Anamorphic Authenticated Key Exchange: Double Key Distribution under Surveillance

Weihao Wang[1,2], Shuai Han[1,2](✉) and Shengli Liu[2,3](✉)

[1] School of Cyber Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[3] Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
{dykler123,dalen17,slliu}@sjtu.edu.cn

Abstract. Anamorphic encryptions and anamorphic signatures assume a double key pre-shared between two parties so as to enable the transmission of covert messages. How to securely and efficiently distribute a double key under the dictator's surveillance is a central problem for anamorphic cryptography, especially when the users are forced to surrender their long-term secret keys or even the randomness used in the algorithms to the dictator.

In this paper, we propose Anamorphic Authentication Key Exchange (AM-AKE) to solve the problem. Similar to anamorphic encryption, AM-AKE contains a set of anamorphic algorithms besides the normal algorithms. With the help of the anamorphic algorithms in AM-AKE, the initiator and the responder are able to exchange not only a session key but also a double key. We define robustness and security notions for AM-AKE, and also prove some impossibility results on plain AM-AKE whose anamorphic key generation algorithm only outputs a key-pair. To bypass the impossibility results, we work on two sides.

– On the one side, for plain AM-AKE, the securities have to be relaxed to resist only passive attacks from the dictator. Under this setting, we propose a generic construction of two-pass plain AM-AKE from a two-pass AKE with partially randomness-recoverable algorithms.
– On the other side, we consider (non-plain) AM-AKE whose key generation algorithm also outputs an auxiliary trapdoor besides the key-pairs. We ask new properties from AKE: its key generation algorithm has secret extractability and other algorithms have separability. Based on such a two-pass AKE, we propose a generic construction of two-pass (non-plain) AM-AKE. The resulting AM-AKE enjoys not only robustness but also the strong security against any dictator knowing both users' secret keys and even the internal randomness of the AKE algorithms and implementing active attacks.

Finally, we present concrete AM-AKE schemes from the popular SIG+KEM paradigm and three-KEM paradigm for constructing AKE.

1 Introduction

Cryptography provides fundamental technical tools for achieving authenticity and confidentiality in our daily electronic data communications. For a cryptographic algorithm to work, it is critical that the underlying secret key is not compromised by the adversary. However, an authority dictator may force citizens to surrender their secret keys, and as a result, cryptographic algorithms may completely lose their functionalities of authenticity and confidentiality.

To save cryptographic functionalities in face of dictator, the so-called anamorphic algorithms were introduced [22,15,26,1,5].

Anamorphic Algorithms Supported by Double Key. In [22], Persiano, Phan, and Yung proposed the concept of anamorphic encryption (AME), which is partitioned into receiver-AME and sender-AME depending on whether the receivers are forced to surrender their secret keys or the senders are forced to send designated messages. As for receiver-AME, it is a public-key encryption (PKE) deployed either in normal model with $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ or in the anamorphic mode with $(\mathsf{aGen}, \mathsf{aEnc}, \mathsf{aDec})$. In the anamorphic mode, the receiver initially generates an anamorphic key-pair $(\mathsf{ask}, \mathsf{apk})$ and a *double key* dk via aGen. Any sender who shares dk with the receiver is able to use apk and dk to encrypt not only a normal plaintext m but also a covert plaintext $\hat{m}$. The anamorphic public key apk and the resulting anamorphic ciphertext $\hat{c}$ should be indistinguishable from the normal public key pk and normal ciphertext c to the dictator who also obtains the corresponding secret key. With the knowledge of secret key, the dictator can always decrypt the ciphertext to learn m, but the covert message $\hat{m}$ remains hidden owing to the secrecy of the double key dk.

Later, Kutylowski et al. extended anamorphic encryption to anamorphic signature [15], where a signing party can use the anamorphic signing key ask and the double key dk to send undetectable secure messages using signature tags which are indistinguishable from regular tags for the dictator who sees the signing key ask but not the double key dk.

The original anamorphic schemes bind dk to the anamorphic key pair $(\mathsf{apk}, \mathsf{ask})$. Once the public key is deployed, it is not possible to associate the public key with a double key. This limitation is lifted in [1] by allowing double keys to be created independently of key-pairs, which makes it possible to create double keys at anytime even after the public key is deployed.

