon}$  does not happen, there exists a unique oracle  $\pi_j^t$  such that  $\hat{pk}$  was output by  $\pi_j^t$ . Furthermore,  $n = (j-1)\mu + t$  is the index of that public key. Then  $\mathcal{B}_{\text{KEM}}$  uses  $(c_n, K_n)$  as ciphertext and key. If  $P_j$  is corrupted,  $\mathcal{B}_{\text{KEM}}$  runs  $\text{Encap}(\hat{pk})$  itself to compute  $(c, K)$ .
- Queries  $\text{Send}(i, s, j, \top)$ ,  $\text{Send}(i, s, j, (c, \sigma_2))$ ,  $\text{Corrupt}$ ,  $\text{RegisterCorrupt}$ ,  $\text{Test}$ ,  $\text{StateReveal}$  and  $\text{SessionKeyReveal}$  can be simulated as in  $\text{G}_3$  and  $\text{G}_4$ , except for the partnership check  $\text{Partner}(\pi_i^s \leftrightarrow \pi_j^t)$ . Recall that  $\mathcal{B}_{\text{KEM}}$  needs to compute  $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$  in  $\text{Send}(i, s, j, \text{msg})$  to set  $\text{FirstAcc}$ , in  $\text{Test}(i, s)$  and  $\text{StateReveal}(i, s)$  to detect **TA7**, and compute  $\text{Partner}(\pi_i^s \leftrightarrow \pi_j^t)$  in  $\text{Test}(i, s)$  and  $\text{SessionKeyReveal}(i, s)$  to detect **TA4** and **TA5**.
  - For the set of  $\text{FirstAcc}$  in  $\text{Send}(i, s, j, \text{msg})$  and for the detection of **TA7** in  $\text{Test}(i, s)$  and  $\text{StateReveal}(i, s)$ ,  $\mathcal{B}_{\text{KEM}}$  simulates  $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$  with  $\text{MsgCon}(\pi_i^s \leftarrow \pi_j^t)$ . This simulation is perfect since  $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$  is involved only when  $\text{Aflag}_i^s = \text{false}$ .
  - For the detection of **TA4** and **TA5** in  $\text{Test}(i, s)$  and  $\text{SessionKeyReveal}(i, s)$ ,  $\mathcal{B}_{\text{KEM}}$  simulates  $\text{Partner}(\pi_i^s \leftrightarrow \pi_j^t)$  as follows.  $\mathcal{B}_{\text{KEM}}$  first checks  $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$  with  $\text{MsgCon}(\pi_i^s \leftarrow \pi_j^t)$ . (Again, since  $\text{Partner}(\pi_i^s \leftrightarrow \pi_j^t)$  is involved only when  $\text{Aflag}_i^s = \text{false}$ ,  $\text{Partner}(\pi_i^s \leftarrow \pi_j^t) = \text{MsgCon}(\pi_i^s \leftarrow \pi_j^t)$ .) If  $\text{Partner}(\pi_i^s \leftarrow \pi_j^t) = 0$ ,  $\mathcal{B}_{\text{KEM}}$  outputs 0 directly for  $\text{Partner}(\pi_i^s \leftrightarrow \pi_j^t)$ . Otherwise,  $\pi_i^s$  is partner to and message-consistent with  $\pi_j^t$ , hence  $k_i^s$  must be equal to the

<p style="text-align: center;"><math>G_3, G_4, G_{4,\eta,0}, G_{4,\eta,1}, G_{4,\eta,2}, G_5</math></p> <p><math>pp_{SIG} \leftarrow \text{SIG.Setup}</math>  <math>pp_{KEM} \leftarrow \text{KEM.Setup}</math>  For <math>i \in [\mu]</math>:  <math>(vk_i, ssk_i) \leftarrow \text{SIG.Gen}(pp_{SIG})</math>;  <math>s_i \leftarrow \mathcal{K}_{SE}</math>  <math>crp_i := \text{false}</math>  PKList := <math>\{vk_i\}_{i \in [\mu]}</math>; <math>b \leftarrow \{0,1\}</math>  For <math>(i, s) \in [\mu] \times [\ell]</math>:  <math>\text{var}_i^s := (st_i^s, \text{Pid}_i^s, k_i^s, \Psi_i^s) := (\emptyset, \emptyset, \emptyset, \emptyset)</math>  <math>(\text{Sent}_i^s, \text{Recv}_i^s) := (\emptyset, \emptyset)</math>  <math>\text{Aflag}_i^s := \text{false}</math>; <math>\text{FirstAcc}_i^s := \emptyset</math>  <math>T_i^s := \text{false}</math>; <math>kRev_i^s := \text{false}</math>; <math>stRev_i^s := \text{false}</math>  Repeat := <b>false</b>; NoMsgCon := <b>false</b>  <math>b^* \leftarrow \mathcal{A}^{\mathcal{O}_{AKE}(\cdot)}(pp_{AKE}, \text{PKList})</math></p> <p>// During the execution the game checks if one of the following  // flags is set to true and if so, it aborts immediately:  Repeat := <b>true</b>, If <math>\exists i, s, s' \in [\mu] \times [\ell]^2, N \in \{0,1\}^\lambda</math> s.t.  <math>N \in \text{Sent}_i^s \wedge N \in \text{Sent}_{i'}^{s'}</math>  NoMsgCon := <b>true</b>, If <math>\exists i, s \in [\mu] \times [\ell]</math> s.t. (1') <math>\wedge</math> (2') <math>\wedge</math> (3').  Let <math>j := \text{Pid}_i^s</math>.  (1') <math>\Psi_i^s = \text{accept}</math>  (2') <math>\text{Aflag}_i^s = \text{false}</math>  (3') <math>\nexists t \in [\ell]</math> s.t. <math>\pi_i^s</math> is message-consistent with <math>\pi_j^t</math></p> <p>Win<sub>Ind</sub> := <b>false</b>  If <math>b^* = b</math>: Win<sub>Ind</sub> = <b>true</b>; Return 1  Else: Return 0</p> <p><math>\mathcal{O}_{AKE}(\text{query})</math>:  If query = Test(<math>i, s</math>):  If <math>\Psi_i^s \neq \text{accept} \vee \text{Aflag}_i^s = \text{true} \vee kRev_i^s = \text{true} \vee T_i^s = \text{true}</math>:  Return <math>\perp</math>  If <math>\text{FirstAcc}_i^s = \text{false}</math>:  If <math>crp_i = \text{true} \wedge stRev_i^s = \text{true}</math>: Return <math>\perp</math>  Let <math>j := \text{Pid}_i^s</math>  If <math>\exists t \in [\ell]</math> s.t. <math>\text{Partner}(\pi_i^s \leftarrow \pi_j^t)</math>:  If <math>kRev_j^t = \text{true} \vee T_j^t = \text{true}</math>: Return <math>\perp</math>  If <math>\exists t \in [\ell]</math> s.t. <math>\text{Partner}(\pi_i^s \leftarrow \pi_j^t)</math>:  If <math>\text{FirstAcc}_j^t = \text{false} \wedge crp_j = \text{true} \wedge stRev_j^t = \text{true}</math>:  Return <math>\perp</math>  <math>T_i^s := \text{true}</math>  <math>k_0 := k_i^s</math></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;"> If <math>\text{FirstAcc}_i^s = \text{false}</math>:  <math>k_0 := \begin{cases} k \leftarrow \mathcal{K} &amp; \text{if } i &lt; \eta \\ k_i^s &amp; \text{if } i \geq \eta \end{cases}</math>  If <math>\text{FirstAcc}_i^s = \text{true}</math>:  Let <math>\pi_j^t</math> be the partner of <math>\pi_i^s</math>  <math>k_0 := \begin{cases} k \leftarrow \mathcal{K} &amp; \text{if } j &lt; \eta \\ k_i^s &amp; \text{if } j \geq \eta \end{cases}</math> </td> <td style="padding: 5px;"> If <math>\text{FirstAcc}_i^s = \text{false}</math>:  <math>k_0 := \begin{cases} k \leftarrow \mathcal{K} &amp; \text{if } i \leq \eta \\ k_i^s &amp; \text{if } i &gt; \eta \end{cases}</math>  If <math>\text{FirstAcc}_i^s = \text{true}</math>:  Let <math>\pi_j^t</math> be the partner of <math>\pi_i^s</math>  <math>k_0 := \begin{cases} k \leftarrow \mathcal{K} &amp; \text{if } j \leq \eta \\ k_i^s &amp; \text{if } j &gt; \eta \end{cases}</math> </td> </tr> </table> <p><math>k_0 \leftarrow \mathcal{K}</math>  <math>k_1 \leftarrow \mathcal{K}</math>; Return <math>k_b</math></p>	If $\text{FirstAcc}_i^s = \text{false}$ : $k_0 := \begin{cases} k \leftarrow \mathcal{K} & \text{if } i < \eta \\ k_i^s & \text{if } i \geq \eta \end{cases}$ If $\text{FirstAcc}_i^s = \text{true}$ : Let $\pi_j^t$ be the partner of $\pi_i^s$ $k_0 := \begin{cases} k \leftarrow \mathcal{K} & \text{if } j < \eta \\ k_i^s & \text{if } j \geq \eta \end{cases}$	If $\text{FirstAcc}_i^s = \text{false}$ : $k_0 := \begin{cases} k \leftarrow \mathcal{K} & \text{if } i \leq \eta \\ k_i^s & \text{if } i > \eta \end{cases}$ If $\text{FirstAcc}_i^s = \text{true}$ : Let $\pi_j^t$ be the partner of $\pi_i^s$ $k_0 := \begin{cases} k \leftarrow \mathcal{K} & \text{if } j \leq \eta \\ k_i^s & \text{if } j > \eta \end{cases}$	<p><math>\mathcal{O}_{AKE}(\text{query})</math>:  If query = Send(<math>i, s, j, \text{msg}</math>):  If <math>\Psi_i^s = \text{accept}</math>: Return <math>\perp</math>  If <math>\text{msg} = \top</math>: //session is initiated  <math>\text{Pid}_i^s := j</math>;  <math>N \leftarrow \{0,1\}^\lambda</math>  <math>\text{msg}' := N</math>  If <math>\text{msg} = N</math>: //first message  <math>\text{Pid}_i^s := j</math>  <math>(pk, \hat{sk}) \leftarrow \text{KEM.Gen}</math>  <math>\sigma_1 \leftarrow \text{Sign}(ssk_i, (P_i, P_j, \hat{pk}, N))</math>  <math>st_i^s \leftarrow \mathcal{E}(s_i, \hat{sk})</math></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;"> If <math>i = \eta</math>:  <math>sk_i^s := \hat{sk}</math>  <math>r \leftarrow \mathcal{S}\mathcal{K}</math>; <math>st_i^s \leftarrow \mathcal{E}(s_i, r)</math> </td> </tr> </table> <p><math>\text{msg}' := (pk, \sigma_1)</math>  If <math>\text{msg} = (pk, \sigma_1)</math>: //second message  Choose <math>N \in \text{Sent}_i^s</math>  If <math>\text{Pid}_i^s \neq j</math> or <math>\text{Ver}(vk_j, (P_j, P_i, \hat{pk}, N), \sigma_1) \neq 1</math>:  <math>\Psi_i^s := \text{reject}</math>; Return <math>\perp</math>  <math>(c, K) \leftarrow \text{Encap}(pk)</math></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;"> If <math>crp_j = \text{false}</math>:  Then <math>\exists</math> unique <math>t</math> s.t. <math>\hat{pk}</math> output by <math>\pi_j^t</math>  Choose the corresponding <math>sk</math>  <math>(c, K) \leftarrow \text{Encap}^*(sk)</math> </td> </tr> </table> <p><math>\sigma_2 \leftarrow \text{Sign}(ssk_i, (P_j, P_i, \hat{pk}, \sigma_1, c, N))</math>  <math>k_i^s := K</math>; <math>\Psi_i^s := \text{accept}</math>  <math>\text{msg}' := (c, \sigma_2)</math>  If <math>\text{msg} = (c, \sigma_2)</math>: //third message  Choose <math>N \in \text{Recv}_i^s</math> and <math>(pk, \sigma_1) \in \text{Sent}_i^s</math>  If <math>\text{Pid}_i^s \neq j</math> or <math>\text{Ver}(vk_j, (P_i, P_j, \hat{pk}, \sigma_1, c, N), \sigma_2) \neq 1</math>:  <math>\Psi_i^s := \text{reject}</math>; Return <math>\perp</math>  <math>\hat{sk} \leftarrow \mathcal{D}(s_i, st_i^s)</math></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;"> If <math>i = \eta</math>: <math>sk := sk_i^s</math>  <math>K \leftarrow \text{Decap}(\hat{sk}, c)</math>  <math>st_i^s := \emptyset</math>; <math>k_i^s := K</math>; <math>\Psi_i^s := \text{accept}</math>  <math>\text{msg}' := \emptyset</math> </td> </tr> </table> <p><math>\text{Recv}_i^s := \text{Recv}_i^s \cup \{\text{msg}'\}</math>; <math>\text{Sent}_i^s := \text{Sent}_i^s \cup \{\text{msg}'\}</math>  If <math>\Psi_i^s = \text{accept}</math>:  If <math>crp_j = \text{true}</math>: <math>\text{Aflag}_i^s := \text{true}</math>  If <math>crp_j = \text{false}</math>: <math>\exists t \in [\ell]</math> s.t. <math>\text{Partner}(\pi_i^s \leftarrow \pi_j^t)</math>:  If <math>\Psi_j^t \neq \text{accept}</math>:  <math>\text{FirstAcc}_i^s := \text{true}</math>; <math>\text{FirstAcc}_j^t := \text{false}</math>  If <math>\Psi_j^t = \text{accept}</math>:  <math>\text{FirstAcc}_i^s := \text{false}</math>; <math>\text{FirstAcc}_j^t := \text{true}</math>  Return <math>\text{msg}'</math></p> <p><math>\text{Partner}(\pi_i^s \leftarrow \pi_j^t)</math>  If <math>\Psi_i^s = \text{accept} \wedge \text{Aflag}_i^s = \text{false}</math>: //message-consistency check  Return <math>\text{MsgCon}(\pi_i^s \leftarrow \pi_j^t)</math>  Else: //original partnership check  If <math>\pi_i^s</math> sent the first message  and <math>k_i^s = \mathcal{K}(\pi_i^s, \pi_j^t) \neq \emptyset</math>: Return 1  If <math>\pi_i^s</math> received the first message  and <math>k_i^s = \mathcal{K}(\pi_j^t, \pi_i^s) \neq \emptyset</math>: Return 1  Return 0</p>	If $i = \eta$ : $sk_i^s := \hat{sk}$ $r \leftarrow \mathcal{S}\mathcal{K}$ ; $st_i^s \leftarrow \mathcal{E}(s_i, r)$	If $crp_j = \text{false}$ : Then $\exists$ unique $t$ s.t. $\hat{pk}$ output by $\pi_j^t$ Choose the corresponding $sk$ $(c, K) \leftarrow \text{Encap}^*(sk)$	If $i = \eta$ : $sk := sk_i^s$ $K \leftarrow \text{Decap}(\hat{sk}, c)$ $st_i^s := \emptyset$ ; $k_i^s := K$ ; $\Psi_i^s := \text{accept}$ $\text{msg}' := \emptyset$
If $\text{FirstAcc}_i^s = \text{false}$ : $k_0 := \begin{cases} k \leftarrow \mathcal{K} & \text{if } i < \eta \\ k_i^s & \text{if } i \geq \eta \end{cases}$ If $\text{FirstAcc}_i^s = \text{true}$ : Let $\pi_j^t$ be the partner of $\pi_i^s$ $k_0 := \begin{cases} k \leftarrow \mathcal{K} & \text{if } j < \eta \\ k_i^s & \text{if } j \geq \eta \end{cases}$	If $\text{FirstAcc}_i^s = \text{false}$ : $k_0 := \begin{cases} k \leftarrow \mathcal{K} & \text{if } i \leq \eta \\ k_i^s & \text{if } i > \eta \end{cases}$ If $\text{FirstAcc}_i^s = \text{true}$ : Let $\pi_j^t$ be the partner of $\pi_i^s$ $k_0 := \begin{cases} k \leftarrow \mathcal{K} & \text{if } j \leq \eta \\ k_i^s & \text{if } j > \eta \end{cases}$					
If $i = \eta$ : $sk_i^s := \hat{sk}$ $r \leftarrow \mathcal{S}\mathcal{K}$ ; $st_i^s \leftarrow \mathcal{E}(s_i, r)$						
If $crp_j = \text{false}$ : Then $\exists$ unique $t$ s.t. $\hat{pk}$ output by $\pi_j^t$ Choose the corresponding $sk$ $(c, K) \leftarrow \text{Encap}^*(sk)$						
If $i = \eta$ : $sk := sk_i^s$ $K \leftarrow \text{Decap}(\hat{sk}, c)$ $st_i^s := \emptyset$ ; $k_i^s := K$ ; $\Psi_i^s := \text{accept}$ $\text{msg}' := \emptyset$						

**Fig. 7.** Games  $G_3$ - $G_5$  for the proof of Theorem 1. Queries to  $\mathcal{O}_{AKE}$  where  $\text{query} \in \{\text{Corrupt}, \text{RegisterCorrupt}, \text{SessionKeyReveal}, \text{StateReveal}\}$  are defined as in the original game in Figure 5.

original key between  $\pi_i^s$  and  $\pi_j^t$ , so  $\mathcal{B}_{\text{KEM}}$  can further check  $\text{Partner}(\pi_i^s \rightarrow \pi_j^t)$  by simply testing whether  $k_j^t = k_i^s$ . This simulation is perfect as well.

Finally,  $\mathcal{A}$  outputs  $b^*$ . If  $b = b^*$ ,  $\mathcal{B}_{\text{KEM}}$  outputs 1. Otherwise, it outputs 0.

We want to elaborate in more detail on why a tuple  $(pk_n, sk_n, c_n, K_n)$  is used at most once. As  $\text{NoMsgCon}$  does not happen, there exists a partner for each oracle that has accepted when the intended partner was not corrupted. Thus, for each query  $\text{Send}(i, s, j, (pk, \sigma_1))$ , where  $P_j$  is uncorrupted, there exists a partner oracle  $\pi_j^t$  that has sent  $(pk, \cdot)$ . Finally, as  $\text{Repeat}$  does not happen,  $\mathcal{A}$  cannot replay  $(pk, \sigma_1)$  to another oracle  $\pi_{i'}^s$  because  $\sigma_1$  includes the identities and the nonce  $N$ .

If  $\mathcal{B}_{\text{KEM}}$ 's input  $(c_n, K_n)$  is computed using the original  $\text{Encap}$  algorithm, then  $\mathcal{B}_{\text{KEM}}$  perfectly simulates  $\mathbf{G}_3$ . Otherwise, if  $(c_n, K_n)$  is computed using the  $\text{Encap}^*$  algorithm, then  $\mathcal{B}_{\text{KEM}}$  perfectly simulates  $\mathbf{G}_4$ . Hence,

$$|\Pr[\text{Win}_3] - \Pr[\text{Win}_4]| \leq \text{Adv}_{\text{KEM, Encap}^*, \mu\ell}^{\text{mu-sim}}(\mathcal{B}_{\text{KEM}}).$$

**Game  $\mathbf{G}_{4,\eta,0}$** ,  $\eta \in \{1, \dots, \mu + 1\}$ : From  $\mathbf{G}_{4,1,0}$  to  $\mathbf{G}_{4,\mu+1,0}$ , we will use hybrid arguments to replace the test keys  $k_0$  with random keys for all oracles  $\pi_i^s$ . Here, we have to take into account whether the oracle sent the first message or whether it received the first message. We will consider one user after another and in each step, we will replace the session key in test queries where that user's oracle has received the first message. At the same time, we replace the session keys in test queries where that user's oracle is a partner oracle that has received the first message. In particular, in game  $\mathbf{G}_{4,\eta,0}$ , when  $\mathcal{A}$  queries  $\text{Test}(i, s)$ , instead of setting  $k_0$  to the real session key  $k_i^s$ , we choose a random key if

- (1)  $\pi_i^s$  has received the first message and  $i < \eta$ , or
- (2)  $\pi_i^s$  has sent the first message,  $\pi_j^t$  is the partner oracle and  $j < \eta$ .

Clearly,  $\mathbf{G}_{4,1,0}$  is identical to  $\mathbf{G}_4$  and  $\mathbf{G}_{4,\mu+1,0}$  is identical to  $\mathbf{G}_5$ .

**Lemma 2.** *Let  $\text{Encap}^*$  be the additional algorithm associated to a KEM and let the key encapsulated by  $\text{Encap}^*$  be  $\epsilon$ -uniform. Then for  $\eta \in \{1, \dots, \mu\}$ ,*

$$|\Pr[\text{Win}_{4,\eta,0}] - \Pr[\text{Win}_{4,\eta+1,0}]| \leq 2 \cdot \text{Adv}_{\text{SE}, \ell}^{\text{mrpa}}(\mathcal{B}_{\text{SE}}) + 2\ell \cdot \epsilon.$$

*Proof.* We will consider two cases: (1) The adversary corrupts  $P_\eta$  and (2) the adversary does not corrupt  $P_\eta$ . We have

$$\begin{aligned} |\Pr[\text{Win}_{4,\eta,0}] - \Pr[\text{Win}_{4,\eta+1,0}]| &\leq |\Pr[\text{Win}_{4,\eta,0} \wedge \text{crp}_\eta] - \Pr[\text{Win}_{4,\eta+1,0} \wedge \text{crp}_\eta]| \\ &\quad + |\Pr[\text{Win}_{4,\eta,0} \wedge \neg \text{crp}_\eta] - \Pr[\text{Win}_{4,\eta+1,0} \wedge \neg \text{crp}_\eta]|. \end{aligned}$$

First, we consider the case that  $P_\eta$  is corrupted. Let  $\pi_\eta^s$  be any oracle of  $P_\eta$ . If  $\mathcal{A}$  does not issue a test query on any  $\pi_\eta^s$  directly or where  $\pi_\eta^s$  is the partner, then  $\Pr[\text{Win}_{4,\eta,0} \wedge \text{crp}_\eta] = \Pr[\text{Win}_{4,\eta+1,0} \wedge \text{crp}_\eta]$ .

Otherwise, we have to consider the following two cases.

**Case 1:**  $\mathcal{A}$  asks  $\text{Test}(i, s)$  for  $i = \eta$  and  $\pi_\eta^s$  received the first message ( $\text{FirstAcc}_\eta^s = \text{false}$ ). We know that the partner oracle  $\pi_j^t$  received the ephemeral public key  $pk$  output by  $\pi_\eta^s$  and has sent a ciphertext  $c$  computed by  $\text{Encap}^*$ . Then, as  $P_\eta$  is corrupted and  $\mathcal{A}$  queries  $\text{Test}(\eta, s)$ ,  $\mathcal{A}$  is disallowed to ask  $\text{StateReveal}(\eta, s)$  (**TA6**). So the information of the ephemeral secret key  $sk$  leaked to  $\mathcal{A}$  is limited in  $pk$ . By the  $\epsilon$ -uniformity of  $\text{Encap}^*$ , we can replace the corresponding session key in  $\text{Test}(\eta, s)$  with a random key. Note that in this case, the  $\pi_j^t$  is message-consistent with  $\pi_\eta^s$  (i.e.,  $\text{Partner}(\pi_\eta^s \leftrightarrow \pi_j^t)$ ) and  $\mathcal{A}$  can neither test nor reveal the key of  $\pi_j^t$  (**TA4**, **TA5**).

**Case 2:**  $\mathcal{A}$  asks  $\text{Test}(i, s)$ , where  $\pi_i^s$  sent the first message ( $\text{FirstAcc}_i^s = \text{true}$ ) and is partnered to  $\pi_\eta^t$ . We know that the ephemeral public key  $pk$  received by  $\pi_i^s$  was sent by  $\pi_\eta^t$  as  $P_\eta$  has to be uncorrupted when  $\pi_i^s$  accepts. The ciphertext  $c$  sent by  $\pi_i^s$  was computed by  $\text{Encap}^*$ . As we consider that  $P_\eta$  is corrupted later and  $\mathcal{A}$  queries  $\text{Test}(i, s)$ ,  $\mathcal{A}$  is disallowed to ask  $\text{StateReveal}(\eta, t)$  (**TA7**).

However,  $P_i$  may be corrupted and  $\mathcal{A}$  can create a new ciphertext  $c' \neq c$ , sent it to  $\pi_\eta^t$ . In this case  $\pi_\eta^t$  is not message-consistent with  $\pi_i^s$ . If  $\mathcal{A}$  reveals the session key of  $\pi_\eta^t$ , it will get  $\text{Decap}(\hat{sk}, c')$ . Overall, the information of the ephemeral secret key  $sk$  leaked to  $\mathcal{A}$  is limited in  $pk$  and  $\text{Decap}(\hat{sk}, c')$ . By  $\epsilon$ -uniformity of  $\text{Encap}^*$ , we can still replace the session key of  $\pi_i^s$  with a random key.

As one party has at most  $\ell$  sessions, union bound yields

$$|\Pr[\text{Win}_{4,\eta,0} \wedge \text{crp}_\eta] - \Pr[\text{Win}_{4,\eta+1,0} \wedge \text{crp}_\eta]| \leq \ell \cdot \epsilon.$$

Now we will look at the case that  $P_\eta$  is not corrupted and we introduce two further intermediate games  $\mathsf{G}_{4,\eta,1}$  and  $\mathsf{G}_{4,\eta,2}$ .

**Game  $\mathsf{G}_{4,\eta,1}$ ,  $\eta \in \{1, \dots, \mu\}$ :** On a query  $\text{Send}(i, s, j, N)$ , where  $i = \eta$ , we do not encrypt the ephemeral secret key  $\hat{sk}$  in the state, but a random secret key  $r \leftarrow_{\$} \hat{\mathcal{SK}}$ . We store the real secret key in an additional variable  $\text{sk}_\eta^s$  such that we can later access it for decapsulation. The rest remains unchanged.

We now construct an IND-mRPA adversary  $\mathcal{B}_{\text{SE},\eta}$  against the symmetric encryption scheme SE with message space  $\hat{\mathcal{SK}}$ .  $\mathcal{B}_{\text{SE},\eta}$  inputs  $\ell$  message-ciphertext pairs  $\{(\hat{sk}_n, c_n)\}_{n \in [\ell]}$  for  $\ell$  random messages  $\hat{sk}_n \leftarrow_{\$} \hat{\mathcal{SK}}$ , where  $c_n$  is either an encryption of  $\hat{sk}_n$  or that of a random message  $r_n \leftarrow_{\$} \hat{\mathcal{SK}}$ .  $\mathcal{B}_{\text{SE},\eta}$  generates the public parameter and signature key pairs  $(vk_i, \text{ssk}_i)$  for  $i \in [\mu]$ . For all  $i \neq \eta$ , it also generates symmetric keys  $s_i$ . It sets  $\text{PKList} := \{vk_i\}_{i \in [\mu]}$ , initializes all variables, chooses  $b \leftarrow_{\$} \{0, 1\}$  and then runs  $\mathcal{A}$ . If  $\mathcal{A}$  queries  $\mathcal{O}_{\text{AKE}}$ ,  $\mathcal{B}_{\text{SE},\eta}$  responds as follows:

- $\text{Send}(i, s, j, \text{msg} = N)$ : If  $i = \eta$ ,  $\mathcal{B}_{\text{SE},\eta}$  computes  $\hat{pk} := \text{KEM.PK}(\hat{sk}_s)$  from the  $s$ -th message. It sets  $\text{sk}_\eta^s := \hat{sk}_s$  and the state variable  $\text{st}_\eta^s := c_s$ .
- $\text{Send}(i, s, j, \text{msg} = (c, \sigma_2))$ : If  $i = \eta$ ,  $\mathcal{B}_{\text{SE},\eta}$  chooses  $\hat{sk}$  from  $\text{sk}_\eta^s$  instead of decrypting the state.
- $\text{Corrupt}(i)$ : If  $i = \eta$ ,  $\mathcal{B}_{\text{SE},\eta}$  aborts.
- Queries  $\text{Send}(i, s, j, \top)$ ,  $\text{Send}(i, s, j, (\hat{pk}, \sigma_1))$ ,  $\text{RegisterCorrupt}$ ,  $\text{StateReveal}$ ,  $\text{SessionKeyReveal}$  and  $\text{Test}$  can be simulated as in  $\mathsf{G}_{4,\eta,0}$ .

Finally,  $\mathcal{A}$  outputs  $b^*$ . If  $b = b^*$  and  $\mathcal{B}_{\text{SE},\eta}$  does not abort,  $\mathcal{B}_{\text{SE},\eta}$  outputs 1. Otherwise, it outputs 0. If the input ciphertexts are encryptions of the messages  $sk_1, \dots, sk_\ell$  and  $\mathcal{B}_{\text{SE},\eta}$  does not abort, it perfectly simulates  $\mathsf{G}_{4,\eta,0} \wedge \neg \text{crp}_\eta$ . If the input ciphertexts are encryptions of random messages and  $\mathcal{B}_{\text{SE},\eta}$  does not abort, it perfectly simulates  $\mathsf{G}_{4,\eta,1} \wedge \neg \text{crp}_\eta$ . Thus,

$$|\Pr[\text{Win}_{4,\eta,0} \wedge \neg \text{crp}_\eta] - \Pr[\text{Win}_{4,\eta,1} \wedge \neg \text{crp}_\eta]| \leq \text{Adv}_{\text{SE},\ell}^{\text{mrpa}}(\mathcal{B}_{\text{SE},\eta}).$$

**Game  $\mathsf{G}_{4,\eta,2}$ ,  $\eta \in \{1, \dots, \mu\}$ :** In game  $\mathsf{G}_{4,\eta,2}$ , we switch all session keys output by  $\text{Test}$  where oracle  $\pi_\eta^s$  received the first message to random. Also, we switch all session keys output by  $\text{Test}$  where oracle  $\pi_\eta^s$  is the partner that has sent the first message to random. In particular, when  $\mathcal{A}$  queries  $\text{Test}(i, s)$ , instead of setting  $k_0$  to the real session key  $k_i^s$ , we choose a random key if

- (1)  $\pi_i^s$  has received the first message and  $i \leq \eta$ , or
- (2)  $\pi_i^s$  has sent the first message,  $\pi_j^t$  is the partner oracle and  $j \leq \eta$ .

Similar to the case where  $P_\eta$  is corrupted, we will argue that the difference between the two games is bounded by the  $\epsilon$ -uniformity of  $\text{Encap}^*$ . Again, we consider the two cases where we deviate from the previous game.

**Case 1:**  $\mathcal{A}$  asks  $\text{Test}(i, s)$  for  $i = \eta$  and  $\pi_\eta^s$  received the first message ( $\text{FirstAcc}_\eta^s = \text{false}$ ). We know that the partner oracle  $\pi_j^t$  received the ephemeral public key  $\hat{pk}$  output by  $\pi_\eta^s$  and has sent a ciphertext  $c$  computed by  $\text{Encap}^*$ .  $\mathcal{A}$  may query  $\text{StateReveal}(\eta, s)$ , but will receive only an encryption of a random secret key. So the information of the ephemeral secret key  $\hat{sk}$  leaked to  $\mathcal{A}$  is limited in  $\hat{pk}$ . If  $\mathcal{A}$  queries  $\text{Test}(\eta, s)$ ,  $\mathcal{A}$  can neither test nor reveal the key of  $\pi_j^t$  as  $\pi_j^t$  is also partnered to  $\pi_\eta^s$ . Due to  $\epsilon$ -uniformity of  $\text{Encap}^*$ , we can replace the session key of  $\pi_\eta^s$  with a random key.

**Case 2:**  $\mathcal{A}$  asks  $\text{Test}(i, s)$ , where  $\pi_i^s$  sent the first message ( $\text{FirstAcc}_i^s = \text{true}$ ) and is partnered to  $\pi_\eta^t$ . We know that the ephemeral public key  $\hat{pk}$  received by  $\pi_i^s$  was sent by the partner oracle  $\pi_\eta^t$  as  $P_\eta$  is uncorrupted. The ciphertext  $c$  sent by  $\pi_i^s$  was computed by  $\text{Encap}^*$ .  $\mathcal{A}$  may reveal the state of  $\pi_\eta^t$ , but will receive only an encryption of a random secret key.  $\mathcal{A}$  may corrupt  $P_i$ , create a new ciphertext  $c' \neq c$  and sent it to  $\pi_\eta^t$ . In this case, if  $\pi_\eta^t$  is not message-consistent with  $\pi_i^s$ ,  $\mathcal{A}$  can reveal the session key of  $\pi_\eta^t$  and receives  $\text{Decap}(\hat{sk}, c')$ . Overall, the information of the ephemeral secret key  $\hat{sk}$  leaked to  $\mathcal{A}$  is limited in  $\hat{pk}$  and  $\text{Decap}(\hat{sk}, c')$ . Thus, we can still replace the session key of  $\pi_i^s$  with a random key due to  $\epsilon$ -uniformity of  $\text{Encap}^*$ .

As there are at most  $\ell$  test sessions for one party, we have

$$|\Pr[\text{Win}_{4,\eta,1} \wedge \neg \text{crp}_\eta] - \Pr[\text{Win}_{4,\eta,2} \wedge \neg \text{crp}_\eta]| \leq \ell \cdot \epsilon .$$

Now, we can switch back the encryption of a random ephemeral secret key to the real ephemeral key. Note that this is  $G_{4,\eta+1,0}$ . We can construct an IND-mRPA adversary  $\mathcal{B}'_{\text{SE},\eta}$  against SE such that

$$|\Pr[\text{Win}_{4,\eta,2} \wedge \neg \text{crp}_\eta] - \Pr[\text{Win}_{4,\eta+1,0} \wedge \neg \text{crp}_\eta]| \leq \text{Adv}_{\text{SE},\ell}^{\text{mrpa}}(\mathcal{B}'_{\text{SE},\eta}) ,$$

where  $\mathcal{B}'_{\text{SE},\eta}$  deviates from  $\mathcal{B}_{\text{SE},\eta}$  only in the simulation of the Test oracle as introduced in game  $G_{4,\eta,2}$ .

Lemma 2 now follows from collecting the probabilities and folding adversaries  $\mathcal{B}_{\text{SE},\eta}$  and  $\mathcal{B}'_{\text{SE},\eta}$  into a single adversary  $\mathcal{B}_{\text{SE}}$ .  $\square$

**Game  $G_5$ :** Finally, game  $G_5$  is identical to  $G_{4,\mu+1,0}$ . In this game, the Test oracle always outputs a random key, independent of the bit  $b$ . Hence,

$$\Pr[\text{Win}_5] = \frac{1}{2} ,$$

which concludes the proof of Theorem 1.  $\square$

**Theorem 2 (Security of  $\text{AKE}_{3\text{msg}}$  with Replay Attacks and without State Reveals).**

For any adversary  $\mathcal{A}$  against  $\text{AKE}_{3\text{msg}}$  with replay attacks and without state reveals, there exist an MU-EUF-CMA<sup>corr</sup> adversary  $\mathcal{B}_{\text{SIG}}$  against SIG and an MUSC-otCCA adversary  $\mathcal{B}_{\text{KEM}}$  against KEM such that

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

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

We first give a proof sketch, then present the formal proof of Theorem 2.

*Proof Sketch.* This proof is very similar to the proof of Theorem 1. The signatures and nonce in the protocol ensure that the adversary can only forward messages for test sessions and cannot replay a particular ephemeral public key  $\hat{pk}$ .

As we do not consider state reveals, the reduction does not have to know the corresponding secret key  $\hat{sk}$ . Instead, we can use a weaker security notion for KEM, which allows for one challenge query and one decapsulation query for each  $\hat{pk}$ . Also, there is no need for a hybrid argument and we can output a random key for sessions which will be tested as well as for sessions that will be revealed, using MUSC-otCCA security of KEM.

**Proof of Theorem 2.** The proof is very similar to that of Theorem 1. We consider a sequence of games  $G_0$ - $G_3$ , which are the same as in the proof of Theorem 1, only that we start with  $G_0$  as the  $\text{Exp}_{\text{AKE}_{3\text{msg}},\mu,\ell,\mathcal{A}}^{\text{replay}}$  experiment. Note that the game changes from  $G_0$ - $G_3$  in Theorem 1 do not involve state reveals. Thus by a similar analysis, we establish

$$|\Pr[\text{Exp}_{\text{AKE}_{3\text{msg}},\mu,\ell,\mathcal{A}}^{\text{replay}} \Rightarrow 1] - \Pr[\text{Win}_3]| \leq \text{Adv}_{\text{SIG},\mu}^{\text{mu-corr}}(\mathcal{B}_{\text{SIG}}) + (\mu\ell)^2 \cdot 2^{-\gamma} + \mu\ell^2 \cdot 2^{-\lambda} ,$$

and

$$\Pr[\text{Win}_{\text{Auth}}] \leq 2 \cdot \text{Adv}_{\text{SIG},\mu}^{\text{mu-corr}}(\mathcal{B}_{\text{SIG}}) + 2(\mu\ell)^2 \cdot 2^{-\gamma} + \mu\ell^2 \cdot 2^{-\lambda} .$$

Recall that in  $G_3$ , the game defines partnering using message-consistency for all oracles  $\pi_i^s$  that have accepted with  $\text{Aflag}_i^s = \text{false}$ . In order to bound  $\Pr[\text{Win}_3]$ , we construct an adversary  $\mathcal{B}_{\text{KEM}}$  against MUSC-otCCA security of KEM (see Figure 8). We will show that

$$|\Pr[\text{Win}_3] - \frac{1}{2}| \leq 2 \cdot \text{Adv}_{\text{KEM},\mu\ell}^{\text{musc-otcca}}(\mathcal{B}_{\text{KEM}}) .$$

The idea is that we do not only replace the session key of an oracle when it is tested, but we replace the session keys of all oracles that are possibly tested. In particular, these are sessions

where the intended partner is uncorrupted when the oracle accepts. These oracles can then either be tested or revealed. However, as we do not consider state reveals, the adversary will never see the ephemeral secret key and thus we can also output a random key for a `SessionKeyReveal` query.

Let  $\beta$  be the random bit of  $\mathcal{B}_{\text{KEM}}$ 's challenger.  $\mathcal{B}_{\text{KEM}}$  inputs the public parameter  $\text{pp}_{\text{KEM}}$  and  $\{pk_n\}_{n \in [\mu\ell]}$ .  $\mathcal{B}_{\text{KEM}}$  generates the public parameter for SIG and signature key pairs  $(vk_i, ssk_i)$  for  $i \in [\mu]$  and sets  $\text{PKList} := \{vk_i\}_{i \in [\mu]}$ . It initializes all variables, chooses a random challenge bit  $b \leftarrow_s \{0, 1\}$  and runs  $\mathcal{A}$ . If  $\mathcal{A}$  makes a query to  $\mathcal{O}_{\text{AKE}}$ ,  $\mathcal{B}_{\text{KEM}}$  simulates the response as follows:

- `Send`( $i, s, j, \text{msg} = N$ ):  $\mathcal{B}_{\text{KEM}}$  uses the public key with index  $(i-1)\mu + s$  as ephemeral public key, i.e.  $\hat{pk} := pk_{(i-1)\mu + s}$ .
- `Send`( $i, s, j, \text{msg} = (pk, \sigma_1)$ ): If  $P_j$  is uncorrupted, then due to the fact that `NoMsgCon` does not happen, there exists a unique oracle  $\pi_j^t$  such that  $\hat{pk}$  was output by  $\pi_j^t$ . Furthermore,  $n = (j-1)\mu + t$  is the index of that public key. Then  $\mathcal{B}_{\text{KEM}}$  queries  $\mathcal{O}_{\text{ENCAP}}^\beta(n)$ , receives a ciphertext and key  $(c, K_\beta)$  and sets  $k_i^s := K_\beta$ . If  $P_j$  is corrupted,  $\mathcal{B}_{\text{KEM}}$  runs `Encap`( $\hat{pk}$ ) itself to compute  $(c, K)$ . It also computes a signature  $\sigma_2$  as the protocol specifies and outputs  $(c, \sigma_2)$ .
- `Send`( $i, s, j, \text{msg} = (c, \sigma_2)$ ): Let  $n = (i-1)\mu + s$ . Then,  $\pi_i^s$  sent  $\hat{pk}_n$ . If there exists an oracle  $\pi_j^t$  that has received  $pk_n$  and has sent  $c$ , then  $\mathcal{B}_{\text{KEM}}$  sets  $k_i^s := k_j^t$ . Otherwise,  $\mathcal{B}_{\text{KEM}}$  queries  $\mathcal{O}_{\text{DECAP}}(n, c)$ , receives  $K$  and sets  $k_i^s := K$ .
- `Test`( $i, s$ ): After ruling out trivial attacks **TA1**, **TA2** and **TA3**,  $\mathcal{B}_{\text{KEM}}$  checks for trivial attacks **TA4** and **TA5** using message-consistency check `MsgCon`( $\pi_i^s \leftarrow \pi_j^t$ ) and tests if  $k_j^t = k_i^s$ . If it does not output  $\perp$ ,  $\mathcal{B}_{\text{KEM}}$  sets  $k_0 = k_i^s$  and  $k_1 \leftarrow_s \mathcal{K}$  and outputs  $k_b$ .
- `SessionKeyReveal`( $i, s$ ): After ruling out trivial attack **TA2**,  $\mathcal{B}_{\text{KEM}}$  checks for trivial attack **TA4** by checking if there exists an oracle  $\pi_j^t$  such that  $\pi_j^t$  is tested and `MsgCon`( $\pi_j^t \leftarrow \pi_i^s$ ). If further  $k_j^t = k_i^s$ ,  $\mathcal{B}_{\text{KEM}}$  returns  $\perp$ . Otherwise, it outputs  $k_i^s$ .
- Queries `Send`( $i, s, j, \top$ ), `Corrupt` and `RegisterCorrupt` can be simulated as in  $\mathsf{G}_3$ .

Finally,  $\mathcal{A}$  outputs  $b^*$  and  $\mathcal{B}_{\text{KEM}}$  outputs  $\beta^* := 0$  if  $b^* = b$  and  $\beta^* := 1$  otherwise.

As events `NoMsgCon` and `Repeat` do not happen,  $\mathcal{B}_{\text{KEM}}$  queries  $\mathcal{O}_{\text{ENCAP}}^\beta$  only once for each  $\hat{pk}$ , equivalently to the proof of Theorem 1. Also,  $\mathcal{O}_{\text{DECAP}}$  is queried at most once, as each ephemeral public key is only output by one oracle which accepts the session key after calling  $\mathcal{O}_{\text{DECAP}}$ . In the following, we will argue that  $\mathcal{B}_{\text{KEM}}$  perfectly simulates  $\mathsf{G}_3$  if  $\beta = 0$ , and that  $\mathcal{A}$ 's view is independent of  $b$  if  $\beta = 1$ .