Double Key Distribution. The double key dk is essential to anamorphic encryption and anamorphic signature, whose security relies on the secrecy of dk (to the dictator). Now the crucial problem is how to secretly distribute the double key dk between sender and receiver in face of the dictator who may obtain the secret key of all users. The offline physical delivery of dk is expensive and even infeasible in the Internet era. In [22], a two-step bootstrap method in [12] was suggested for distributing dk secretly: Superficially, two parties send to each other abundant ciphertexts generated by a PKE scheme. Covertly, they implement a key-exchange (KE) protocol. Each ciphertext from PKE embeds a tiny piece of the pseudo-random transcript of KE. If they can collect the complete transcript of KE, they can compute a common dk. This method is very inefficient, since its embedding rate is very low. Even worse, this method is too fragile to be practical, since the parties have to collect all these ciphertexts (e.g., hundreds or even thousands of ciphertexts) to recover the KE transcripts, and any active attack or transmission disordering will ruin the distribution of dk. Another possible way for double key distribution might be via sender-AME [26]. However, sender-AME does not allow the dictator to obtain the users' secret keys, which is not compatible to the security settings considered by other anamorphic schemes like receiver-AME or anamorphic signature, and thus this method seems hardly useful for distributing double keys for those anamorphic schemes. Therefore, a natural and important question is:

How to distribute double keys dk *in a secure and efficient way under the surveillance of the dictator?*

Our answer to the question is *Anamorphic Authenticated Key Exchange.*

1.1 Our Contributions

In this paper, we initiate the study of Anamorphic Authenticated Key Exchange (AM-AKE) and formalize security requirements for it, including robustness, indistinguishability of working modes ($\mathsf{IND\text{-}WM}$) and pseudo-randomness of double keys ($\mathsf{PR\text{-}DK}$). Then we provide impossibility results and possibility results on achieving secure AM-AKE. In particular, we show that two popular paradigms for constructing AKE are good candidates for obtaining AM-AKE: the first one is the signed Diffie-Hellman paradigm [19] which uses a digital signature scheme (SIG) and a key encapsulation mechanism (KEM), referred to as the SIG+KEM paradigm in this paper, and the second one is the three-KEM paradigm [20] which invokes KEM three times. Actually, many existing AKE schemes are designed following these paradigms, such as the IKE protocol [11], the protocol used in TLS 1.3 [23], the 2KEM+Diffie-Hellman protocol [4] and more in [13,17,10,21,27,8]. For efficiency consideration, we focus on two-pass AM-AKE.

Syntax, Robustness and Security Notions of AM-AKE. We define two-pass AM-AKE with AKE's normal algorithms and a set of anamorphic algorithms. With the normal algorithms, a session key K is agreed between the initiator and the responder. With the anamorphic ones, both a session key K and a double key dk are agreed between the initiator and the responder.

To make AM-AKE useful, we define initiator-robustness (resp., responder-robustness) as the initiator's (resp., responder's) capability of telling whether its partner is working with the normal or anamorphic algorithms. The robustness helps the party invalidate its double key when its partner works with normal algorithms (i.e., its partner has no intention to share any double key).

As for security, we define indistinguishability between parties' different working modes ($\mathsf{IND\text{-}WM}$ security) against the dictator who possesses the secret keys of all the parties, the session keys, and even the states of the initiator in any completed AKE sessions, and is permitted to conduct active attacks. This ensures that the dictator cannot realize that the parties are actually invoking anamorphic algorithms to establish double keys. For such a dictator, we also define pseudo-randomness of the double keys ($\mathsf{PR\text{-}DK}$ security) to capture that the dictator learns no information about the double keys. This guarantees that the pseudo-random double keys can be later used in anamorphic encryption or signature schemes to transmit covert messages.