**Case  $\beta = 0$ :** For each query to  $\mathcal{O}_{\text{ENCAP}}^0$ ,  $\mathcal{B}_{\text{KEM}}$  receives the real key. When  $\mathcal{A}$  queries `Test`( $i, s$ ),  $\mathcal{B}_{\text{KEM}}$  needs to check partnering to avoid **TA4** and **TA5**. We know that  $\text{Aflag}_i^s = \text{false}$  because otherwise  $\mathcal{B}_{\text{KEM}}$  would have returned  $\perp$ . Thus,  $\mathcal{B}_{\text{KEM}}$  checks `MsgCon`( $\pi_i^s \leftarrow \pi_j^t$ ) as in  $\mathsf{G}_3$ , but instead of checking `Partner`( $\pi_j^t \leftarrow \pi_i^s$ ), it checks whether  $k_j^t = k_i^s$ . As  $\beta = 0$ ,  $k_i^s$  is the real session key and original key between  $\pi_i^s$  and  $\pi_j^t$ . Thus, `Partner`( $\pi_j^t \leftarrow \pi_i^s$ ) can be efficiently checked by testing if  $k_j^t = k_i^s$ .  $\mathcal{B}_{\text{KEM}}$  simulates `Test` queries as in  $\mathsf{G}_3$ . When  $\mathcal{A}$  queries `SessionKeyReveal`( $i, s$ ),  $\mathcal{B}_{\text{KEM}}$  also needs to check partnering to avoid **TA4**. Therefore, for each oracle  $\pi_j^t$  that is tested and thus  $\text{Aflag}_j^t = \text{false}$ ,  $\mathcal{B}_{\text{KEM}}$  checks if  $\pi_j^t$  is partnered to  $\pi_i^s$  by `MsgCon`( $\pi_j^t \leftarrow \pi_i^s$ ) as in  $\mathsf{G}_3$ . Instead of checking `Partner`( $\pi_i^s \leftarrow \pi_j^t$ ), it checks whether  $k_j^t = k_i^s$ . We know that  $k_j^t$  is the real session key, thus  $\mathcal{B}_{\text{KEM}}$  simulates `SessionKeyReveal` queries as in  $\mathsf{G}_3$ . Consequently,  $\mathcal{B}_{\text{KEM}}$  simulates  $\mathsf{G}_3$  perfectly for  $\mathcal{A}$  in this case, and  $\Pr[\beta^* = \beta \mid \beta = 0] = \Pr[\text{Win}_3]$ .

**Case  $\beta = 1$ :** For each query to  $\mathcal{O}_{\text{ENCAP}}^1$ ,  $\mathcal{B}_{\text{KEM}}$  receives a random key. When  $\mathcal{A}$  queries `Test`( $i, s$ ),  $\mathcal{B}_{\text{KEM}}$  checks `MsgCon`( $\pi_i^s \leftarrow \pi_j^t$ ) and if  $k_j^t = k_i^s$ . As  $\beta = 1$ ,  $k_0 = k_i^s$  and  $k_1$  are both random keys. Thus,  $\mathcal{B}_{\text{KEM}}$ 's output is independent of  $b$ . When  $\mathcal{A}$  queries `SessionKeyReveal`( $i, s$ ),  $\mathcal{B}_{\text{KEM}}$  checks for each oracle  $\pi_j^t$  that is tested and thus  $\text{Aflag}_j^t = \text{false}$ , if  $\pi_j^t$  is partnered to  $\pi_i^s$  by `MsgCon`( $\pi_j^t \leftarrow \pi_i^s$ ). If  $k_j^t \neq k_i^s$ ,  $\mathcal{B}_{\text{KEM}}$  outputs  $k_i^s$ . As  $k_j^t$  is a random key,  $\mathcal{A}$  learns nothing about the bit  $b$ . We have  $\Pr[b^* = b] = \frac{1}{2}$  in this case and further  $\Pr[\beta^* = \beta \mid \beta = 1] = \frac{1}{2}$ .

It follows that

$$\begin{aligned} \text{Adv}_{\text{KEM}, \mu\ell}^{\text{musc-otcca}}(\mathcal{B}_{\text{KEM}}) &= \left| \Pr[\beta^* = \beta] - \frac{1}{2} \right| \\ &= \left| \frac{1}{2} \cdot \Pr[\beta^* = \beta \mid \beta = 0] + \frac{1}{2} \cdot \Pr[\beta^* = \beta \mid \beta = 1] - \frac{1}{2} \right| \\ &= \left| \frac{1}{2} \cdot \Pr[\text{Win}_3] + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} \right| = \frac{1}{2} \left| \Pr[\text{Win}_3] - \frac{1}{2} \right|. \end{aligned}$$

Collecting the probabilities yields the bound in Theorem 2.  $\square$

<p><math>\mathcal{B}_{\text{KEM}}^{\mathcal{O}_{\text{ENCAP}}^{\beta}(\cdot), \mathcal{O}_{\text{DECAP}}(\cdot, \cdot)}(\text{pp}_{\text{KEM}}, pk_1, \dots, pk_{\mu\ell}) :</math></p> <p><math>\text{pp}_{\text{SIG}} \leftarrow \text{SIG.Setup}</math>  For <math>i \in [\mu]</math>:      <math>(vk_i, ssk_i) \leftarrow \text{SIG.Gen}(\text{pp}_{\text{SIG}});</math>      <math>crp_i := \text{false}</math>  PKList := <math>\{vk_i\}_{i \in [\mu]}</math>; <math>b \leftarrow \{0, 1\}</math>  For <math>(i, s) \in [\mu] \times [\ell]</math>:      <math>\text{var}_i^s := (\text{Pid}_i^s, k_i^s, \Psi_i^s) := (\emptyset, \emptyset, \emptyset)</math>      <math>(\text{Sent}_i^s, \text{Recv}_i^s) := (\emptyset, \emptyset)</math>      Aflag<math>_i^s := \text{false}</math>; T<math>_i^s := \text{false}</math>; kRev<math>_i^s := \text{false}</math>  Repeat := <b>false</b>; NoMsgCon := <b>false</b>  <math>b^* \leftarrow \mathcal{A}^{\mathcal{O}_{\text{AKE}}(\cdot)}(\text{pp}_{\text{AKE}}, \text{PKList})</math>  If <math>b^* = b</math>: Return <math>\beta^* := 0</math>  Else: Return <math>\beta^* := 1</math></p> <p>// During the execution <math>\mathcal{B}_{\text{KEM}}</math> checks if one of the following  // flags is set to true and if so, it aborts immediately:  Repeat := <b>true</b>, If <math>\exists i, s, s' \in [\mu] \times [\ell]^2, N \in \{0, 1\}^\lambda</math> s.t.      <math>N \in \text{Sent}_i^s \wedge N \in \text{Sent}_{i'}^{s'}</math>  NoMsgCon := <b>true</b>, If <math>\exists i, s \in [\mu] \times [\ell]</math> s.t. (1') <math>\wedge</math> (2') <math>\wedge</math> (3').      Let <math>j := \text{Pid}_i^s</math>.      (1') <math>\Psi_i^s = \text{accept}</math>      (2') Aflag<math>_i^s = \text{false}</math>      (3') <math>\nexists t \in [\ell]</math> s.t. <math>\text{MsgCon}(\pi_i^s \leftarrow \pi_j^t)</math></p> <p><math>\mathcal{O}_{\text{AKE}}(\text{query}) :</math>  If query = <b>Test</b>(<math>i, s</math>):      If <math>\Psi_i^s \neq \text{accept} \vee \text{Aflag}_i^s = \text{true} \vee k\text{Rev}_i^s = \text{true}</math>      <math>\vee T_i^s = \text{true}</math>:          Return <math>\perp</math>      Let <math>j := \text{Pid}_i^s</math>      If <math>\exists t \in [\ell]</math> s.t. <math>\text{MsgCon}(\pi_i^s \leftarrow \pi_j^t) \wedge k_j^t = k_i^s</math>:          If <math>k\text{Rev}_j^t = \text{true} \vee T_j^t = \text{true}</math>: Return <math>\perp</math>      T<math>_i^s := \text{true}</math>      <math>k_0 := k_i^s</math>; <math>k_1 \leftarrow \mathcal{K}</math>      Return <math>k_b</math></p> <p>If query = <b>SessionKeyReveal</b>(<math>i, s</math>):      If <math>\Psi_i^s \neq \text{accept}</math>: Return <math>\perp</math>      If T<math>_i^s = \text{true}</math>: Return <math>\perp</math>      Let <math>j := \text{Pid}_i^s</math>      If <math>\exists t \in [\ell]</math> s.t. T<math>_j^t = \text{true}</math>:          If <math>\text{MsgCon}(\pi_j^t \leftarrow \pi_i^s) \wedge k_j^t = k_i^s</math>: Return <math>\perp</math>      kRev<math>_i^s := \text{true}</math>; Return <math>k_i^s</math></p>	<p><math>\mathcal{O}_{\text{AKE}}(\text{query}) :</math>  If query = <b>Send</b>(<math>i, s, j, \text{msg}</math>):      If <math>\Psi_i^s = \text{accept}</math>: Return <math>\perp</math>      If <math>\Psi_i^s = \top</math>: //session is initiated          Pid<math>_i^s := j</math>          <math>N \leftarrow \{0, 1\}^\lambda</math>          msg' := <math>N</math>      If msg = <math>N</math>: //first message          Pid<math>_i^s := j</math>          Let <math>n := (i - 1)\mu + s</math>; <math>\hat{pk} := pk_n</math>          <math>\sigma_1 \leftarrow \text{Sign}(ssk_i, (P_i, P_j, \hat{pk}, N))</math>          msg' := <math>(\hat{pk}, \sigma_1)</math>      If msg = <math>(pk, \sigma_1)</math>: //second message          Choose <math>N \in \text{Sent}_i^s</math>          If Pid<math>_i^s \neq j</math> or <math>\text{Ver}(vk_j, (P_j, P_i, \hat{pk}, N), \sigma_1) \neq 1</math>:              <math>\Psi_i^s := \text{reject}</math>; Return <math>\perp</math>          If <math>crp_j = \text{false}</math>:              Then <math>\exists</math> unique <math>t</math> s.t. <math>\hat{pk}</math> output by <math>\pi_j^t</math>              Let <math>n := (j - 1)\mu + t</math>              <math>(c, K_\beta) \leftarrow \mathcal{O}_{\text{ENCAP}}^\beta(n)</math>; <math>k_i^s := K_\beta</math>          Else:              <math>(c, K) \leftarrow \text{Encap}(\hat{pk})</math>; <math>k_i^s := K</math>          <math>\sigma_2 \leftarrow \text{Sign}(ssk_i, (P_j, P_i, \hat{pk}, \sigma_1, c, N))</math>          <math>\Psi_i^s := \text{accept}</math>          msg' := <math>(c, \sigma_2)</math>      If msg = <math>(c, \sigma_2)</math>: //third message          Choose <math>N \in \text{Recv}_i^s</math> and <math>(\hat{pk}, \sigma_1) \in \text{Sent}_i^s</math>          If Pid <math>\neq j</math> or <math>\text{Ver}(vk_j, (P_i, P_j, \hat{pk}, \sigma_1, c, N), \sigma_2) \neq 1</math>:              <math>\Psi_i^s := \text{reject}</math>; Return <math>\perp</math>          Let <math>n := (i - 1)\mu + s</math> and <math>j := \text{Pid}_i^s</math>          If <math>\exists t</math> s. t. <math>\text{Recv}_j^t = \{(pk, \cdot)\} \wedge \text{Sent}_j^t = \{N, (c, \cdot)\}</math>:              <math>k_i^s := k_j^t</math>          Else:              <math>K \leftarrow \mathcal{O}_{\text{DECAP}}(n, c)</math>; <math>k_i^s := K</math>          <math>\Psi_i^s := \text{accept}</math>          msg' := <math>\emptyset</math>      Recv<math>_i^s := \text{Recv}_i^s \cup \{\text{msg}\}</math>; Sent<math>_i^s := \text{Sent}_i^s \cup \{\text{msg}'\}</math>      If <math>\Psi_i^s = \text{accept}</math>:          If <math>crp_j = \text{true}</math>: Aflag<math>_i^s := \text{true}</math>      Return msg'</p>
---	--

**Fig. 8.** Adversary  $\mathcal{B}_{\text{KEM}}$  against MUSC-otCCA security of KEM for the proof of Theorem 2. Queries to  $\mathcal{O}_{\text{AKE}}$  where query  $\in \{\text{Corrupt}, \text{RegisterCorrupt}\}$  are defined as in the original game  $\text{Exp}_{\text{AKE}_{3\text{msg}}, \mu, \ell, \mathcal{A}}^{\text{replay}}$  in Figure 5.

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

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

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

*Proof Sketch.* In the two-message protocol, the signatures ensure that the adversary can only forward messages for test sessions. However, the adversary may also replay a message containing a particular ephemeral public key  $\hat{pk}$  in another session. Thus, we require multi-challenge security of the KEM. We still only need one decapsulation query as the session is closed after it receives the last message and has accepted or rejected the session key.

As we do not consider state reveals and the adversary will not see any ephemeral secret key  $\hat{sk}$ , we can follow the strategy of the proof of Theorem 2. Thus, we do not only replace the session keys of test sessions, but also of those that will be revealed, using MUC-otCCA security of KEM. The full proof is given in Appendix A.

## 6 Signatures with Tight Adaptive Corruptions

### 6.1 Pairing Groups and MDDH Assumptions

Let  $\text{GGen}$  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  $\lambda$ -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 [17]. 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  $\mathbb{G}_s$ .  $\text{Span}(\mathbf{A}) := \{\mathbf{A}\mathbf{r} \mid \mathbf{r} \in \mathbb{Z}_q^m\} \subset \mathbb{Z}_q^n$  denotes the linear span of  $\mathbf{A}$ , and similarly  $\text{Span}([\mathbf{A}]_s) := \{[\mathbf{A}\mathbf{r}]_s \mid \mathbf{r} \in \mathbb{Z}_q^m\} \subset \mathbb{G}_s^n$ . Note that it is efficient to compute  $[\mathbf{A}\mathbf{B}]_s$  given  $([\mathbf{A}]_s, \mathbf{B})$  or  $(\mathbf{A}, [\mathbf{B}]_s)$  with matching dimensions. We define  $[\mathbf{A}]_1 \circ [\mathbf{B}]_2 := e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{A}\mathbf{B}]_T$ , 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 [17].

**Definition 16 (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  $\eta$  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  $\mathbf{A} \leftarrow_s \mathcal{D}_{\ell, k}$  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  $([\mathbf{A}], [\mathbf{A}\mathbf{w}])$  and  $([\mathbf{A}], [\mathbf{u}])$  where  $\mathbf{A} \leftarrow_s \mathcal{D}_{\ell, k}$ ,  $\mathbf{w} \leftarrow_s \mathbb{Z}_q^k$  and  $\mathbf{u} \leftarrow_s \mathbb{Z}_q^\ell$ .

**Definition 17 ( $\mathcal{D}_{\ell, k}$ -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  $\text{GGen}$  in group  $\mathbb{G}_s$  if for all adversaries  $\mathcal{A}$ , it holds that*

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

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

**Definition 18 (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 3** ( $\mathcal{D}_{\ell,k}$ -MDDH  $\Rightarrow$   $\mathcal{U}_k$ -MDDH [17]). *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*

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

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

The Kernel-Diffie-Hellman assumption ( $\mathcal{D}_k$ -KMDH) [33] is a (weaker) computational analogue of the  $\mathcal{D}_k$ -MDDH Assumption.

**Definition 19** ( $\mathcal{D}_k$ -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  $\mathcal{A}$ ,*

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

where the probabilities are taken over  $\mathcal{P}\mathcal{G} \leftarrow_s \text{GGen}(1^\lambda)$  and  $\mathbf{A} \leftarrow_s \mathcal{D}_k$ .

## 6.2 Previous Schemes with Tight Adaptive Corruptions

We will construct a signature scheme with tight MU-EUF-CMA<sup>corr</sup> security and only small constant number of elements in signatures. Such a scheme has been proposed in [2, Section 2.3] (called SIG<sub>C</sub>), 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 SIG<sub>C</sub> 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<sup>corr</sup> 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 SIG<sub>C</sub> as MAC<sub>BHJKL</sub>: For message space  $\{0, 1\}^\ell$ , it chooses  $\mathbf{A}' \leftarrow_s \mathcal{D}_k$  and random vectors  $\mathbf{x}_{i,j} \leftarrow_s \mathbb{Z}_q^k$  (for  $1 \leq i \leq \ell$  and  $j = 0, 1$ ). Then it defines  $\mathbf{B} := \mathbf{A}' \in \mathbb{Z}_q^{k \times k}$  and publishes system parameters  $\text{pp} := ([\mathbf{B}]_1, ([\mathbf{B}^\top \mathbf{x}_{i,j}]_1)_{1 \leq i \leq \ell, j=0,1})$ . For each user  $n$ , it chooses its MAC secret key as  $[x'_n]_1 \leftarrow_s \mathbb{G}_1$ , and its MAC tag consist of  $([t]_1, [u]_1)$ , where

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

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

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

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

In their final step (between  $H_\ell$  and GAME 4), they argue that the distribution of  $u = \text{RF}_{n,\ell}(\mathbf{m}') + \mathbf{t}^\top \mathbf{x}(\mathbf{m}')$  is uniformly random (as in GAME 4) even for an unbounded adversary, given  $\text{pp}$  and adaptive corruptions. Then they conclude that  $H_\ell$  (where  $u = \text{RF}_{n,\ell}(\mathbf{m}') + \mathbf{t}^\top \mathbf{x}(\mathbf{m}')$ ) 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  $\text{pp}$ ,  $\mathbf{x}_{i,j} \in \mathbb{Z}_q^k$  is uniquely defined. (For concreteness, imagine a simple example where an (unbounded) adversary  $\mathcal{A}$  only queries one MAG tag for message  $\mathbf{m}$  for user  $n$  and then asks for the secret key  $[x'_n]_1 := \text{RF}_{n,\ell}(\mathbf{m})$  of user  $n$ . Then,  $\mathcal{A}$  sees that  $u = \text{RF}_{n,\ell}(\mathbf{m}) + \mathbf{t}^\top \mathbf{x}(\mathbf{m})$  is uniquely defined by  $[x'_n]_1, [\mathbf{t}]_1$  and  $\text{pp}$  in  $H_\ell$ , 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

<sup>12</sup> 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.

the same reason why the BKP MAC cannot be used to construct a tightly secure hierarchical IBE (HIBE) (cf. [28] for more discussion).

To resolve this, we follow the tightly secure HIBE approach in [28] and choose  $\mathbf{B} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{3k \times k}$ . 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  $\mathbf{B}^\top \mathbf{x}_{i,j}$ .

### 6.3 Our Construction

Let  $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$  be a function chosen from a collision-resistant hash function family  $\mathcal{H}$ . Our signature scheme  $\text{SIG}_{\text{MDDH}} := (\text{SIG.Setup}, \text{SIG.Gen}, \text{Sign}, \text{Ver})$  is defined in Figure 9. Correct-

<p><b>SIG.Setup:</b>  <math>\overline{\mathcal{PG}} \leftarrow_{\mathcal{S}} \text{GGen}</math>  <math>\mathbf{A} \leftarrow_{\mathcal{S}} \mathcal{D}_k; \mathbf{B} \leftarrow_{\mathcal{S}} \mathcal{U}_{3k,k}</math>  For <math>1 \leq i \leq \lambda</math> and <math>j = 0, 1</math>:  <math>\mathbf{x}_{i,j} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{3k}; \mathbf{Y}_{i,j} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{3k \times k}</math>  <math>\mathbf{Z}_{i,j} := (\mathbf{Y}_{i,j} \parallel \mathbf{x}_{i,j}) \cdot \mathbf{A} \in \mathbb{Z}_q^{3k \times k}</math>  <math>\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)}</math>  <math>\text{pp} := (\overline{\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})</math>  Return pp</p> <p><b>SIG.Gen(pp)</b>  <math>x' \leftarrow_{\mathcal{S}} \mathbb{Z}_q; \mathbf{y}' \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{1 \times k}</math>  <math>ssk := ([x']_1, [\mathbf{y}']_1)</math>  <math>vk := [\mathbf{z}']_2 := [(\mathbf{y}' \parallel x') \mathbf{A}]_2 \in \mathbb{G}_2^{1 \times k}</math>  Return <math>(vk, ssk)</math></p>	<p><b>Sign(ssk, m):</b>  <math>\mathbf{s} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^k; \mathbf{t} := \mathbf{B}\mathbf{s} \in \mathbb{Z}_q^{3k}</math>  <math>\text{hm} := H(vk, m)</math>  <math>u := x' + \mathbf{s}^\top \mathbf{B}^\top \mathbf{x}(\text{hm}) \in \mathbb{Z}_q</math>  <math>\mathbf{v} := \mathbf{y}' + \mathbf{s}^\top \mathbf{B}^\top \mathbf{Y}(\text{hm}) \in \mathbb{Z}_q^{1 \times k}</math>  Return <math>\sigma := ([t]_1, [u]_1, [\mathbf{v}]_1)</math></p> <p><b>Ver(vk, m, <math>\sigma := ([t]_1, [u]_1, [\mathbf{v}]_1)</math>):</b>  <math>\text{hm} := H(vk, m)</math>  If <math>[\mathbf{v}, u]_1 \circ [\mathbf{A}]_2 = [1]_1 \circ [\mathbf{z}']_2 + [\mathbf{t}^\top]_1 \circ [\mathbf{Z}(\text{hm})]_2</math>:  Return 1  Else: Return 0</p>
---	---

**Fig. 9.** Our signature scheme with tight adaptive corruptions, where for  $\text{hm} \in \{0, 1\}^\lambda$  we define the functions  $\mathbf{x}(\text{hm}) := \sum_{i=1}^\lambda \mathbf{x}_{i,\text{hm}_i}$ ,  $\mathbf{Y}(\text{hm}) := \sum_{i=1}^\lambda \mathbf{Y}_{i,\text{hm}_i}$ ,  $\mathbf{Z}(\text{hm}) := \sum_{i=1}^\lambda \mathbf{Z}_{i,\text{hm}_i}$ , and  $\mathbf{P}(\text{hm}) := \sum_{i=1}^\lambda \mathbf{P}_{i,\text{hm}_i}$ .

ness can be verified as

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

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

**Theorem 4 (Security of  $\text{SIG}_{\text{MDDH}}$ ).** *For any adversary  $\mathcal{A}$  against the MU-EUF-CMA<sup>corr</sup> security of  $\text{SIG}_{\text{MDDH}}$ , 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*

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

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<sup>corr</sup> security of  $\text{SIG}_{\text{MDDH}}$  with a sequence of games given in Figure 10. Let  $\mathcal{A}$  be an adversary against the MU-EUF-CMA<sup>corr</sup> security of  $\text{SIG}_{\text{MDDH}}$ , and let  $\text{Win}_i$  denote the probability that  $\mathcal{G}_i$  returns 1.

**Game  $\mathcal{G}_0$ :**  $\mathcal{G}_0$  is the original experiment  $\text{Exp}_{\text{SIG}, \mu, \mathcal{A}}^{\text{mu-corr}}$  (cf. Definition 3). In addition to the original game, we add a rejection rule if there is a collision between the forgery and a signing query, namely,  $H(vk_{i^*}, m^*) = H(vk_i, m)$  where  $(i, m)$  is one of the signing queries. By the collision resistance of  $H$ , we have

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

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  $([t]_1, [u]_1, [\mathbf{v}]_1)$  and answer

$\mathbb{G}_0, \mathbb{G}_1, \boxed{\mathbb{G}_2}$ ; $\mathcal{PG} \leftarrow \mathcal{GGen}; \mathbf{A} \leftarrow \mathcal{D}_k; \mathbf{B} \leftarrow \mathcal{U}_{3k,k}$ For $1 \leq i \leq \lambda$ and $j = 0, 1$ : $\mathbf{x}_{i,j} \leftarrow \mathbb{Z}_q^{3k}; \mathbf{Y}_{i,j} \leftarrow \mathbb{Z}_q^{3k \times k}$ $\mathbf{Z}_{i,j} := (\mathbf{Y}_{i,j} \parallel \mathbf{x}_{i,j}) \cdot \mathbf{A} \in \mathbb{Z}_q^{3k \times k}$ $\mathbf{P}_{i,j} := \mathbf{B}^\top \cdot (\mathbf{Y}_{i,j} \parallel \mathbf{x}_{i,j}) \in \mathbb{Z}_q^{k \times (k+1)}$ <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <math>\mathbf{Z}_{i,j} \leftarrow \mathbb{Z}_q^{3k \times k}</math>  <math>\mathbf{d}_{i,j} := \mathbf{B}^\top \mathbf{x}_{i,j} \in \mathbb{Z}_q^k</math>  <math>\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}</math>  <math>\mathbf{P}_{i,j} := (\mathbf{E}_{i,j} \parallel \mathbf{d}_{i,j})</math> </div> $\text{pp} := (\mathcal{PG}, [\mathbf{A}]_2, [\mathbf{B}]_1, ([\mathbf{Z}_{i,j}]_2, [\mathbf{P}_{i,j}]_1)_{1 \leq i \leq \lambda, j=0,1})$ For $1 \leq i \leq \mu$ : $x'_i \leftarrow \mathbb{Z}_q; \mathbf{y}'_i \leftarrow \mathbb{Z}_q^{1 \times k}$ $\mathbf{z}'_i := (\mathbf{y}'_i \parallel x'_i) \mathbf{A} \in \mathbb{Z}_q^{1 \times k}$ <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <math>\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}</math> </div> $\text{ssk}_i := ([x'_i]_1, [\mathbf{y}'_i]_1)$ $vk_i := [\mathbf{z}'_i]_2$ $(i^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{SIGN}}(\cdot, \cdot), \mathcal{O}_{\text{CORR}}(\cdot)}(\text{pp}, \{vk_i\}_{1 \leq i \leq \mu})$ If $(i^* \in \mathcal{S}^{\text{corr}}) \vee (m^* \in \mathcal{M}_{i^*}) \vee (\text{Ver}(vk_{i^*}, m^*, \sigma^*) = 0)$ : Return 0 $\text{hm}^* := H(vk_{i^*}, m^*)$ If $\exists 1 \leq i \leq \mu \wedge m \in \mathcal{M}_i : H(vk_i, m) = \text{hm}^*$ Return 0 <div style="border: 1px dashed black; padding: 5px; margin: 5px 0;"> Parse <math>\sigma^* := ([\mathbf{t}^*]_1, [u^*]_1, [\mathbf{v}^*]_1)</math>  If <math>[u^*]_1 \neq [x'_{i^*}]_1 + [\mathbf{t}^*]_1^\top \cdot \mathbf{x}(\text{hm}^*)</math>  Return 0  Return 1 </div>	$\mathcal{O}_{\text{SIGN}}(i, m)$ : $\mathbf{s} \leftarrow \mathbb{Z}_q^k; \mathbf{t} := \mathbf{B}\mathbf{s} \in \mathbb{Z}_q^{3k}$ $\text{hm} := H(vk_i, m)$ $u := x'_i + \mathbf{s}^\top \mathbf{B}^\top \mathbf{x}(\text{hm}) \in \mathbb{Z}_q$ $\mathbf{v} := \mathbf{y}'_i + \mathbf{s}^\top \mathbf{B}^\top \mathbf{Y}(\text{hm}) \in \mathbb{Z}_q^{1 \times k}$ <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <math>\mathbf{v} := (\mathbf{z}'_i + \mathbf{t}^\top \mathbf{Z}(\text{hm}) - u \cdot \underline{\mathbf{A}}) \cdot (\overline{\mathbf{A}})^{-1}</math> </div> $\mathcal{M}_i := \mathcal{M}_i \cup \{m\}$ Return $\sigma := ([\mathbf{t}]_1, [u]_1, [\mathbf{v}]_1)$  $\mathcal{O}_{\text{CORR}}(i)$ : $\mathcal{S}^{\text{corr}} := \mathcal{S}^{\text{corr}} \cup \{i\}$ Return $\text{ssk}_i$
---	--

Fig. 10. Games used to prove Theorem 4.

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  $\mathbf{s}' \leftarrow \mathbb{Z}_q^k$ ),  $u' := u + \mathbf{s}'^\top (\mathbf{B}^\top \mathbf{x}(\text{hm}))$  and  $\mathbf{v}' := \mathbf{v} + \mathbf{s}'^\top (\mathbf{B}^\top \mathbf{Y}(\text{hm}))$  and  $\text{hm} := H(vk_i, m)$ . Note that this will not change the view of  $\mathcal{A}$ .

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

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

We note that

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

Thus, if Equation (5) does not hold, then the vector  $[(\mathbf{v} \parallel u)]_1 - ([\mathbf{y}'_{i^*} \parallel x'_{i^*}]_1 + [\mathbf{t}^*]_1^\top \cdot \mathbf{x}(\text{hm}^*)) \in \mathbb{G}_1^{1 \times (k+1)}$  is non-zero and orthogonal to  $[\mathbf{A}]_2$ . Therefore, we bound the difference between  $\mathbb{G}_0$  and  $\mathbb{G}_1$  with the  $\mathcal{D}_k$ -KMDH assumption as

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

**Game  $\mathbb{G}_2$ :** We do not use the values  $\mathbf{Y}_{j,b}$  (for  $1 \leq j \leq \lambda$  and  $b = 0, 1$ ) and  $\mathbf{y}'_i$  (for  $1 \leq i \leq \mu$ ) to simulate  $\mathbb{G}_2$ . We make this change by substituting all  $\mathbf{Y}_{j,b}$  and  $\mathbf{y}'_i$  using the formulas

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

respectively. More precisely, the public parameters  $\text{pp}$  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  $\mathcal{O}_{\text{SIGN}}(i, m)$ , we now compute

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

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

The changes in  $\mathbf{G}_2$  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

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

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

- The public parameters consist of  $\text{pp}_{\text{MAC}} := (\mathcal{P}\mathcal{G}, [\mathbf{B}]_1, ([\mathbf{d}_{i,j}]_1)_{1 \leq i \leq \lambda, j=0,1})$ , 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  $\text{hm}$  is  $([\mathbf{t}]_1, [u]_1)$ , where  $\mathbf{t} := \mathbf{B}\mathbf{s}$  and  $u := x' + \mathbf{t}^\top \mathbf{x}(\text{hm})$ , 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<sup>corr</sup> security of MAC.

**Lemma 4 (Core Lemma).** *For every adversaries  $\mathcal{A}$  interacting with UF-CMA<sup>corr</sup>, there exists an adversary  $\mathcal{B}$  against the  $\mathcal{U}_{3k,k}$ -MDDH assumption in  $\mathbb{G}_1$  with*

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

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

<p><u>UF-CMA<sub>A</sub><sup>corr</sup>:</u>  <math>\beta = 0</math>  <math>\mathcal{P}\mathcal{G} \leftarrow_{\\$} \text{GGen}</math>  <math>\mathbf{B} \leftarrow_{\\$} \mathcal{U}_{3k,k}</math>          For <math>1 \leq i \leq \lambda</math> and <math>j = 0, 1</math>:  <math>\mathbf{x}_{i,j} \leftarrow_{\\$} \mathbb{Z}_q^{3k}</math>  <math>\text{pp}_{\text{MAC}} := (\mathcal{P}\mathcal{G}, [\mathbf{B}]_1, ([\mathbf{B}^\top \mathbf{x}_{i,j}]_1)_{1 \leq i \leq \lambda, j=0,1})</math>          For <math>1 \leq i \leq \mu</math>:  <math>x'_i \leftarrow_{\\$} \mathbb{Z}_q</math>  <math>\mathcal{A}^{\mathcal{O}_{\text{MAC}}(\cdot), \mathcal{O}_{\text{VER}}(\cdot, \cdot), \mathcal{O}'_{\text{CORR}}(\cdot)}(\text{pp}_{\text{MAC}})</math>          Return <math>\beta</math></p>	<p><u><math>\mathcal{O}_{\text{MAC}}(i, \text{hm})</math>:</u>  <math>\mathcal{Q} := \mathcal{Q} \cup \{(i, \text{hm})\}</math>  <math>\mathbf{s} \leftarrow_{\\$} \mathbb{Z}_q^k, \mathbf{t} := \mathbf{B}\mathbf{s} \in \mathbb{Z}_q^{3k}</math>  <math>u := x'_i + \mathbf{t}^\top \mathbf{x}(\text{hm}) \in \mathbb{Z}_q</math>          Return <math>\sigma := ([\mathbf{t}]_1, [u]_1)</math></p> <p><u><math>\mathcal{O}_{\text{VER}}(i^*, \text{hm}^*, ([\mathbf{t}^*]_1, [u^*]_1))</math>:</u> //at most once          If <math>(i^*, \text{hm}^*) \in \mathcal{Q} \vee (i^* \in \mathcal{L})</math>:          Return 0          If <math>[u^*]_1 := [x'_{i^*}]_1 + [\mathbf{t}^{*\top}]_1 \cdot \mathbf{x}(\text{hm}^*)</math>:  <math>\beta := 1</math>          Return 1          Else: Return 0</p> <p><u><math>\mathcal{O}'_{\text{CORR}}(i)</math></u>  <math>\mathcal{L} := \mathcal{L} \cup \{i\}</math>          Return <math>[x'_i]_1</math></p>
--	--

**Fig. 11.** Game UF-CMA<sup>corr</sup> for Lemma 4.

The proof is postponed to Appendix B.

Finally, we bound the probability that the adversary wins in  $\mathbf{G}_2$  using our Core Lemma (Lemma 4) by constructing an adversary  $\mathcal{B}_{\text{MAC}}$  as in Figure 12.

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

In order to analyze  $\Pr[\text{Win}_2]$  we argue as follows. The simulated  $\text{pp}$  and  $(vk_i)_{1 \leq i \leq \mu}$  are distributed as in  $\mathbf{G}_2$ . Further, queries to  $\mathcal{O}_{\text{SIGN}}$  and  $\mathcal{O}_{\text{CORR}}$  from  $ssk_i$  can be perfectly simulated using  $\mathcal{O}_{\text{MAC}}$  and  $\mathcal{O}'_{\text{CORR}}$ , respectively. The additional group elements  $[\mathbf{v}]_1$  from  $\sigma$  and  $[\mathbf{y}'_i]_1$  can be simulated as in  $\mathbf{G}_2$ . Finally, using a valid forgery  $(i^*, \text{m}^*, \sigma^*)$  output by  $\mathcal{A}$ ,  $\mathcal{B}_{\text{MAC}}$  wins its own game by calling  $\mathcal{O}_{\text{VER}}(i^*, \text{hm}^*, ([\mathbf{t}^*]_1, [u^*]_1))$ , where  $([\mathbf{t}^*]_1, [u^*]_1)$  is a valid MAC tag on  $\text{hm}^*$  for user  $i^*$ .  $\square$

## 7 Concrete Instantiation of Our AKE Protocols

In this section, we present concrete instantiation of our AKE protocols. We first provide a generic construction of  $\epsilon$ -MU-SIM KEM from Universal<sub>2</sub> Hash Proof System (HPS) in Subsection 7.2,

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

**Fig. 12.** Reduction  $\mathcal{B}_{\text{MAC}}$  to bound the winning probability in  $\mathcal{G}_2$ .  $\mathcal{B}_{\text{MAC}}$  receives  $\text{pp}_{\text{MAC}}$  and gets oracle access to  $\mathcal{O}_{\text{MAC}}$  and  $\mathcal{O}_{\text{VER}}$ , and  $\mathcal{O}'_{\text{CORR}}$  as in Figure 11.

then give an instantiation of HPS from MDDH (and a function  $H$ ) in Subsection 7.3. This yields concrete  $\epsilon$ -MU-SIM KEM schemes based on the MDDH assumptions.

For  $\text{AKE}_{3\text{msg}}$ , we use our new signature scheme  $\text{SIG}_{\text{MDDH}}$  (Figure 9) and the  $\epsilon$ -MU-SIM KEM constructed from the MDDH-based hash proof system. For  $\text{AKE}_{3\text{msg}}^{\text{state}}$ , 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 (cf. Remark 1).

In practice, we can consider the function  $H$  used in the HPS instantiation in Subsection 7.3 as a collision-resistant hash function and thus choose parameter  $t = 1$  (see Remark 7 and Remark 8). Then, the resulting 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 9. Therefore, the first message is a bitstring of length  $\lambda$ , 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  $\text{AKE}_{2\text{msg}}$  using our signature scheme from Figure 9 and the MUC-otCCA secure KEM from Han *et al.* [24].  $\gamma$ -diversity of the KEM is proven in [31, Appendix D.2]. We analyze the communication complexity of  $\text{AKE}_{2\text{msg}}$  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.

## 7.1 Definitions of HPS

We give the formal definition of Hash Proof System (HPS) according to [10].

**Definition 20 (HPS).** *A hash proof system  $\text{HPS} = (\text{HPS.Setup}, \text{Pub}, \text{Priv})$  consists of a tuple of PPT algorithms:*

- $\text{pp} \leftarrow_{\mathcal{S}} \text{HPS.Setup}$ : *The setup algorithm outputs a public parameter  $\text{pp}$ , which implicitly defines  $(\mathcal{L}, \mathcal{X}, \mathcal{SK}, \mathcal{PK}, \Pi, \Lambda_{(\cdot)}, \alpha)$ , where  $\mathcal{L} \subseteq \mathcal{X}$  is an NP-language with universe  $\mathcal{X}$ ,  $\mathcal{SK}$  is the hashing key space,  $\mathcal{PK}$  is the projection key space,  $\Pi$  is the hash value space,  $\Lambda_{(\cdot)} : \mathcal{X} \rightarrow \Pi$  is a family of efficiently computable hash functions indexed by a hashing key  $sk \in \mathcal{SK}$ , and  $\alpha : \mathcal{SK} \rightarrow \mathcal{PK}$  is an efficiently computable projection function.*

*We assume that there are PPT algorithms for sampling  $x \leftarrow_{\mathcal{S}} \mathcal{L}$  uniformly together with a witness  $w$ , sampling  $x \leftarrow_{\mathcal{S}} \mathcal{X} \setminus \mathcal{L}$  uniformly, sampling  $x \leftarrow_{\mathcal{S}} \mathcal{X}$  uniformly, and sampling  $sk \leftarrow_{\mathcal{S}} \mathcal{SK}$  uniformly. We require  $\text{pp}$  to be an implicit input of other algorithms.*

- $\pi \leftarrow \text{Pub}(pk, x, w)$ : The deterministic public evaluation algorithm outputs the hash value  $\pi = \Lambda_{sk}(x) \in \Pi$  of  $x \in \mathcal{L}$ , with help of a projection key  $pk = \alpha(sk)$  and a witness  $w$  for  $x \in \mathcal{L}$ .
- $\pi \leftarrow \text{Priv}(sk, x)$ : The deterministic private evaluation algorithm outputs the hash value  $\pi = \Lambda_{sk}(x) \in \Pi$  of  $x \in \mathcal{X}$  with help of the hashing key  $sk$ .

We require that for all  $\text{pp} \in \text{HPS.Setup}$ , all hashing keys  $sk \in \mathcal{SK}$  with the corresponding projection key  $pk := \alpha(sk)$ , all  $x \in \mathcal{L}$  with all possible witnesses  $w$ , it holds that  $\text{Pub}(pk, x, w) = \Lambda_{sk}(x) = \text{Priv}(sk, x)$ .

HPS is associated with a subset membership problem (SMP). Any SMP can be extended to multi-fold SMP with a security loss of the number of folds.

**Definition 21 (SMP).** Let  $\mathcal{A}$  be an adversary against the subset membership problem (SMP) of HPS. The advantage of  $\mathcal{A}$  is defined as

$$\text{Adv}_{\text{HPS}}^{\text{smp}}(\mathcal{A}) := |\Pr[\mathcal{A}(\text{pp}, x) = 1] - \Pr[\mathcal{A}(\text{pp}, x') = 1]|,$$

where  $\text{pp} \leftarrow_{\$} \text{HPS.Setup}$ ,  $x \leftarrow_{\$} \mathcal{L}$ , and  $x' \leftarrow_{\$} \mathcal{X} \setminus \mathcal{L}$ .

**Definition 22 (Multi-fold SMP).** Let  $\mathcal{A}$  be an adversary against the multi-fold subset membership problem (SMP) of HPS. The advantage of  $\mathcal{A}$  is defined as

$$\text{Adv}_{\text{HPS}, \mu}^{\text{msmp}}(\mathcal{A}) := |\Pr[\mathcal{A}(\text{pp}, \{x_j\}_{j \in [\mu]}) = 1] - \Pr[\mathcal{A}(\text{pp}, \{x'_j\}_{j \in [\mu]}) = 1]|$$

where  $\text{pp} \leftarrow_{\$} \text{HPS.Setup}$ ,  $x_1, \dots, x_\mu \leftarrow_{\$} \mathcal{L}$  and  $x'_1, \dots, x'_\mu \leftarrow_{\$} \mathcal{X} \setminus \mathcal{L}$ .

For some random-self reducible problems like MDDH, the hardness of multi-fold SMP can be tightly reduced to that of SMP, i.e.,  $\text{Adv}_{\text{HPS}, \mu}^{\text{msmp}}(\mathcal{A}) \approx \text{Adv}_{\text{HPS}}^{\text{smp}}(\mathcal{B})$ . See Subsection 6.1 for more details. Our instantiations of HPS from MDDH is shown in Subsection 7.3.

**Definition 23 ( $\epsilon$ -Universal<sub>2</sub> of HPS).** A hash proof system HPS is  $\epsilon$ -universal<sub>2</sub>, if for all  $\text{pp} \in \text{HPS.Setup}$ , for all  $pk \in \mathcal{PK}$ , all  $x, x^* \in \mathcal{X}$  with  $x^* \notin \mathcal{L} \cup \{x\}$ , and all  $\pi, \pi^* \in \Pi$ , it holds that

$$\Pr[\Lambda_{sk}(x^*) = \pi^* \mid \alpha(sk) = pk, \Lambda_{sk}(x) = \pi] \leq \epsilon,$$

where the probability is over  $sk \leftarrow_{\$} \mathcal{SK}$ . If  $\epsilon = 1/|\Pi|$ , then HPS is perfectly universal<sub>2</sub>.

Below we define an extracting notion, which is adapted from [16].

**Definition 24 ( $\gamma$ -Extracting of HPS).** A hash proof system HPS is  $\gamma$ -extracting, if it is  $\gamma$ -extracting<sub>1</sub> and  $\gamma$ -extracting<sub>2</sub>.