We also consider *strong security* notions of $\mathsf{IND\text{-}WM}$ and $\mathsf{PR\text{-}DK}$, denoted by $\mathsf{sIND\text{-}WM}$ and $\mathsf{sPR\text{-}DK}$ respectively. $\mathsf{sIND\text{-}WM}$ and $\mathsf{sPR\text{-}DK}$ are defined similar to $\mathsf{IND\text{-}WM}$ and $\mathsf{PR\text{-}DK}$, but they deal with a stronger dictator who not only implements passive and active attacks, obtains secret keys of the parties, the session keys and the internal states, but also forces the parties to surrender their internal randomness used in AKE sessions.

Impossibility Results for Plain AM-AKE. For a *plain* AM-AKE where the output of the anamorphic key generation algorithm aGen only contains an anamorphic key-pair $(\mathsf{apk}, \mathsf{ask})$, we prove three impossibility results.

- It's impossible for a two-pass plain AM-AKE to achieve responder-robustness.
- It's impossible for a plain AM-AKE to achieve both initiator-robustness and $\mathsf{IND\text{-}WM}$ security, due to the dictator's active attacks of impersonating the initiator with its secret key to test the working mode of its partner.
- It's impossible for a plain AM-AKE to achieve $\mathsf{PR\text{-}DK}$ security under active attacks, since the dictator can impersonate any party with its secret key to agree on a double key.

Generic Construction of Plain AM-AKE with Relaxed Security. To bypass the impossibility results, we relax the security requirements for plain AM-AKE by restricting the active attacks by adversary. The relaxed $\mathsf{IND\text{-}WM}$ security excludes the attacks of impersonating the initiator, while the relaxed $\mathsf{PR\text{-}DK}$ security excludes the attacks of impersonating either the initiator or the responder. Then we propose a generic construction of plain AM-AKE achieving initiator-robustness, the relaxed $\mathsf{IND\text{-}WM}$ and $\mathsf{PR\text{-}DK}$ security from any AKE with partially randomness-recoverable algorithms. We prove that those AKE under the SIG+KEM paradigm and those under the three-KEM paradigm are both good candidates, as long as the underlying SIG and/or KEM schemes are randomness-recoverable.

Generic Construction of Robust AM-AKE with Strong Security. Recall that the impossibility results apply to plain AM-AKE, so another possible way of bypassing the impossibility is designing *non-plain* AM-AKE, where the anamorphic key generation algorithm $(\mathsf{apk}, \mathsf{ask}, \mathsf{aux}) \leftarrow \mathsf{aGen}$ outputs a related auxiliary trapdoor aux along with the anamorphic key-pair $(\mathsf{apk}, \mathsf{ask})$. We note that aux is only kept by the party who generates it and does not need to share it with others. In other words, we do not require the parties pre-share any prior information anyway.

To construct such AM-AKE, we require *secret extractability* for aGen which enables the initiator and the responder to agree on a common secret s computed from the auxiliary trapdoor aux of one party and the public key apk' of the other party. Then we propose a generic construction of AM-AKE achieving the strong $\mathsf{IND\text{-}WM}$ and strong $\mathsf{PR\text{-}DK}$ security from any AKE whose algorithm Gen is secret extractable. We prove that those AKE under the SIG+KEM paradigm and those under the three-KEM paradigm are both good candidates, as long as the underlying SIG and/or KEM schemes have secret extractable key generation algorithm.

1.2 Technique Overview

For a two-pass AKE scheme $\mathsf{AKE} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI})$, the key generation algorithm Gen returns a key-pair $(\mathsf{pk}, \mathsf{sk})$, the initialization algorithm Init computes the first-pass message msg_i, the derivation algorithm DerR for responder derives the second-pass message msg_r and the session key K_r, and the derivation algorithm DerI for initiator derives the session key K_i. For AM-AKE, it additionally has a set of anamorphic algorithms $(\mathsf{aGen}, \mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI})$ for deriving double keys (and session keys as well). Let P_i and P_r denote the initiator and responder respectively.

Impossibility Results for Plain AM-AKE. Roughly speaking, AM-AKE is called a plain one, if the anamorphic key generation algorithm aGen only outputs an anamorphic key-pair $(\mathsf{apk}, \mathsf{ask})$. For a plain AM-AKE, the parties will have no advantage over the dictator who owns their key-pairs as well, leading to the consequence that the dictator can impersonate any party to conduct active attacks. So we have the following impossibility results on plain AM-AKE.