(1)  **$\gamma$ -Extracting<sub>1</sub>**: For all  $\text{pp} \in \text{HPS.Setup}$ , all  $x \in \mathcal{L}$  and all  $\pi \in \Pi$ , it holds that  $\Pr_{sk \leftarrow_{\$} \mathcal{SK}}[\Lambda_{sk}(x) = \pi] \leq 2^{-\gamma}$ ;

(2)  **$\gamma$ -Extracting<sub>2</sub>**: For all  $\text{pp} \in \text{HPS.Setup}$ , all  $x, x^* \in \mathcal{L}$  with  $x^* \neq x$ , and all  $\pi, \pi^* \in \Pi$ , it holds that  $\Pr_{sk \leftarrow_{\$} \mathcal{SK}}[\Lambda_{sk}(x^*) = \pi^* \mid \Lambda_{sk}(x) = \pi] \leq 2^{-\gamma}$ .

If  $\gamma = \log |\Pi|$ , then HPS is perfectly extracting.

## 7.2 $\epsilon$ -MU-SIM Secure KEM from Universal<sub>2</sub>-HPS

Let  $\text{HPS} = (\text{HPS.Setup}, \text{Pub}, \text{Priv})$  be a universal<sub>2</sub>-HPS associated with a hard multi-fold subset membership problem (SMP) and enjoying an extracting property. We present a simple construction  $\text{KEM}_{\text{HPS}} = (\text{KEM.Setup}, \text{KEM.Gen}, \text{Encap}, \text{Decap}, \text{Encap}^*)$  from HPS as follows.

- **KEM.Setup**:  $\text{pp} \leftarrow_{\$} \text{HPS.Setup}$ , where  $\text{pp} = (\mathcal{L}, \mathcal{X}, \mathcal{SK}, \mathcal{PK}, \Pi, \Lambda_{(\cdot)}, \alpha)$ . Return  $\text{pp}_{\text{KEM}} := \text{pp}$ . Here the encapsulation key space  $\mathcal{K} := \Pi$ , the ciphertext space  $\mathcal{CT} := \mathcal{X}$  and the public key & secret key spaces are  $\mathcal{PK} \times \mathcal{SK}$ .
- **KEM.Gen**( $\text{pp}_{\text{KEM}}$ ): Choose  $sk \leftarrow_{\$} \mathcal{SK}$  and return  $(pk := \alpha(sk), sk)$ .
- **Encap**( $pk$ ): Sample  $x \leftarrow_{\$} \mathcal{L}$  with witness  $w$  and return  $(c := x, K := \text{Pub}(pk, x, w) = \Lambda_{sk}(x))$ .
- **Decap**( $sk, c$ ): Compute and return  $K' := \text{Priv}(sk, c)$ .
- **Encap\***( $sk$ ): Sample  $x \leftarrow_{\$} \mathcal{X} \setminus \mathcal{L}$  and return  $(c := x, K := \text{Priv}(sk, x) = \Lambda_{sk}(x))$ .

The correctness of  $\text{KEM}_{\text{HPS}}$  follows from the correctness of HPS. Now we prove the strong  $\epsilon$ -MU-SIM security of  $\text{KEM}_{\text{HPS}}$ .

**Lemma 5 ( $\epsilon$ -MU-SIM Security of  $\text{KEM}_{\text{HPS}}$ ).** *Let  $\epsilon'$  be a real number and  $\Pi$  the hash value space of HPS. If HPS is an  $\epsilon'$ -universal<sub>2</sub> HPS associated with a hard multi-fold SMP, then  $\text{KEM}_{\text{HPS}}$  is  $\epsilon$ -MU-SIM secure with uniformity parameter  $\epsilon = |\Pi|^2 \cdot (\epsilon' - 1/|\Pi|)$ .*

*Concretely, for any polynomial  $\mu$ , any adversary  $\mathcal{A}$ , there exists an adversary  $\mathcal{B}$ , such that  $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$  and*

$$\text{Adv}_{\text{KEM,Encap}^*,\mu}^{\text{mu-sim}}(\mathcal{A}) \leq \text{Adv}_{\text{HPS},\mu}^{\text{msmp}}(\mathcal{B}). \quad (7)$$

*Remark 5.* If HPS is perfectly universal<sub>2</sub> (i.e.,  $\epsilon' = 1/|\Pi|$ ), then  $\text{KEM}_{\text{HPS}}$  has 0-uniformity for the key encapsulated by  $\text{Encap}^*$  (i.e.,  $\epsilon = 0$ ).

**Proof of Lemma 5.** To prove (7), we build an adversary  $\mathcal{B}$  solving the multi-fold SMP by invoking  $\mathcal{A}$ . Given a challenge  $(\text{pp}, \{x_i\}_{i \in [\mu]})$ ,  $\mathcal{B}$  wants to determine whether  $x_i \leftarrow \mathcal{L}$  or  $x_i \leftarrow \mathcal{X} \setminus \mathcal{L}$ .  $\mathcal{B}$  sets  $\text{pp}_{\text{KEM}} := \text{pp}$ , samples  $(pk_i, sk_i) \leftarrow \text{KEM.Gen}(\text{pp}_{\text{KEM}})$ , and computes  $(c_i, K_i) := (x_i, \Lambda_{sk_i}(x_i))$ . Then  $\mathcal{B}$  invokes  $\mathcal{A}(\{pk_i, sk_i, c_i, K_i\}_{i \in [\mu]})$ , and returns the output of  $\mathcal{A}$  to its challenger. If  $x_i \leftarrow \mathcal{L}$ ,  $(c_i, K_i) = (x_i, \Lambda_{sk_i}(x_i))$  has the same distribution as the output of  $\text{Encap}(pk_i)$ ; if  $x_i \leftarrow \mathcal{X} \setminus \mathcal{L}$ ,  $(c_i, K_i) = (x_i, \Lambda_{sk_i}(x_i))$  has the same distribution as the output of  $\text{Encap}^*(sk_i)$ . Consequently,  $\mathcal{B}$  solves the multi-fold SMP as long as  $\mathcal{A}$  distinguishes  $\text{Encap}$  and  $\text{Encap}^*$ , and (7) holds.

We now proceed to prove  $\epsilon$ -Uniformity of  $\text{Encap}^*$  as defined in (1). Firstly,  $\epsilon'$ -universal<sub>2</sub> of HPS means

$$\Pr[\Lambda_{sk}(c^*) = \pi^* \mid \alpha(sk) = pk, \Lambda_{sk}(c) = \pi] \leq \epsilon',$$

for all  $\text{pp} \leftarrow \text{HPS.Setup}$ , all  $pk \in \mathcal{PK}$ , all  $c, c^* \in \mathcal{X}$  with  $c^* \notin \mathcal{L} \cup \{c\}$ , and all  $\pi, \pi^* \in \Pi$ , where the probability is over  $sk \leftarrow \mathcal{SK}$ . By an averaging argument over  $(c, c^*, pk)$ ,  $\epsilon'$ -universal<sub>2</sub> implies that for any (unbounded) adversary  $\mathcal{B}$ , it holds that

$$\begin{aligned} & \left| \Pr[c \leftarrow \mathcal{B}(pk, c^*) : c \neq c^* \wedge \mathcal{B}(pk, c^*, K^*, \text{Decap}(sk, c)) \Rightarrow 1] \right. \\ & \quad \left. - \Pr[c \leftarrow \mathcal{B}(pk, c^*) : c \neq c^* \wedge \mathcal{B}(pk, c^*, R, \text{Decap}(sk, c)) \Rightarrow 1] \right| \\ & \leq |\Pi| \cdot (\epsilon' - 1/|\Pi|), \end{aligned} \quad (8)$$

where the probability is over  $\text{pp}_{\text{KEM}} \leftarrow \text{KEM.Setup}$ ,  $(pk, sk) \leftarrow \text{KEM.Gen}(\text{pp}_{\text{KEM}})$ ,  $(c^*, K^*) \leftarrow \text{Encap}^*(sk)$ ,  $R \leftarrow \mathcal{K}$  and the internal randomness of  $\mathcal{B}$ . The averaging argument essentially uses the law of total probability over  $(c, c^*, pk)$ . We note that here  $c$  is not allowed to depend on  $K^*$  or  $R$ , but  $\epsilon$ -Uniformity allows  $c$  arbitrarily dependent on  $K^*$  or  $R$ . This gap can be filled by a leveraging argument, as shown below.

Suppose towards a contradiction that there exists an (unbounded) adversary  $\mathcal{A}$ , so that

$$\begin{aligned} & \left| \Pr[c \leftarrow \mathcal{A}(pk, c^*, K^*) : c \neq c^* \wedge \mathcal{A}(pk, c^*, K^*, \text{Decap}(sk, c)) \Rightarrow 1] \right. \\ & \quad \left. - \Pr[c \leftarrow \mathcal{A}(pk, c^*, R) : c \neq c^* \wedge \mathcal{A}(pk, c^*, R, \text{Decap}(sk, c)) \Rightarrow 1] \right| > \epsilon. \end{aligned} \quad (9)$$

Then we can construct an (unbounded) adversary  $\mathcal{B}$  to contradict with (8).  $\mathcal{B}$  is constructed by the following leveraging argument.

- Given  $(pk, c^*)$ ,  $\mathcal{B}$  samples a  $T' \leftarrow \Pi$  uniformly, invokes  $c \leftarrow \mathcal{A}(pk, c^*, T')$  and outputs  $c$ .
- Then  $\mathcal{B}$  receives  $(pk, c^*, T, \text{Decap}(sk, c))$ , where  $T = K^*$  or  $T = R$ , and invokes  $b \leftarrow \mathcal{A}(pk, c^*, T, \text{Decap}(sk, c))$ .
- If  $T' = T$ ,  $\mathcal{B}$  outputs the bit  $b$  output by  $\mathcal{A}$ ; otherwise,  $\mathcal{B}$  outputs 0.

If  $T' = T$ , which happens with probability  $1/|\Pi|$ ,  $\mathcal{B}$  perfectly simulates the experiment defined in (9) for  $\mathcal{A}$ , thus  $\mathcal{B}$  distinguishes  $T = K^*$  from  $T = R$  as long as  $\mathcal{A}$  does; if  $T' \neq T$ ,  $\mathcal{B}$  always outputs 0 no matter  $T = K^*$  or  $T = R$ . Overall,

$$\begin{aligned} & \mathcal{B}'\text{'s distinguishing advantage in the experiment defined in (8)} \\ & = \mathcal{A}'\text{'s distinguishing advantage in the experiment defined in (9)} / |\Pi| \\ & > \epsilon / |\Pi| = |\Pi| \cdot (\epsilon' - 1/|\Pi|), \end{aligned}$$

which contradicts with (8). This completes the proof of  $\epsilon$ -Uniformity.  $\square$

Moreover, we show the diversity of  $\text{KEM}_{\text{HPS}}$  as long as HPS is extracting.

**Lemma 6 ( $\gamma$ -Diversity of  $\text{KEM}_{\text{HPS}}$ ).** Let  $\gamma'$  be a real number,  $\Pi$  the hash value space of HPS and  $\mathcal{L}$  the language space. If HPS is  $\gamma'$ -extracting, then  $\text{KEM}_{\text{HPS}}$  has  $\gamma$ -diversity with  $\gamma = 2\gamma' + \log |\mathcal{L}| - \log(|\Pi| \cdot |\mathcal{L}| + 2^{2\gamma'})$ .

*Remark 6.* If HPS is perfectly extracting (i.e.,  $\gamma' = \log |\Pi|$ ), then  $\text{KEM}_{\text{HPS}}$  is  $\gamma$ -diverse with  $\gamma = \log |\Pi| + \log |\mathcal{L}| - \log(|\Pi| + |\mathcal{L}|)$ . For our concrete instantiation  $\text{HPS}_{\text{MDDH}}$  shown in Subsection 7.3, which is perfectly extracting and has  $|\Pi| = q$  and  $|\mathcal{L}| = q^k - 1$ , the resulting  $\text{KEM}_{\text{HPS}_{\text{MDDH}}}$  is  $\gamma$ -diverse with  $\gamma = \log q + \log(q^k - 1) - \log(q + q^k - 1) \geq \log(q/3)$ .

**Proof of Lemma 6.** By construction, we have

$$\begin{aligned}
(1) \quad & \Pr \left[ \begin{array}{l} (pk, sk) \leftarrow_{\$} \text{KEM.Gen}(\text{pp}_{\text{KEM}}); \\ r, r' \leftarrow_{\$} \mathcal{R}; (c, K) \leftarrow \text{Encap}(pk; r); (c', K') \leftarrow \text{Encap}(pk; r') : K = K' \end{array} \right] \\
&= \Pr [sk \leftarrow_{\$} \mathcal{SK}; x, x' \leftarrow_{\$} \mathcal{L} : \Lambda_{sk}(x) = \Lambda_{sk}(x')] \\
&= \sum_{a \in \mathcal{L}} \Pr_{x \leftarrow_{\$} \mathcal{L}} [x = a] \cdot \sum_{a' \in \mathcal{L}} \Pr_{x' \leftarrow_{\$} \mathcal{L}} [x' = a'] \cdot \Pr_{sk \leftarrow_{\$} \mathcal{SK}} [\Lambda_{sk}(a) = \Lambda_{sk}(a')] \\
&= \sum_{a \in \mathcal{L}} \frac{1}{|\mathcal{L}|} \cdot \left( \sum_{a' \in \mathcal{L} \setminus \{a\}} \frac{1}{|\mathcal{L}|} \cdot \sum_{\pi \in \Pi} \Pr_{sk \leftarrow_{\$} \mathcal{SK}} [\Lambda_{sk}(a) = \Lambda_{sk}(a') = \pi] + \frac{1}{|\mathcal{L}|} \cdot 1 \right) \\
&= \sum_{a \in \mathcal{L}} \frac{1}{|\mathcal{L}|} \cdot \left( \sum_{a' \in \mathcal{L} \setminus \{a\}} \frac{1}{|\mathcal{L}|} \cdot \sum_{\pi \in \Pi} \underbrace{\Pr_{sk \leftarrow_{\$} \mathcal{SK}} [\Lambda_{sk}(a) = \pi]}_{\leq 2^{-\gamma'} \text{ by } \gamma'\text{-extracting}_1} \cdot \underbrace{\Pr_{sk \leftarrow_{\$} \mathcal{SK}} [\Lambda_{sk}(a') = \pi | \Lambda_{sk}(a) = \pi]}_{\leq 2^{-\gamma'} \text{ by } \gamma'\text{-extracting}_2 \text{ of HPS}} + \frac{1}{|\mathcal{L}|} \right) \\
&\leq |\Pi| \cdot (2^{-\gamma'})^2 + 1/|\mathcal{L}|, \\
(2) \quad & \Pr \left[ \begin{array}{l} (pk, sk) \leftarrow_{\$} \text{KEM.Gen}(\text{pp}_{\text{KEM}}); (pk', sk') \leftarrow_{\$} \text{KEM.Gen}(\text{pp}_{\text{KEM}}); \\ r \leftarrow_{\$} \mathcal{R}; (c, K) \leftarrow \text{Encap}(pk; r); (c', K') \leftarrow \text{Encap}(pk'; r) : K = K' \end{array} \right] \\
&= \Pr [sk, sk' \leftarrow_{\$} \mathcal{SK}; x \leftarrow_{\$} \mathcal{L} : \Lambda_{sk}(x) = \Lambda_{sk'}(x)] \\
&= \sum_{a \in \mathcal{L}} \Pr_{x \leftarrow_{\$} \mathcal{L}} [x = a] \cdot \sum_{\pi \in \Pi} \Pr [sk, sk' \leftarrow_{\$} \mathcal{SK} : \Lambda_{sk}(a) = \Lambda_{sk'}(a) = \pi] \\
&= \sum_{a \in \mathcal{L}} \frac{1}{|\mathcal{L}|} \cdot \sum_{\pi \in \Pi} \underbrace{\Pr_{sk \leftarrow_{\$} \mathcal{SK}} [\Lambda_{sk}(a) = \pi]}_{\leq 2^{-\gamma'} \text{ by } \gamma'\text{-extracting}_1 \text{ of HPS}} \cdot \underbrace{\Pr_{sk' \leftarrow_{\$} \mathcal{SK}} [\Lambda_{sk'}(a) = \pi]}_{\leq 2^{-\gamma'} \text{ by } \gamma'\text{-extracting}_1 \text{ of HPS}} \\
&\leq |\Pi| \cdot (2^{-\gamma'})^2.
\end{aligned}$$

Thus,  $\text{KEM}_{\text{HPS}}$  has  $\gamma$ -diversity with  $\gamma = 2\gamma' + \log |\mathcal{L}| - \log(|\Pi| \cdot |\mathcal{L}| + 2^{2\gamma'})$ .  $\square$

### 7.3 Universal<sub>2</sub> Hash Proof System from MDDH

In this subsection, we construct an MDDH-based universal<sub>2</sub> hash proof system  $\text{HPS}_{\text{MDDH}}$  which is also extracting. Then together with the transformation in Subsection 7.2, we immediately obtain an MDDH-based  $\epsilon$ -MU-SIM secure KEM which also enjoys  $\gamma$ -diversity.

Our  $\text{HPS}_{\text{MDDH}}$  extends the DDH-based hash proof system proposed by Cramer and Shoup in [10] to MDDH assumptions. Let  $\mathcal{G} = (\mathbb{G}, q, \mathcal{P})$  be a description of cyclic group  $\mathbb{G}$  of prime order  $q$  and with generator  $\mathcal{P}$ . Let  $\mathcal{D}_k$  be a matrix distribution,  $t \in \mathbb{N}$ , and  $\mathcal{H} = \{\text{H} : \mathbb{G}^{k+1} \rightarrow \mathbb{Z}_q^t\}$  a family of hash functions from  $\mathbb{G}^{k+1}$  to  $\mathbb{Z}_q^t$ .

- $\text{HPS.Setup}$  picks  $\mathbf{A} \leftarrow_{\$} \mathcal{D}_k$ ,  $\text{H} \leftarrow_{\$} \mathcal{H}$ , and outputs public parameter  $\text{pp} := ([\mathbf{A}], \text{H})$ .  $\text{pp}$  implicitly defines  $(\mathcal{L}, \mathcal{X}, \mathcal{SK}, \mathcal{PK}, \Pi, \Lambda_{(\cdot)}, \alpha)$  as follows.
  - $\mathcal{X} := \mathbb{G}^{k+1} \setminus \{[\mathbf{0}]\}$  and the language  $\mathcal{L} := \mathcal{L}_{\mathbf{A}} := \{[c] = [\mathbf{A}\mathbf{w}] \in \mathbb{G}^{k+1} : \mathbf{w} \in \mathbb{Z}_q^k \setminus \{\mathbf{0}\}\} \subseteq \mathcal{X}$ . The value  $\mathbf{w}$  is a witness of  $[c] \in \mathcal{L}$ .
  - $\mathcal{SK} := \mathbb{Z}_q^{(t+1) \times (k+1)}$ ,  $\mathcal{PK} := \mathbb{G}^{(t+1) \times k}$ , and hash value space  $\Pi := \mathbb{G}$ .
  - For  $sk = \mathbf{K} \in \mathcal{SK}$ , define  $pk = \alpha(sk) := [\mathbf{K}\mathbf{A}] \in \mathcal{PK}$ .
  - For  $[c] \in \mathcal{X}$  and  $sk = \mathbf{K} \in \mathcal{SK}$ , define  $\Lambda_{sk}([c]) := (1, \tau^\top) \cdot \mathbf{K} \cdot [c] \in \mathbb{G}$  with  $\tau := \text{H}([c]) \in \mathbb{Z}_q^t$ .
- $\text{Pub}(pk = [\mathbf{K}\mathbf{A}], [c] \in \mathcal{L}, \mathbf{w} \in \mathbb{Z}_q^k)$ : Compute  $\tau := \text{H}([c]) \in \mathbb{Z}_q^t$ , and return  $[\pi] := (1, \tau^\top) \cdot [\mathbf{K}\mathbf{A}] \cdot \mathbf{w} \in \mathbb{G}$ .
- $\text{Priv}(sk = \mathbf{K}, [c] \in \mathcal{X})$ : Compute  $\tau := \text{H}([c]) \in \mathbb{Z}_q^t$ , and return  $[\pi] := (1, \tau^\top) \cdot \mathbf{K} \cdot [c] \in \mathbb{G}$ .

It is straightforward to check the correctness of  $\text{HPS}_{\text{MDDH}}$ . The associated SMP is exactly the  $\mathcal{D}_k$ -MDDH (cf. Definition 17), and the associated multi-fold SMP is exactly multi-fold  $\mathcal{D}_k$ -MDDH. By the random self-reducibility of  $\mathcal{D}_k$ -MDDH (cf. Lemma 10), we have the following corollary.

**Corollary 1 (Multi-fold SMP).** *For any  $\mu \in \mathbb{N}$  and any  $\mathcal{A}$  there exists an adversary  $\mathcal{B}$  with  $\text{Adv}_{\text{HPS}_{\text{MDDH}, \mu}}^{\text{msmp}}(\mathcal{A}) = \text{Adv}_{\text{GGen}, \mathcal{D}_k, \mathbb{G}}^{\mu\text{-MDDH}}(\mathcal{A}) \leq \text{Adv}_{\text{GGen}, \mathcal{D}_k, \mathbb{G}}^{\text{MDDH}}(\mathcal{B}) + \frac{1}{q-1}$ .*

Below we show the perfect universal<sub>2</sub> and extracting properties, respectively.

**Lemma 7 (Perfectly Universal<sub>2</sub> of  $\text{HPS}_{\text{MDDH}}$ ).** *(1) If  $\mathcal{H} = \mathcal{H}_{\text{inj}} = \{\text{H} : \mathbb{G}^{k+1} \rightarrow \mathbb{Z}_q^t\}$  is a family of injective functions (in this case  $t \geq k+1$ ), then  $\text{HPS}_{\text{MDDH}}$  is perfectly universal<sub>2</sub>. (2) If  $\mathcal{H} = \mathcal{H}_{\text{cr}} = \{\text{H} : \mathbb{G}^{k+1} \rightarrow \mathbb{Z}_q^t\}$  is a family of collision-resistant hash functions (in this case  $t = 1$ ), then  $\text{HPS}_{\text{MDDH}}$  is perfectly universal<sub>2</sub> assuming that there are no collisions.*

**Proof of Lemma 7.** For  $sk = \mathbf{K} \leftarrow_s \mathbb{Z}_q^{(t+1) \times (k+1)}$ , for any  $[\mathbf{c}] \in \mathcal{X}$ , any  $[\mathbf{c}^*] \in \mathcal{X} \setminus (\mathcal{L} \cup \{[\mathbf{c}]\})$ , any  $pk \in \mathcal{PK}$  and any  $[\pi] \in \mathbb{G}$ , we consider the distribution of  $\Lambda_{sk}([\mathbf{c}^*]) = (1, \boldsymbol{\tau}^{*\top}) \cdot \mathbf{K} \cdot [\mathbf{c}^*]$  conditioned on  $pk = \alpha(sk) = [\mathbf{KA}]$  and  $[\pi] = \Lambda_{sk}([\mathbf{c}]) = (1, \boldsymbol{\tau}^\top) \cdot \mathbf{K} \cdot [\mathbf{c}]$ , where  $\boldsymbol{\tau}^* := \text{H}([\mathbf{c}^*])$ ,  $\boldsymbol{\tau} := \text{H}([\mathbf{c}]) \in \mathbb{Z}_q^t$ .

Firstly, we prove (1). Since  $[\mathbf{c}^*] \neq [\mathbf{c}]$  and  $\text{H}$  is injective, we have  $\boldsymbol{\tau}^* \neq \boldsymbol{\tau}$ . Let  $\mathbf{a}^\perp \in \mathbb{Z}_q^{k+1}$  be an arbitrary non-zero vector in the kernel space of  $\mathbf{A}$  such that  $(\mathbf{a}^\perp)^\top \mathbf{A} = \mathbf{0}$  holds. It is clear that  $(\mathbf{a}^\perp)^\top \cdot [\mathbf{c}^*] \neq [0]$  since  $[\mathbf{c}^*] \notin \text{span}([\mathbf{A}])$ . Let  $\mathbf{b} \in \mathbb{Z}_q^{t+1}$  be an arbitrary non-zero vector such that  $(1, \boldsymbol{\tau}^\top) \cdot \mathbf{b} = 0$  but  $(1, \boldsymbol{\tau}^{*\top}) \cdot \mathbf{b} = 1$ . We can always find such  $\mathbf{b}$  since  $(1, \boldsymbol{\tau}^\top)$  and  $(1, \boldsymbol{\tau}^{*\top})$  are linearly independent (due to  $\boldsymbol{\tau}^* \neq \boldsymbol{\tau}$ ). Equivalently,  $sk$  can be sampled via  $sk = \mathbf{K} := \tilde{\mathbf{K}} + \mu \cdot \mathbf{b}(\mathbf{a}^\perp)^\top \in \mathbb{Z}_q^{(t+1) \times (k+1)}$ , where  $\tilde{\mathbf{K}} \leftarrow_s \mathbb{Z}_q^{(t+1) \times (k+1)}$  and  $\mu \leftarrow_s \mathbb{Z}_q$ . In this case, we have  $pk = \alpha(sk) = [\mathbf{KA}] = [\tilde{\mathbf{K}}\mathbf{A}]$ ,  $[\pi] = \Lambda_{sk}([\mathbf{c}]) = (1, \boldsymbol{\tau}^\top) \cdot \mathbf{K} \cdot [\mathbf{c}] = (1, \boldsymbol{\tau}^\top) \cdot \tilde{\mathbf{K}} \cdot [\mathbf{c}]$ , which may leak  $\tilde{\mathbf{K}}$ , but the value of  $\mu$  is completely hidden. Moreover,

$$\Lambda_{sk}([\mathbf{c}^*]) = (1, \boldsymbol{\tau}^{*\top}) \cdot \mathbf{K} \cdot [\mathbf{c}^*] = (1, \boldsymbol{\tau}^{*\top}) \cdot \tilde{\mathbf{K}} \cdot [\mathbf{c}^*] + \underbrace{\mu \cdot (1, \boldsymbol{\tau}^{*\top}) \cdot \mathbf{b}}_{=1} \cdot \underbrace{(\mathbf{a}^\perp)^\top}_{\neq [0]} \cdot [\mathbf{c}^*].$$

Thanks to the uniformity of  $\mu$ ,  $\Lambda_{sk}([\mathbf{c}^*])$  is uniformly distributed over  $\mathbb{G}$  conditioned on  $pk = \alpha(sk)$  and  $[\pi] = \Lambda_{sk}([\mathbf{c}])$ . This shows that  $\text{HPS}_{\text{MDDH}}$  is perfectly universal<sub>2</sub>.

For (2), we also have  $\boldsymbol{\tau}^* \neq \boldsymbol{\tau}$  as long as no collision happens. Then the analysis of perfectly universal<sub>2</sub> is similar to (1).  $\square$

*Remark 7.* Instantiating  $\text{HPS}_{\text{MDDH}}$  using  $\mathcal{H}_{\text{cr}}$  is more efficient than using  $\mathcal{H}_{\text{inj}}$  since  $t$  can be as small as 1. Taking into account the collision resistance of  $\mathcal{H}_{\text{cr}}$ ,  $\text{HPS}_{\text{MDDH}}$  using  $\mathcal{H}_{\text{cr}}$  is perfectly universal<sub>2</sub> only in a computational sense, and the  $\text{KEM}_{\text{HPS}_{\text{MDDH}}}$  derived from such  $\text{HPS}_{\text{MDDH}}$  (cf. Subsection 7.2) has only computational  $\epsilon$ -uniformity. Nevertheless, this does not affect the tight security reduction from  $\text{AKE}_{3\text{msg}}^{\text{state}}$  to  $\text{KEM}_{\text{HPS}_{\text{MDDH}}}$  in Theorem 1 and  $\text{AKE}_{3\text{msg}}$  to  $\text{KEM}_{\text{HPS}_{\text{MDDH}}}$  in Theorem 2, since we can always add an extra game (e.g., between  $\text{G}_0$  and  $\text{G}_1$ ) to deal with collisions in the security proofs. In this extra game, the challenger aborts immediately when collision happens. Then in subsequent games, the adversary wins only if no collision happens, and in such scenarios,  $\text{HPS}_{\text{MDDH}}$  using  $\mathcal{H}_{\text{cr}}$  is perfect universal<sub>2</sub> and the  $\text{KEM}_{\text{HPS}_{\text{MDDH}}}$  derived from  $\text{HPS}_{\text{MDDH}}$  (cf. Subsection 7.2) has  $\epsilon$ -uniformity. On the other hand, we note that it is easy to construct hash functions  $\mathcal{H}_{\text{cr}}$  whose collision resistance can be tightly reduced to the discrete logarithm assumption [12, 8], so this extra game does not affect the tightness of our AKE protocols.

**Lemma 8 (Perfect Extracting of  $\text{HPS}_{\text{MDDH}}$ ).**  *$\text{HPS}_{\text{MDDH}}$  is perfectly extracting<sub>1</sub>. Moreover, (1) If  $\mathcal{H} = \mathcal{H}_{\text{inj}} = \{\text{H} : \mathbb{G}^{k+1} \rightarrow \mathbb{Z}_q^t\}$  is a family of injective functions (in this case  $t \geq k+1$ ), then  $\text{HPS}_{\text{MDDH}}$  is perfectly extracting<sub>2</sub>. (2) If  $\mathcal{H} = \mathcal{H}_{\text{cr}} = \{\text{H} : \mathbb{G}^{k+1} \rightarrow \mathbb{Z}_q^t\}$  is a family of collision-resistant hash functions (in this case  $t = 1$ ), then  $\text{HPS}_{\text{MDDH}}$  is perfectly extracting<sub>2</sub> assuming that there are no collisions.*

**Proof of Lemma 8.** Firstly, we prove the perfect extracting<sub>1</sub>. For  $sk = \mathbf{K} \leftarrow_s \mathbb{Z}_q^{(t+1) \times (k+1)}$ , for any  $[\mathbf{c}] \in \mathcal{L} = \text{span}[\mathbf{A}] \setminus \{[\mathbf{0}]\}$ , we consider the distribution of  $\Lambda_{sk}([\mathbf{c}]) = (1, \boldsymbol{\tau}^\top) \cdot \mathbf{K} \cdot [\mathbf{c}]$ , where  $\boldsymbol{\tau} := \text{H}([\mathbf{c}]) \in \mathbb{Z}_q^t$ . Since  $[\mathbf{c}] \neq [\mathbf{0}]$  and  $\mathbf{K} \leftarrow_s \mathbb{Z}_q^{(t+1) \times (k+1)}$ , it follows that  $\mathbf{K} \cdot [\mathbf{c}]$  is uniformly distributed over  $\mathbb{G}^{t+1}$ . Moreover,  $(1, \boldsymbol{\tau}^\top)$  is non-zero, thus  $\Lambda_{sk}([\mathbf{c}]) = (1, \boldsymbol{\tau}^\top) \cdot \mathbf{K} \cdot [\mathbf{c}]$  is uniformly distributed over  $\mathbb{G}$ . This shows the perfect extracting<sub>1</sub> of  $\text{HPS}_{\text{MDDH}}$ .

Next, we prove the perfect extracting<sub>2</sub>. For  $sk = \mathbf{K} \leftarrow_s \mathbb{Z}_q^{(t+1) \times (k+1)}$ , for any  $[\mathbf{c}] \in \mathcal{L}$ , any  $[\mathbf{c}^*] \in \mathcal{L} \setminus \{[\mathbf{c}]\}$ , and any  $[\pi] \in \mathbb{G}$ , we consider the distribution of  $\Lambda_{sk}([\mathbf{c}^*]) = (1, \boldsymbol{\tau}^{*\top}) \cdot \mathbf{K} \cdot [\mathbf{c}^*]$  conditioned on  $[\pi] = \Lambda_{sk}([\mathbf{c}]) = (1, \boldsymbol{\tau}^\top) \cdot \mathbf{K} \cdot [\mathbf{c}]$ , where  $\boldsymbol{\tau}^* := \text{H}([\mathbf{c}^*])$ ,  $\boldsymbol{\tau} := \text{H}([\mathbf{c}]) \in \mathbb{Z}_q^t$ .

- For (1), since  $\mathbf{H}$  is injective, we have  $\boldsymbol{\tau}^* \neq \boldsymbol{\tau}$ . Let  $\mathbf{b} \in \mathbb{Z}_q^{t+1}$  be an arbitrary non-zero vector such that  $(1, \boldsymbol{\tau}^\top) \cdot \mathbf{b} = 0$  but  $(1, \boldsymbol{\tau}^{*\top}) \cdot \mathbf{b} = 1$ . We can always find such  $\mathbf{b}$  since  $(1, \boldsymbol{\tau}^\top)$  and  $(1, \boldsymbol{\tau}^{*\top})$  are linearly independent (due to  $\boldsymbol{\tau}^* \neq \boldsymbol{\tau}$ ). Equivalently,  $sk$  can be sampled via  $sk = \mathbf{K} := \tilde{\mathbf{K}} + \mathbf{b} \cdot \mathbf{r}^\top \in \mathbb{Z}_q^{(t+1) \times (k+1)}$ , where  $\tilde{\mathbf{K}} \leftarrow_s \mathbb{Z}_q^{(t+1) \times (k+1)}$  and  $\mathbf{r} \leftarrow_s \mathbb{Z}_q^{k+1}$ . In this case, we have  $[\pi] = \Lambda_{sk}([\mathbf{c}]) = (1, \boldsymbol{\tau}^\top) \cdot \mathbf{K} \cdot [\mathbf{c}] = (1, \boldsymbol{\tau}^\top) \cdot \tilde{\mathbf{K}} \cdot [\mathbf{c}]$ , which may leak  $\tilde{\mathbf{K}}$ , but the value of  $\mathbf{r}$  is completely hidden. Moreover,

$$\Lambda_{sk}([\mathbf{c}^*]) = (1, \boldsymbol{\tau}^{*\top}) \cdot \mathbf{K} \cdot [\mathbf{c}^*] = (1, \boldsymbol{\tau}^{*\top}) \cdot \tilde{\mathbf{K}} \cdot [\mathbf{c}^*] + \underbrace{(1, \boldsymbol{\tau}^{*\top}) \cdot \mathbf{b} \cdot \mathbf{r}^\top}_{=1} \cdot [\mathbf{c}^*].$$

Thanks to the uniformity of  $\mathbf{r}$  and the fact that  $[\mathbf{c}^*] \neq [\mathbf{0}]$ ,  $\mathbf{r}^\top \cdot [\mathbf{c}^*]$  is uniformly distributed over  $\mathbb{G}$ , and this implies that uniformity of  $\Lambda_{sk}([\mathbf{c}^*])$  conditioned on  $[\pi] = \Lambda_{sk}([\mathbf{c}])$ . Hence the perfectly extracting<sub>2</sub> of  $\text{HPS}_{\text{MDDH}}$  follows.

- For (2), we also have  $\boldsymbol{\tau}^* \neq \boldsymbol{\tau}$  as long as no collision happens. Then the analysis of perfect extracting<sub>2</sub> is similar to (1).  $\square$

*Remark 8.* As in Remark 7,  $\text{HPS}_{\text{MDDH}}$  using  $\mathcal{H}_{cr}$  is perfectly extracting<sub>2</sub> only in a computational sense, and the  $\text{KEM}_{\text{HPS}_{\text{MDDH}}}$  derived from such  $\text{HPS}_{\text{MDDH}}$  (cf. Subsection 7.2) has only computational diversity. Nevertheless, this does not affect the tight security reduction from  $\text{AKE}_{3\text{msg}}^{\text{state}}$  to  $\text{KEM}_{\text{HPS}_{\text{MDDH}}}$  in Theorem 1 and  $\text{AKE}_{3\text{msg}}$  to  $\text{KEM}_{\text{HPS}_{\text{MDDH}}}$  in Theorem 2, since we can always add an extra game to deal with collisions in the security proofs.

**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] Chaum, D., van Heijst, E., Pfitzmann, B.: Cryptographically strong undeniable signatures, unconditionally secure for the signer. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 470–484. Springer, Heidelberg (Aug 1992)
- [9] 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)
- [10] 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)
- [11] 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)
- [12] Damgård, I.: Collision free hash functions and public key signature schemes. In: Chaum, D., Price, W.L. (eds.) EUROCRYPT'87. LNCS, vol. 304, pp. 203–216. Springer, Heidelberg (Apr 1988)
- [13] Davis, H., Günther, F.: Tighter proofs for the SIGMA and TLS 1.3 key exchange protocols. Cryptology ePrint Archive, Report 2020/1029 (2020), <https://eprint.iacr.org/2020/1029>
- [14] 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)
- [15] 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>
- [16] Dodis, Y., Kiltz, E., Pietrzak, K., Wichs, D.: Message authentication, revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 355–374. Springer, Heidelberg (Apr 2012)
- [17] 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)
- [18] 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)
- [19] 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)
- [20] 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)
- [21] 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)
- [22] Gjosteen, 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)
- [23] 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)
- [24] 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)
- [25] Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2021 (2021)
- [26] 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)
- [27] 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)
- [28] 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)
- [29] 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)
- [30] 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)
- [31] 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)

- [32] 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)
- [33] 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)
- [34] Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS. pp. 458–467. IEEE Computer Society Press (Oct 1997)
- [35] 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)

## A Proof of Theorem 3

Let us first define message-consistency for the 2-move protocol  $\text{AKE}_{2\text{msg}}$  in Figure 6.

**Message Consistency.** We say that an oracle  $\pi_i^s$  is *message-consistent* with another oracle  $\pi_j^t$ , denoted by  $\text{MsgCon}(\pi_i^s \leftarrow \pi_j^t)$ , if  $\text{Pid}_i^s := j$  and  $\text{Pid}_j^t := i$  and either

- (1)  $\pi_i^s$  has sent the first message, the same ephemeral public key  $\hat{pk}$  is contained in  $\text{Sent}_i^s$  and  $\text{Recv}_j^t$  and the same ciphertext  $c$  is contained in  $\text{Recv}_i^s$  and  $\text{Sent}_j^t$ , or
- (2)  $\pi_i^s$  has received the first message and the same ephemeral public key  $\hat{pk}$  is contained in  $\text{Recv}_i^s$  and  $\text{Sent}_j^t$ .

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

We now define a sequence of games  $G_0$ - $G_2$ . Let  $\text{Win}_i$  denote the probability that  $G_i$  returns 1.

**Game  $G_0$ :**  $G_0$  is the original experiment  $\text{Exp}_{\text{AKE}_{2\text{msg}}, \mu, \ell, \mathcal{A}}$ . In addition to the original game, we add the sets  $\text{Sent}_i^s$  and  $\text{Recv}_i^s$  which is only a conceptual change. We have

$$\Pr[\text{Exp}_{\text{AKE}_{2\text{msg}}, \mu, \ell, \mathcal{A}} \Rightarrow 1] = \Pr[\text{Win}_0] .$$

**Game  $G_1$ :** In  $G_1$ , we define the event  $\text{NoMsgCon}$  which happens for  $(i, s)$  if  $\pi_i^s$  accepts, the intended partner  $j := \text{Pid}_i^s$  is uncorrupted when  $\pi_i^s$  accepts and there does not exist  $t \in [\ell]$  such that  $\pi_i^s$  is message-consistent with  $\pi_j^t$ . If event  $\text{NoMsgCon}$  happens, the game will abort. Due to the difference lemma,

$$|\Pr[\text{Win}_0] - \Pr[\text{Win}_1]| \leq \Pr[\text{NoMsgCon}] .$$

We will prove the following lemma.

**Lemma 9.** *There exists an adversary  $\mathcal{B}_{\text{SIG}}$  against SIG such that*

$$\Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.1)] \leq \Pr[\text{NoMsgCon}] \leq \text{Adv}_{\text{SIG}, \mu}^{\text{mu-corr}}(\mathcal{B}_{\text{SIG}}) .$$

*Proof.* If there exists an oracle  $\pi_j^t$  such that  $\pi_i^s$  is message-consistent with  $\pi_j^t$  and  $\text{Pid}_j^t = i$ , then due to correctness of KEM,  $\pi_i^s$  is also partnered to  $\pi_j^t$ . It follows that  $\Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.1)] \leq \Pr[\text{NoMsgCon}]$ .

To prove that  $\Pr[\text{NoMsgCon}] \leq \text{Adv}_{\text{SIG}, \mu}^{\text{mu-corr}}(\mathcal{B}_{\text{SIG}})$ , we construct adversary  $\mathcal{B}_{\text{SIG}}$  against MU-EUF-CMA<sup>corr</sup> security of SIG.  $\mathcal{B}_{\text{SIG}}$  inputs the public parameter  $\text{pp}_{\text{SIG}}$  and a list of verification keys  $\{vk_i\}_{i \in [\mu]}$  and has access to a signing oracle  $\mathcal{O}_{\text{SIGN}}(\cdot, \cdot)$  and a corrupt oracle  $\mathcal{O}_{\text{CORR}}(\cdot)$ .  $\mathcal{B}_{\text{SIG}}$  then runs  $\text{pp}_{\text{KEM}} \leftarrow \text{KEM.Setup}$  and sets  $\text{pp}_{\text{AKE}} := (\text{pp}_{\text{SIG}}, \text{pp}_{\text{KEM}})$  and  $\text{PKList} := \{vk_i\}_{i \in [\mu]}$ . It initializes all variables and then runs  $\mathcal{A}$  on  $\text{pp}_{\text{AKE}}$  and  $\text{PKList}$ . If  $\mathcal{A}$  queries  $\mathcal{O}_{\text{AKE}}$ ,  $\mathcal{B}_{\text{SIG}}$  responds as follows.

- $\text{Send}(i, s, j, \text{msg} = \top)$ : In order to get  $\sigma_1$ ,  $\mathcal{B}_{\text{SIG}}$  queries its signing oracle  $\mathcal{O}_{\text{SIGN}}(i, (P_i, P_j, \hat{pk}))$ .
- $\text{Send}(i, s, j, \text{msg} = (pk, \sigma_1))$ : In order to get  $\sigma_2$ ,  $\mathcal{B}_{\text{SIG}}$  queries its signing oracle  $\mathcal{O}_{\text{SIGN}}(i, (P_j, P_i, \hat{pk}, \sigma_1, c))$ .
- $\text{Corrupt}(i)$ :  $\mathcal{B}_{\text{SIG}}$  queries its own oracle  $\mathcal{O}_{\text{CORR}}(i)$  to obtain the signing key  $ssk_i$  and returns  $ssk_i$  to  $\mathcal{A}$ .
- Queries  $\text{Send}(i, s, j, (c, \sigma_2))$ ,  $\text{RegisterCorrupt}$ ,  $\text{SessionKeyReveal}$  and  $\text{Test}$  can be simulated as in  $G_0$ .

During the simulation,  $\mathcal{B}_{\text{SIG}}$  checks if  $\text{NoMsgCon}$  happens. If this is the case, there exists an oracle  $\pi_i^s$  such that  $\pi_i^s$  has accepted and  $j := \text{Pid}_i^s$  is uncorrupted at that point in time.