- It's impossible for a two-pass plain AM-AKE to achieve responder-robustness. The responder-robustness means that P_r can decide whether P_i invokes normal algorithms or anamorphic algorithms upon receiving the first-pass message from P_i. Note that the adversary who obtains the secret key of P_r can also make the same judgement, thus distinguishing the working mode of P_i and breaking the security of AM-AKE.
- It's impossible for a plain AM-AKE to achieve both initiator-robustness and $\mathsf{IND\text{-}WM}$ security. With the secret key of P_i, the adversary can impersonate P_i to generate an anamorphic message amsg_i, send it to P_r, and receive a second-pass message from P_r. Note that the initiator-robustness ensures that the adversary who obtains the secret key of P_i can decide whether P_r invokes normal algorithms or anamorphic algorithms, thus breaking the $\mathsf{IND\text{-}WM}$ security of AM-AKE.
- It's impossible for a plain AM-AKE to achieve the $\mathsf{PR\text{-}DK}$ security under active attacks. Similarly, the adversary can impersonate P_i by sending amsg_i to P_r, and compute its double key dk_i upon receiving anamorphic message amsg_r from P_r. Note that the correctness of AM-AKE ensures the consistency of double keys $\mathsf{dk}_i = \mathsf{dk}_r$, and thus the adversary trivially knows P_r's double key dk_r $(= \mathsf{dk}_i)$ and breaks the $\mathsf{PR\text{-}DK}$ security of AM-AKE.

Generic Construction of Plain AM-AKE with Relaxed Security. Let $\mathsf{AKE} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI})$ be a two-pass AKE. Let KE be a two-pass key exchange scheme, like the Diffie-Hellman protocol [6] with the first message g^a and the second message g^b. In the main body of this paper, this KE is accomplished by a KEM scheme with the pseudo-random KEM public key $\widetilde{\mathsf{pk}}$ as the first message and the pseudo-random ciphertext ψ as the second message.

The anamorphic algorithms $(\mathsf{aGen}, \mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI})$ of AM-AKE scheme are almost the same as the normal ones, except that KE's two messages g^a and g^b are used for the (partial) randomnesses to generate AKE's two anamorphic messages $\mathsf{amsg}_i, \mathsf{amsg}_r$:

$$\mathsf{amsg}_i \leftarrow \mathsf{Init}(\underbrace{g^a|\cdots}_{\text{randomness}}), \quad (\mathsf{amsg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}(\mathsf{amsg}_i; \underbrace{g^b|\cdots}_{\text{randomness}}),$$

where the public key and secret key are omitted from the input for simplicity.

If Init and DerR are partially randomness-recoverable, which means there are recovering algorithms for P_i and P_r to recover g^a and g^b from amsg_i and amsg_r, respectively, then the double key $\mathsf{dk} := g^{ab}$ is shared between P_i and P_r.

For passive attacks from the dictator, the uniformity of g^a and g^b guarantees the (statistical) indistinguishability between normal algorithm Init and anamorphic algorithm aInit, and the (statistical) indistinguishability between DerR and aDerR. Meanwhile, the DDH assumption guarantees the pseudo-randomness of dk, even if the dictator obtains both P_i and P_r's secret key and even the underlying randomness $(g^a|\cdots)$ and $(g^b|\cdots)$.

The initiator-robustness can be achieved if we replace randomness $(g^b|\cdots)$ with $(g^b|\sigma := \mathsf{PRF}(g^{ab}, \mathsf{amsg}_i)|\cdots)$ and set $\mathsf{dk} := \mathsf{PRF}(g^{ab}, \mathsf{amsg}_i|\mathsf{amsg}_r)$ with the help of a PRF. In this case P_i is able to tell the working mode of P_r by testing whether $\sigma = \mathsf{PRF}(g^{ab}, \mathsf{amsg}_i)$.

In fact, lots of AKE constructions support the partially randomness-recoverable property. For example, in AKE under the SIG+KEM paradigm [19] and that under the three-KEM paradigm [20], the underlying SIG and KEM have instantiations with randomness-recoverable property [3,2,18,9]. Accordingly, such AKE admits AM-AKE schemes with initiator-robustness and relaxed security.