Now we show that then there is a valid message-signature pair  $(m^*, \sigma^*)$  in  $\text{Sent}_i^s$  and  $\text{Recv}_i^s$  such that  $\text{Ver}(vk_j, m^*, \sigma^*) = 1$  and  $m^*$  is different from any message  $m$  signed by  $\pi_j^t$  for all  $t \in [\ell]$ . Since  $\pi_i^s$  is accepted,  $\text{Sent}_i^s \neq \emptyset$  and  $\text{Recv}_i^s \neq \emptyset$ .

**Case 1:**  $\pi_i^s$  sent the first message. Let  $\text{Sent}_i^s = \{(\hat{pk}, \sigma_1)\}$  and  $\text{Recv}_i^s = \{(c, \sigma_2)\}$ . We have  $\text{Ver}(vk_j, (P_i, P_j, \hat{pk}, \sigma_1, c), \sigma_2) = 1$ , since  $(c, \sigma_2) \in \text{Recv}_i^s$ . For any oracle  $\pi_j^t$  with  $\text{Recv}_j^t = \{(\hat{pk}', \sigma'_1)\} \neq \emptyset$  and  $\text{Sent}_j^t = \{(c', \sigma'_2)\} \neq \emptyset$ ,  $\text{NoMsgCon}$  implies that  $(\hat{pk}, c) \neq (\hat{pk}', c')$ . In this case,  $\mathcal{B}_{\text{SIG}}$  sets  $(m^*, \sigma^*) := ((P_i, P_j, \hat{pk}, \sigma_1, c), \sigma_2)$ .

**Case 2:**  $\pi_i^s$  received the first message. Let  $\text{Recv}_i^s = \{(\hat{pk}, \sigma_1)\}$  and  $\text{Sent}_i^s = \{(c, \sigma_2)\}$ . We have  $\text{Ver}(vk_j, (P_j, P_i, \hat{pk}), \sigma_1) = 1$ , since  $\text{Recv}_i^s \neq \emptyset$ . For any oracle  $\pi_j^t$  with  $\text{Sent}_j^t = \{(\hat{pk}', \sigma'_1)\} \neq \emptyset$ ,  $\text{NoMsgCon}$  implies that  $\hat{pk} \neq \hat{pk}'$ . In this case,  $\mathcal{B}_{\text{SIG}}$  sets  $(m^*, \sigma^*) := ((P_j, P_i, \hat{pk}), \sigma_1)$ .

As soon as event  $\text{NoMsgCon}$  happens,  $\mathcal{B}_{\text{SIG}}$  retrieves the message-signature  $(m^*, \sigma^*)$  pair as just described and outputs  $(j, m^*, \sigma^*)$ . As  $P_j$  is uncorrupted,  $\mathcal{B}_{\text{SIG}}$  has not queried  $\mathcal{O}_{\text{CORR}}(j)$  and  $m^*$  is different from all signing queries for  $j$ , which concludes the proof of Lemma 9.  $\square$

Before moving to  $\mathsf{G}_2$ , let us bound  $(1) \wedge (2) \wedge (3.2)$ .

**Multiple Partners.** Event  $(1) \wedge (2) \wedge (3.2)$  happens if there exists any oracle  $\pi_i^s$  that has accepted with  $\text{Aflag}_i^s = \text{false}$  and has more than one partner oracle. We can show that

$$\Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.2)] \leq (\mu\ell)^2 \cdot 2^{-\gamma}.$$

The session key only depends on the ephemeral public key  $\hat{pk}$  and the ciphertext  $c$ . In the following, we assume that there are two oracles  $\pi_j^t$  and  $\pi_{j'}^{t'}$  such that  $\pi_i^s$  is partnered to both  $\pi_j^t$  and  $\pi_{j'}^{t'}$ . We distinguish two cases:

**Case 1:**  $\pi_i^s$  sent the first message. Let  $\hat{pk}$  be the ephemeral public key determined by the internal randomness of  $\pi_i^s$ . Let  $(c, K) \leftarrow \text{Encap}(\hat{pk}; r)$  and  $(c', K') \leftarrow \text{Encap}(\hat{pk}; r')$ , where  $r, r'$  is the internal randomness of  $\pi_j^t$  and  $\pi_{j'}^{t'}$ , respectively. As  $\pi_i^s$  is partnered to both oracles, this implies that  $k_i^s = \text{Decap}(\hat{sk}, c) = \text{Decap}(\hat{sk}, c')$ . By the correctness and  $\gamma$ -diversity of KEM, we have  $k_i^s = K = K'$  which will happen with probability at most  $2^{-\gamma}$ .

**Case 2:**  $\pi_i^s$  received the first message. Let  $\hat{pk}$  and  $\hat{pk}'$  be the public keys determined by the internal randomness of  $\pi_j^t$  and  $\pi_{j'}^{t'}$ , respectively. Let  $r$  be the internal randomness of  $\pi_i^s$  which is used by  $\text{Encap}$ . The original keys are derived from  $(c, K) \leftarrow \text{Encap}(\hat{pk}; r)$  and  $(c', K') \leftarrow \text{Encap}(\hat{pk}'; r)$ . As  $\pi_i^s$  is partnered to both oracles,  $k_i^s = K = K'$ . Due to  $\gamma$ -diversity of KEM, this will happen only with probability at most  $2^{-\gamma}$ .

As there are  $\mu\ell$  oracles, we can upper bound the probability for event  $(1) \wedge (2) \wedge (3.2)$  by  $(\mu\ell)^2 \cdot 2^{-\gamma}$ .

At this point note that

$$\begin{aligned} \Pr[\text{Win}_{\text{Auth}}] &= \Pr_{\exists(i,s)} [(1) \wedge (2) \wedge ((3.1) \vee (3.2))] \\ &\leq \Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.1)] + \Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.2)] \\ &\leq \text{Adv}_{\text{SIG}, \mu}^{\text{mu-corr}}(\mathcal{B}_{\text{SIG}}) + (\mu\ell)^2 \cdot 2^{-\gamma}. \end{aligned}$$

**Game  $\mathsf{G}_2$ :** In  $\mathsf{G}_2$ , we check the partnership  $\text{Partner}(\pi_i^s \leftarrow \pi_j^t)$  by message-consistency  $\text{MsgCon}(\pi_i^s \leftarrow \pi_j^t)$  if  $\Psi_i^s = \text{accept}$  and  $\text{Aflag}_i^s = \text{false}$ . We claim that

$$|\Pr[\text{Win}_1] - \Pr[\text{Win}_2]| \leq \Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.2)] \leq (\mu\ell)^2 \cdot 2^{-\gamma}.$$

Recall that if  $\text{NoMsgCon}$  does not happen, we know that each oracle  $\pi_i^s$  that has accepted with  $\text{Aflag}_i^s = \text{false}$  is partnered to and message-consistent with an oracle  $\pi_j^t$ . If any such oracle  $\pi_i^s$  has a unique partner, then  $\mathsf{G}_1$  is identical to  $\mathsf{G}_2$ . On the other hand, the probability that there exists an oracle  $\pi_i^s$  that has accepted with  $\text{Aflag}_i^s = \text{false}$  and has multiple partners is  $\Pr_{\exists(i,s)} [(1) \wedge (2) \wedge (3.2)]$ ,

$\mathcal{B}_{\text{KEM}}^{\mathcal{O}_{\text{ENCAP}}(\cdot), \mathcal{O}_{\text{DECAP}}(\cdot, \cdot)}(\text{pp}_{\text{KEM}}, pk_1, \dots, pk_{\mu\ell}) :$ <p> <math>\text{pp}_{\text{SIG}} \leftarrow \text{SIG.Setup}</math>  For <math>i \in [\mu]</math>:  <math>(vk_i, ssk_i) \leftarrow \text{SIG.Gen}(\text{pp}_{\text{SIG}})</math>  <math>crp_i := \text{false}</math>  <math>\text{PKList} := \{vk_i\}_{i \in [\mu]}</math>; <math>b \leftarrow \{0, 1\}</math>  For <math>(i, s) \in [\mu] \times [\ell]</math>:  <math>\text{var}_i^s := (\text{Pid}_i^s, k_i^s, \Psi_i^s) := (\emptyset, \emptyset, \emptyset)</math>  <math>(\text{Sent}_i^s, \text{Recv}_i^s) := (\emptyset, \emptyset)</math>  <math>\text{Aflag}_i^s := \text{false}</math>; <math>T_i^s := \text{false}</math>; <math>kRev_i^s := \text{false}</math>  <math>\text{NoMsgCon} := \text{false}</math>  <math>b^* \leftarrow \mathcal{A}^{\mathcal{O}_{\text{AKE}}(\cdot)}(\text{pp}_{\text{AKE}}, \text{PKList})</math>  If <math>b^* = b</math>: Return <math>\beta^* := 0</math>  Else: Return <math>\beta^* := 1</math> </p> <p>// During the execution <math>\mathcal{B}_{\text{KEM}}</math> checks if the following  // flag is set to true and if so, it aborts immediately:  <math>\text{NoMsgCon} := \text{true}</math>, If <math>\exists i, s \in [\mu] \times [\ell]</math> s.t. <math>(1') \wedge (2') \wedge (3')</math>.  Let <math>j := \text{Pid}_i^s</math>.  <math>(1') \Psi_i^s = \text{accept}</math>  <math>(2') \text{Aflag}_i^s = \text{false}</math>  <math>(3') \nexists t \in [\ell]</math> s.t. <math>\text{MsgCon}(\pi_i^s \leftarrow \pi_j^t)</math></p> <p><math>\mathcal{O}_{\text{AKE}}(\text{query})</math>:  If <math>\text{query} = \text{Test}(i, s)</math>:  If <math>\Psi_i^s \neq \text{accept} \vee \text{Aflag}_i^s = \text{true} \vee kRev_i^s = \text{true}</math>  <math>\vee T_i^s = \text{true}</math>:  Return <math>\perp</math>  Let <math>j := \text{Pid}_i^s</math>  If <math>\exists t \in [\ell]</math> s.t. <math>\text{MsgCon}(\pi_i^s \leftarrow \pi_j^t) \wedge k_j^t = k_i^s</math>:  If <math>kRev_j^t = \text{true} \vee T_j^t = \text{true}</math>: Return <math>\perp</math>  <math>T_i^s := \text{true}</math>  <math>k_0 := k_i^s</math>; <math>k_1 \leftarrow \mathcal{K}</math>  Return <math>k_b</math></p> <p>If <math>\text{query} = \text{SessionKeyReveal}(i, s)</math>:  If <math>\Psi_i^s \neq \text{accept}</math>: Return <math>\perp</math>  If <math>T_i^s = \text{true}</math>: Return <math>\perp</math>  Let <math>j := \text{Pid}_i^s</math>  If <math>\exists t \in [\ell]</math> s.t. <math>T_j^t = \text{true}</math>:  If <math>\text{MsgCon}(\pi_j^t \leftarrow \pi_i^s) \wedge k_j^t = k_i^s</math>: Return <math>\perp</math>  <math>kRev_i^s := \text{true}</math>; Return <math>k_i^s</math></p>	$\mathcal{O}_{\text{AKE}}(\text{query})$ : If $\text{query} = \text{Send}(i, s, j, \text{msg})$ : If $\Psi_i^s = \text{accept}$ : Return $\perp$ If $\text{msg} = \top$ : //session is initiated $\text{Pid}_i^s := j$ Let $n := (i-1)\mu + s$ ; $\hat{pk} := pk_n$ $\sigma_1 \leftarrow \text{Sign}(ssk_i, (P_i, P_j, \hat{pk}))$ $\text{msg}' := (pk, \sigma_1)$ If $\text{msg} = (pk, \sigma_1)$ : //first message $\text{Pid}_i^s := j$ If $\text{Ver}(vk_j, (P_j, P_i, \hat{pk}), \sigma_1) \neq 1$ : $\Psi_i^s := \text{reject}$ ; Return $\perp$ If $crp_j = \text{false}$ : Then $\exists$ unique $t$ s.t. $\hat{pk}$ output by $\pi_j^t$ Let $n := (j-1)\mu + t$ $(c, K_\beta) \leftarrow \mathcal{O}_{\text{ENCAP}}^\beta(n)$ ; $k_i^s := K_\beta$ Else: $(c, K) \leftarrow \text{Encap}(\hat{pk})$ ; $k_i^s := K$ $\sigma_2 \leftarrow \text{Sign}(ssk_i, (P_j, P_i, \hat{pk}, \sigma_1, c))$ $\Psi_i^s := \text{accept}$ $\text{msg}' := (c, \sigma_2)$ If $\text{msg} = (c, \sigma_2)$ : //second message Choose $(pk, \sigma_1) \in \text{Sent}_i^s$ If $\text{Pid} \neq j$ or $\text{Ver}(vk_j, (P_i, P_j, \hat{pk}, \sigma_1, c), \sigma_2) \neq 1$ : $\Psi_i^s := \text{reject}$ ; Return $\perp$ Let $n := (i-1)\mu + s$ and $j := \text{Pid}_i^s$ If $\exists t$ s.t. $\text{Recv}_j^t = \{(pk, \cdot)\} \wedge \text{Sent}_j^t = \{(c, \cdot)\}$ : $k_i^s := k_j^t$ Else: $K \leftarrow \mathcal{O}_{\text{DECAP}}(n, c)$ ; $k_i^s := K$ $\Psi_i^s := \text{accept}$ $\text{msg}' := \emptyset$ $\text{Recv}_i^s := \text{Recv}_i^s \cup \{\text{msg}\}$ ; $\text{Sent}_i^s := \text{Sent}_i^s \cup \{\text{msg}'\}$ If $\Psi_i^s = \text{accept}$ : If $crp_j = \text{true}$ : $\text{Aflag}_i^s := \text{true}$ Return $\text{msg}'$
--	---

**Fig. 13.** Adversary  $\mathcal{B}_{\text{KEM}}$  against MUC-otCCA security of KEM for the proof of Theorem 3. Queries to  $\mathcal{O}_{\text{AKE}}$  where  $\text{query} \in \{\text{Corrupt}, \text{RegisterCorrupt}\}$  are defined as in the original game  $\text{Exp}_{\text{AKE}, \mu, \ell, \mathcal{A}}$  in Figure 5.

which is bounded by  $(\mu\ell)^2 \cdot 2^{-\gamma}$ . Thus, the claims follows by the difference lemma.

In order to bound  $\Pr[\text{Win}_2]$ , we construct an adversary  $\mathcal{B}_{\text{KEM}}$  against MUC-otCCA security of KEM (see Figure 13). We will show that

$$|\Pr[\text{Win}_2] - \frac{1}{2}| \leq 2 \cdot \text{Adv}_{\text{KEM}, \mu\ell}^{\text{muc-otcca}}(\mathcal{B}_{\text{KEM}}).$$

As in the proof of Theorem 2, we make use of the fact that we can not only replace the session key of an oracle when it is tested but of all oracles that are possibly tested. The difference to  $\text{AKE}_{3\text{msg}}$  is that now the adversary can replay the ephemeral public key  $\hat{pk}$  to other oracles, which is why we need multiple  $\mathcal{O}_{\text{ENCAP}}^\beta$  queries.

Let  $\beta$  be the random bit of  $\mathcal{B}_{\text{KEM}}$ 's challenger.  $\mathcal{B}_{\text{KEM}}$  inputs the public parameter  $\text{pp}_{\text{KEM}}$  and  $\{pk_n\}_{n \in [\mu\ell]}$ .  $\mathcal{B}_{\text{KEM}}$  generates the public parameter for SIG and signature key pairs  $(vk_i, sk_i)$  for  $i \in [\mu]$  and sets  $\text{PKList} := \{vk_i\}_{i \in [\mu]}$ . It initializes all variables, chooses a random challenge bit  $b \leftarrow_{\$} \{0, 1\}$  and runs  $\mathcal{A}$ . If  $\mathcal{A}$  makes a query to  $\mathcal{O}_{\text{AKE}}$ ,  $\mathcal{B}_{\text{KEM}}$  simulates the response as follows:

- $\text{Send}(i, s, j, \text{msg} = \top)$ :  $\mathcal{B}_{\text{KEM}}$  uses the public key with index  $(i-1)\mu + s$  as ephemeral public key, i.e.  $\hat{pk} := pk_{(i-1)\mu + s}$ .
- $\text{Send}(i, s, j, \text{msg} = (pk, \sigma_1))$ : If  $P_j$  is uncorrupted, then due to the fact that  $\text{NoMsgCon}$  does not happen, there exists a unique oracle  $\pi_j^t$  such that  $\hat{pk}$  was output by  $\pi_j^t$ . Furthermore,  $n = (j-1)\mu + t$  is the index of that public key. Then  $\mathcal{B}_{\text{KEM}}$  queries  $\mathcal{O}_{\text{ENCAP}}^\beta(n)$ , receives a ciphertext and key  $(c, K_\beta)$  and sets  $k_i^s := K_\beta$ . If  $P_j$  is corrupted,  $\mathcal{B}_{\text{KEM}}$  runs  $\text{Encap}(\hat{pk})$  itself to compute  $(c, K)$ . It also computes a signature  $\sigma_2$  as the protocol specifies and outputs  $(c, \sigma_2)$ .
- $\text{Send}(i, s, j, \text{msg} = (c, \sigma_2))$ : Let  $n = (i-1)\mu + s$ . Then,  $\pi_i^s$  sent  $\hat{pk}_n$ . If there exists an oracle  $\pi_j^t$  that has received  $\hat{pk}_n$  and has sent  $c$ , then  $\mathcal{B}_{\text{KEM}}$  sets  $k_i^s := k_j^t$ . Otherwise,  $\mathcal{B}_{\text{KEM}}$  queries  $\mathcal{O}_{\text{DECAP}}(n, c)$ , receives  $K$  and sets  $k_i^s := K$ .
- $\text{Test}(i, s)$ : After ruling out trivial attacks **TA1**, **TA2** and **TA3**,  $\mathcal{B}_{\text{KEM}}$  checks for trivial attacks **TA4** and **TA5** using message-consistency check  $\text{MsgCon}(\pi_i^s \leftarrow \pi_j^t)$  and tests if  $k_j^t = k_i^s$ . If it does not output  $\perp$ ,  $\mathcal{B}_{\text{KEM}}$  sets  $k_0 = k_i^s$  and  $k_1 \leftarrow_{\$} \mathcal{K}$  and outputs  $k_b$ .
- $\text{SessionKeyReveal}(i, s)$ : After ruling out trivial attack **TA2**,  $\mathcal{B}_{\text{KEM}}$  checks for trivial attack **TA4** by checking if there exists an oracle  $\pi_j^t$  such that  $\pi_j^t$  is tested and  $\text{MsgCon}(\pi_j^t \leftarrow \pi_i^s)$ . If further  $k_j^t = k_i^s$ ,  $\mathcal{B}_{\text{KEM}}$  returns  $\perp$ . Otherwise, it outputs  $k_i^s$ .
- Queries  $\text{Corrupt}$  and  $\text{RegisterCorrupt}$  and can be simulated as in  $\mathcal{G}_2$ .

Finally,  $\mathcal{A}$  outputs  $b^*$  and  $\mathcal{B}_{\text{KEM}}$  outputs  $\beta^* := 0$  if  $b^* = b$  and  $\beta^* := 1$  otherwise.

$\mathcal{B}_{\text{KEM}}$  may query  $\mathcal{O}_{\text{ENCAP}}^\beta$  multiple times on the same ephemeral public key  $\hat{pk}$  as we cannot prevent replay attacks. However, each query to  $\text{Send}(i, s, j, \text{msg} = \top)$  chooses a different ephemeral public key. When the oracle receives a ciphertext, it accepts and thus  $\mathcal{O}_{\text{DECAP}}$  is called at most once. In the following, we will argue that  $\mathcal{B}_{\text{KEM}}$  perfectly simulates  $\mathcal{G}_2$  if  $\beta = 0$ , and that  $\mathcal{A}$ 's view is independent of  $b$  if  $\beta = 1$ . The analysis is the same as in the proof of Theorem 2. We have

$$\begin{aligned} \text{Adv}_{\text{KEM}, \mu\ell}^{\text{muc-otcca}}(\mathcal{B}_{\text{KEM}}) &= |\Pr[\beta^* = \beta] - \frac{1}{2}| \\ &= \left| \frac{1}{2} \cdot \Pr[\beta^* = \beta \mid \beta = 0] + \frac{1}{2} \cdot \Pr[\beta^* = \beta \mid \beta = 1] - \frac{1}{2} \right| \\ &= \left| \frac{1}{2} \cdot \Pr[\text{Win}_2] + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} \right| = \frac{1}{2} |\Pr[\text{Win}_2] - \frac{1}{2}|. \end{aligned}$$

Collecting the probabilities yields the bound in Theorem 3.  $\square$

## B Proof of Lemma 4

We recall random self-reducibility of the MDDH assumption.

For  $Q \in \mathbb{N}$ ,  $\mathbf{W} \leftarrow_{\$} \mathbb{Z}_q^{k \times Q}$ ,  $\mathbf{U} \leftarrow_{\$} \mathbb{Z}_q^{\ell \times Q}$ , we consider the  $Q$ -fold  $\mathcal{D}_{\ell, k}$ -MDDH problem which is to distinguish the distributions  $([\mathbf{A}], [\mathbf{AW}])$  and  $([\mathbf{A}], [\mathbf{U}])$ . Essentially, the  $Q$ -fold  $\mathcal{D}_{\ell, k}$ -MDDH problem contains  $Q$  independent instances of the  $\mathcal{D}_{\ell, k}$ -MDDH problem (with the same  $\mathbf{A}$  but different  $\mathbf{w}_i$ ). The following lemma gives a tight reduction.

**Lemma 10 (Random self-reducibility [17]).** *For  $\ell > k$  and any matrix distribution  $\mathcal{D}_{\ell, k}$ , the  $\mathcal{D}_{\ell, k}$ -MDDH assumption is random self-reducible. In particular, for any  $Q \geq 1$  and any adversary*

There exists an adversary  $\mathcal{B}$  with

$$\begin{aligned} \text{Adv}_{\text{GGen}, \mathcal{D}_{\ell,k}, \mathbb{G}_s}^{Q\text{-MDDH}}(\mathcal{B}) &:= \Pr[\mathcal{B}(\mathcal{P}\mathcal{G}, [\mathbf{A}]_s, [\mathbf{A}\mathbf{W}]_s) \Rightarrow 1] - \Pr[\mathcal{B}(\mathcal{P}\mathcal{G}, [\mathbf{A}]_s, [\mathbf{U}]_s) \Rightarrow 1] \\ &\leq (\ell - k) \text{Adv}_{\text{GGen}, \mathcal{D}_{\ell,k}, \mathbb{G}_s}^{\text{MDDH}}(\mathcal{A}) + \frac{1}{q-1}, \end{aligned}$$

where  $\mathcal{P}\mathcal{G} \leftarrow_s \text{GGen}$ ,  $\mathbf{A} \leftarrow_s \mathcal{D}_{\ell,k}$ ,  $\mathbf{W} \leftarrow_s \mathbb{Z}_q^{k \times Q}$ ,  $\mathbf{U} \leftarrow_s \mathbb{Z}_q^{(k+1) \times Q}$ , and  $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ .

*Proof (Lemma 4).* We prove Lemma 4 by the sequence of games defined in Figure 14. Let  $\mathcal{A}$  be an adversary against the security game  $\text{UF-CMA}^{\text{corr}}$ , and let  $\text{Win}_i$  denote the probability that  $\text{H}_i$  returns 1.

$\boxed{\text{H}_0, \text{H}_1, \text{H}_{2,c}, \text{H}_3}$ $\beta = 0$ $\mathcal{P}\mathcal{G} \leftarrow_s \text{GGen}$ $\mathbf{B} \leftarrow_s \mathcal{U}_{3k,k}$ <p>For <math>1 \leq i \leq \lambda</math> and <math>j = 0, 1</math>:</p> $\mathbf{x}_{i,j} \leftarrow_s \mathbb{Z}_q^{3k}$ $\text{pp}_{\text{MAC}} := (\mathcal{P}\mathcal{G}, [\mathbf{B}]_1, ([\mathbf{B}^\top \mathbf{x}_{i,j}]_1)_{1 \leq i \leq \lambda, j=0,1})$ <p>For <math>1 \leq i \leq \mu</math>:</p> $x'_i \leftarrow_s \mathbb{Z}_q$ $\mathcal{A}^{\mathcal{O}_{\text{MAC}}(\cdot), \mathcal{O}_{\text{VER}}(\cdot, \cdot), \mathcal{O}'_{\text{CORR}}(\cdot)}(\text{pp}_{\text{MAC}})$ <p>Return <math>\beta</math></p> $\mathcal{O}'_{\text{CORR}}(i)$ $\mathcal{L} := \mathcal{L} \cup \{i\}$ <p>Return <math>[x'_i]_1</math></p>	$\mathcal{O}_{\text{MAC}}(i, \text{hm}):$ $\mathcal{Q} := \mathcal{Q} \cup \{(i, \text{hm})\}$ $\mathbf{s} \leftarrow_s \mathbb{Z}_q^k; \mathbf{t} := \mathbf{B}\mathbf{s} \in \mathbb{Z}_q^{3k}; \mathbf{t} \leftarrow_s \mathbb{Z}_q^{3k}$ $u := x'_i + \mathbf{t}^\top \mathbf{x}(\text{hm}) \in \mathbb{Z}_q$ $u := x'_i + \mathbf{t}^\top (\mathbf{B}^\perp \text{RF}_c(\text{hm} _c) + \mathbf{x}(\text{hm}))$ $u \leftarrow_s \mathbb{Z}_q$ <p>Return <math>\sigma := ([\mathbf{t}]_1, [u]_1)</math></p> $\mathcal{O}_{\text{VER}}(i^*, \text{hm}^*, ([\mathbf{t}^*]_1, [u^*]_1)): \text{// at most once}$ <p>If <math>(i^*, \text{hm}^*) \in \mathcal{Q} \vee (i^* \in \mathcal{L})</math>:</p> <p>Return 0</p> $\mathbf{h} := \mathbf{x}(\text{hm}^*)$ $\mathbf{h} := \mathbf{B}^\perp \text{RF}_c(\text{hm}^* _c) + \mathbf{x}(\text{hm}^*)$ $\mathbf{h} := \mathbf{B}^\perp \text{RF}_\lambda(\text{hm}^* _\lambda) + \mathbf{x}(\text{hm}^*)$ <p>If <math>[u^*]_1 = [x'_{i^*}]_1 + [\mathbf{t}^{*\top}]_1 \cdot \mathbf{h}</math>:</p> <p><math>\beta := 1</math></p> <p>Return 1</p> <p>Else: Return 0</p>
--	---

**Fig. 14.** Games  $\text{H}_0, \text{H}_1, \text{H}_{2,c}, \text{H}_3, \text{H}_4$  for proving Lemma 4 where  $0 \leq c \leq \lambda$  and  $\text{RF}_c : \{0, 1\}^c \rightarrow \mathbb{Z}_q^{2k}$  is a random function.

**Game  $\text{H}_0$ :** This is the same game as  $\text{UF-CMA}^{\text{corr}}$ . Thus,

$$\Pr[\text{UF-CMA}^{\text{corr}}_{\mathcal{A}} \Rightarrow 1] = \Pr[\text{Win}_0].$$

**Game  $\text{H}_1$ :** In  $\text{H}_1$ , we switch  $\mathbf{t}$  in  $\mathcal{O}_{\text{MAC}}$  from  $\text{Span}(\mathbf{B})$  to random over  $\mathbb{Z}_q^{3k}$ . By using the  $Q_e$ -fold  $\mathcal{U}_{3k,k}$ -MDDH assumption and Lemma 10, we have

$$|\Pr[\text{Win}_0] - \Pr[\text{Win}_1]| \leq 2k \text{Adv}_{\text{GGen}, \mathcal{U}_{3k,k}, \mathbb{G}_1}^{\text{MDDH}}(\mathcal{B}) + \frac{1}{q-1}.$$

**Game  $\text{H}_{2,c}$  ( $0 \leq c \leq \lambda$ ):** In  $\text{H}_{2,c}$ , we use a random function  $\text{RF}_c : \{0, 1\}^c \rightarrow \mathbb{Z}_q^{2k}$  to randomize  $\mathcal{O}_{\text{MAC}}$  and  $\mathcal{O}_{\text{VER}}$  queries.

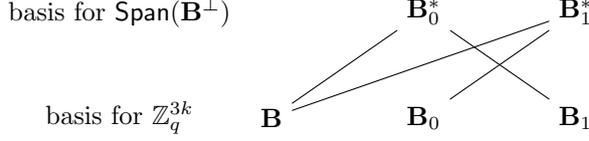
Before we justify the difference, we recall some useful linear algebra facts for the following proofs. For a random matrix  $\mathbf{B} \in \mathbb{Z}_q^{3k \times k}$ , there is a non-zero kernel matrix  $\mathbf{B}^\perp \in \mathbb{Z}_q^{3k \times 2k}$  such that  $\mathbf{B}^\top \mathbf{B}^\perp = \mathbf{0}$ . It is efficient to generate random  $\mathbf{B}_0, \mathbf{B}_1, \mathbf{B}_0^*, \mathbf{B}_1^* \in \mathbb{Z}_q^{3k \times k}$  such that:  $(\mathbf{B} \parallel \mathbf{B}_0 \parallel \mathbf{B}_1)$  is a basis for  $\mathbb{Z}_q^{3k}$ ;  $(\mathbf{B}_0^* \parallel \mathbf{B}_1^*)$  is a basis for  $\text{Span}(\mathbf{B}^\perp)$ ; and  $\mathbf{B}_0^\top \mathbf{B}_1^* = \mathbf{B}_1^\top \mathbf{B}_0^* = \mathbf{0}$ . Figure 15 visualizes these properties.

We show that  $\text{H}_1$  and  $\text{H}_{2,0}$  are identical by viewing  $\mathbf{x}_{1,b}$  as  $\mathbf{x}_{1,b} + \mathbf{B}^\perp \text{RF}_0(\varepsilon)$  (for both  $b = 0, 1$ ) where  $\text{RF}_0(\varepsilon)$  is a fixed random value. Thus,

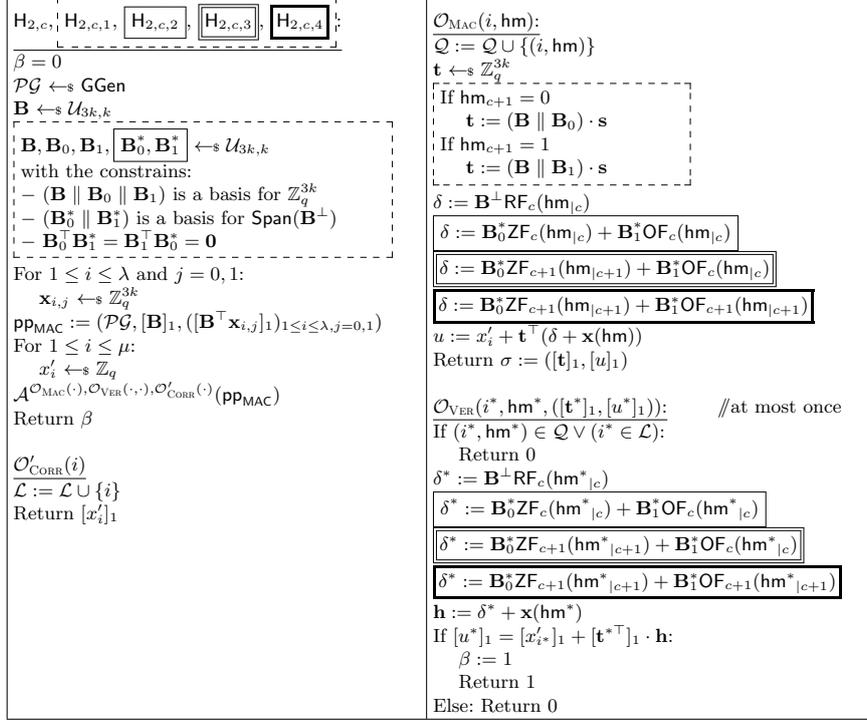
$$\Pr[\text{Win}_1] = \Pr[\text{Win}_{2,0}].$$

To bound the difference between  $\text{H}_{2,c}$  and  $\text{H}_{2,c+1}$  (for  $0 \leq c \leq \lambda - 1$ ), we define a sequence of games in Figure 16.

**Game  $\text{H}_{2,c,1}$ :** In  $\text{H}_{2,c,1}$ , instead of generating a random  $\mathbf{t}$  in  $\mathcal{O}_{\text{MAC}}$ , we choose  $\mathbf{t}$  from  $\text{Span}(\mathbf{B} \parallel \mathbf{B}_0)$  (if  $\text{hm}_{j+1} = 0$ ) or  $\text{Span}(\mathbf{B} \parallel \mathbf{B}_1)$  (if  $\text{hm}_{j+1} = 1$ ). This is the same step as Lemma 17 of [29]. By



**Fig. 15.** Solid lines mean orthogonal:  $\mathbf{B}^\top \mathbf{B}_0^* = \mathbf{B}_1^\top \mathbf{B}_0^* = \mathbf{0} = \mathbf{B}^\top \mathbf{B}_1^* = \mathbf{B}_0^\top \mathbf{B}_1^* \in \mathbb{Z}_q^{k \times k}$ .



**Fig. 16.** Games for bounding the difference between  $\mathsf{H}_{2,c}$  and  $\mathsf{H}_{2,c+1}$  ( $1 \leq c \leq \lambda - 1$ ).

using the  $Q_e$ -fold  $\mathcal{U}_{3k,k}$ -MDDH assumption in  $\mathbb{G}_1$  twice (one with  $[\mathbf{B}_0]_1$  and the other with  $[\mathbf{B}_1]_1$ ) and Lemma 10, we have

$$|\Pr[\text{Win}_{2,c}] - \Pr[\text{Win}_{2,c,1}]| \leq 4k \text{Adv}_{\mathsf{G}\mathsf{Gen}, \mathcal{U}_{3k,k}, \mathbb{G}_1}^{\text{MDDH}}(\mathcal{B}) + \frac{2}{q-1}$$

**Game  $\mathsf{H}_{2,c,2}$ :** Let  $\text{ZF}_c, \text{OF}_c : \{0, 1\}^c \rightarrow \mathbb{Z}_q^k$  be random functions. In  $\mathsf{H}_{2,c,2}$ , we decompose the terms  $\mathbf{B}^\perp \text{RF}_c(\text{hm}_{|c})$  in  $\mathcal{O}_{\text{MAC}}$  as  $\mathbf{B}_0^* \text{ZF}_c(\text{hm}_{|c}) + \mathbf{B}_1^* \text{OF}_c(\text{hm}_{|c})$ . Similarly, we also decompose the terms  $\mathbf{B}^\perp \text{RF}_c(\text{hm}^*_{|c})$  in  $\mathcal{O}_{\text{VER}}$  as  $\mathbf{B}_0^* \text{ZF}_c(\text{hm}^*_{|c}) + \mathbf{B}_1^* \text{OF}_c(\text{hm}^*_{|c})$ . Since  $(\mathbf{B}_0^* \parallel \mathbf{B}_1^*)$  is a basis for  $\text{Span}(\mathbf{B}^\perp)$ , these changes will not modify the distribution, and  $\mathsf{H}_{2,c,1}$  and  $\mathsf{H}_{2,c,2}$  are identical. Thus, we have

$$\Pr[\text{Win}_{2,c,1}] = \Pr[\text{Win}_{2,c,2}].$$

**Game  $\mathsf{H}_{2,c,3}$ :** We define

$$\text{ZF}_{c+1}(\text{hm}_{|c+1}) = \begin{cases} \text{ZF}_c(\text{hm}_{|c}) & (\text{if } \text{hm}_{c+1} = 0) \\ \text{ZF}_c(\text{hm}_{|c}) + \text{ZF}'_c(\text{hm}_{|c}) & (\text{if } \text{hm}_{c+1} = 1) \end{cases}, \quad (10)$$

where  $\text{ZF}'_c : \{0, 1\}^c \rightarrow \mathbb{Z}_q^k$  is another independent random function. Note that  $\text{ZF}_{c+1} : \{0, 1\}^{c+1} \rightarrow \mathbb{Z}_q^k$  is a random function.

In  $\mathsf{H}_{2,c,3}$ , we use this new random function  $\text{ZF}_{c+1}$  to simulate our security game. We observe that:

- In a  $\mathcal{O}_{\text{MAC}}(i, \text{hm})$  query, if  $\text{hm}_{c+1} = 1$ , then  $\mathbf{t} \in \text{Span}(\mathbf{B} \parallel \mathbf{B}_1)$ , and the answer to the query is distributed identically in both  $\mathsf{H}_{2,c,3}$  and  $\mathsf{H}_{2,c,2}$ ; if  $\text{hm}_{c+1} = 0$ , then  $\text{ZF}_{c+1}(\text{hm}_{|c+1}) = \text{ZF}_c(\text{hm}_{|c})$  and its answer is distributed identically in both games.

- In a  $\mathcal{O}_{\text{VER}}(i^*, \text{hm}^*)$  query, if  $\text{hm}_{c+1}^* = 0$ , then the answer is distributed identically in both games. If  $\text{hm}_{c+1}^* = 1$ , we can view  $\mathbf{x}_{c+1,1}$  as  $\mathbf{x}_{c+1,1} + \mathbf{B}_0^* \mathbf{w}$  for  $\mathbf{w} \leftarrow_{\$} \mathbb{Z}_q^k$ . Since  $\mathbf{B}^\top \mathbf{B}_0^* = \mathbf{0}$ ,  $\mathbf{w}$  is perfectly hidden from the term  $[\mathbf{B}^\top \mathbf{x}_{c+1,1}]_1$ . Moreover, in a  $\mathcal{O}_{\text{MAC}}(i, \text{hm})$  query,  $\mathbf{x}_{c+1,1}$  appears if  $\text{hm}_{c+1} = 1$ . But, since  $\mathbf{B}_1^\top \cdot \mathbf{B}_0^* = \mathbf{0}$ ,  $\mathbf{w}$  has never been leaked from  $\mathcal{O}_{\text{MAC}}$  queries. By viewing  $\mathbf{w}$  as  $\text{ZF}_{c+1}(\text{hm}_{c+1}^*)$  (for  $\text{hm}_{c+1}^* = 1$ ), we have the one-time  $\mathcal{O}_{\text{VER}}(i^*, \text{hm}^*)$  query is distributed the same as in both games.

As a result of the above arguments, we have

$$\Pr[\text{Win}_{2,c,2}] = \Pr[\text{Win}_{2,c,3}].$$

**Game  $\text{H}_{2,c,4}$ :** We define

$$\text{OF}_{c+1}(\text{hm}_{c+1}) = \begin{cases} \text{OF}_c(\text{hm}_{c+1}) + \text{OF}'_c(\text{hm}_{c+1}) & (\text{if } \text{hm}_{c+1} = 0) \\ \text{OF}_c(\text{hm}_{c+1}) & (\text{if } \text{hm}_{c+1} = 1) \end{cases}, \quad (11)$$

where  $\text{OF}'_c : \{0, 1\}^c \rightarrow \mathbb{Z}_q^k$  is another independent random function. Note again that  $\text{OF}_{c+1} : \{0, 1\}^{c+1} \rightarrow \mathbb{Z}_q^k$  is a random function.

By a similar argument as in  $\text{H}_{2,c,3}$  (but in a symmetric manner), we can show that

$$\Pr[\text{Win}_{2,c,3}] = \Pr[\text{Win}_{2,c,4}].$$

To bound the difference between  $\text{H}_{2,c,4}$  and  $\text{H}_{2,c+1}$ , we will do the same argument as in  $\text{H}_{2,c,1}$  and  $\text{H}_{2,c,2}$  but in a reverse order. Namely, we first compose  $\mathbf{B}_0^* \text{ZF}_{c+1}(\text{hm}_{c+1}) + \mathbf{B}_1^* \text{OF}_{c+1}(\text{hm}_{c+1})$  to  $\mathbf{B}^\perp \text{RF}_{c+1}(\text{hm}_{c+1})$  for both  $\mathcal{O}_{\text{MAC}}$  and  $\mathcal{O}_{\text{VER}}$  (which is only information-theoretic), and switch  $\mathbf{t}$  in  $\mathcal{O}_{\text{MAC}}$  back to random (which is bounded by using the MDDH assumption). Then we have

$$|\Pr[\text{Win}_{2,c,4}] - \Pr[\text{Win}_{2,c+1}]| \leq 4k \text{Adv}_{\text{GGen}, \mathcal{U}_{3k,k}, \mathbb{G}_1}^{\text{MDDH}}(\mathcal{B}) + \frac{2}{q-1}.$$

Thus, the difference between  $\text{H}_{2,c}$  and  $\text{H}_{2,c+1}$  is bounded by

$$|\Pr[\text{Win}_{2,c}] - \Pr[\text{Win}_{2,c+1}]| \leq 8k \text{Adv}_{\text{GGen}, \mathcal{U}_{3k,k}, \mathbb{G}_1}^{\text{MDDH}}(\mathcal{B}) + \frac{4}{q-1}.$$

**Game  $\text{H}_3$ :** Compared to  $\text{H}_{2,\lambda}$ , the only change  $\text{H}_3$  is to choose  $u$  uniformly at random from  $\mathbb{Z}_q$ . We show that, even with adaptive corruption  $\mathcal{O}'_{\text{CORR}}$  queries, this change does not affect the view of adversary  $\mathcal{A}$ , and  $\text{H}_{2,\lambda}$  is identical to  $\text{H}_3$ .

Our argument is as follows: For a  $\mathcal{O}_{\text{MAC}}(i, \text{hm})$  query in  $\text{H}_{2,\lambda}$ ,  $\mathbf{t}$  is chosen uniformly in  $\mathbb{Z}_q^{3k}$  and thus  $\mathbf{t}^\top \mathbf{B}^\perp \text{RF}_\lambda(\text{hm})$  is a random value in  $\mathbb{Z}_q$  with overwhelming probability  $(1 - 2k/q)$ , even if an (unbounded) adversary corrupts the corresponding  $x'_i$ . Thus, we have

$$|\Pr[\text{Win}_{2,\lambda}] - \Pr[\text{Win}_3]| \leq \frac{2k}{q}.$$

Moreover, in  $\text{H}_3$ , the information about  $x'_{i^*}$  is perfectly hidden from  $\mathcal{A}$ , and thus  $\mathcal{A}$  can compute a  $([\mathbf{t}^*]_1, [u^*]_1)$  such that  $[u^*]_1 = [x'_{i^*}]_1 + [\mathbf{t}^{*\top}]_1 \cdot \mathbf{h}$  with probability  $1/q$ , and we have

$$\Pr[\text{Win}_3] \leq \frac{1}{q}.$$

This completes the proof. □