Generic Construction of Robust AM-AKE with Strong Security. To achieve (strong) $\mathsf{IND\text{-}WM}$ and $\mathsf{PR\text{-}DK}$ security and bypass the impossibility results, we allow the anamorphic key generation algorithm $(\mathsf{apk}, \mathsf{ask}, \mathsf{aux}) \leftarrow \mathsf{aGen}$ of AM-AKE outputs a related auxiliary trapdoor aux along with the anamorphic key-pair $(\mathsf{apk}, \mathsf{ask})$. The auxiliary message aux is only kept privately by the party who generates it and does not need to share it with others.

To construct such AM-AKE, we require new properties for the two-pass AKE scheme $\mathsf{AKE} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI})$, where Gen has *secret extractability*, and Init and DerR have *separable sub-algorithms.*

Roughly speaking, secret extractability of Gen asks a simulating key generation algorithm SimGen and a secret extracting algorithm $\mathsf{Extract}$ satisfying the following properties.

- SimGen outputs not only a key-pair $(\mathsf{pk}, \mathsf{sk})$ that is indistinguishable to the output of Gen, but also a master key msk serving as the auxiliary trapdoor.

- $\mathsf{Extract}(\mathsf{msk}_i, \mathsf{pk}_r) = s = \mathsf{Extract}(\mathsf{msk}_r, \mathsf{pk}_i)$ for all $(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{msk}_i) \leftarrow \mathsf{SimGen}$ and $(\mathsf{pk}_r, \mathsf{sk}_r, \mathsf{msk}_r) \leftarrow \mathsf{SimGen}$. The extracting algorithm can extract a secret s from one party's master key and the other party's public key and make sure that two parties can compute the same secret $s = \mathsf{Extract}(\mathsf{msk}_i, \mathsf{pk}_r) = \mathsf{Extract}(\mathsf{msk}_r, \mathsf{pk}_i)$. The extracted secret s is pseudo-random even in the presence of sk_i and sk_r.

<u>Double Key Generation.</u> Now let SimGen play the role of aGen to generate the anamorphic key-pair $(\mathsf{apk}, \mathsf{ask})$ and the auxiliary trapdoor $\mathsf{aux} := \mathsf{msk}$. Then P_i and P_r use their key-pairs to run the AKE protocol and obtain the two pass messages $(\mathsf{msg}_i, \mathsf{msg}_r)$. At the same time, they can use $\mathsf{Extract}$ to compute a common secret $s = \mathsf{Extract}(\mathsf{msk}_i, \mathsf{pk}_r) = \mathsf{Extract}(\mathsf{msk}_r, \mathsf{pk}_i)$, and then use s as the seed of PRF to compute the double key

$$\mathsf{dk}_i = \mathsf{PRF}(s, (\mathsf{amsg}_i, \mathsf{amsg}_r)) = \mathsf{dk}_r.$$

<u>Achieving Robustness.</u> To achieve robustness, P_i and P_r need to decide the working mode of each other. Our method is that the party invoking anamorphic algorithms provides a proof and embeds the proof in the message, and the other party extracts the proof from the message and verifies the proof. If the proof is valid, then the other party validates its double key and achieves its robustness.

Let us work on responder-robustness first. We require that the normal algorithm Init can be divided into three sub-algorithms $(f_{\mathsf{I},1}, f_{\mathsf{I},2}, \overline{\mathsf{Init}})$ which computes the three parts of $\mathsf{msg}_i = (m_{i,1}, m_{i,2}, m_{i,3})$ respectively. Here $f_{\mathsf{I},1}, f_{\mathsf{I},2}$ make use of independent randomness $d_{i,1}$, $d_{i,2}$ to compute $m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$ and $m_{i,2} := f_{\mathsf{I},2}(d_{i,2})$, and $\overline{\mathsf{Init}}$ uses independently chosen randomness $d_{i,3}$ to compute $m_{i,3} := \overline{\mathsf{Init}}(d_{i,1}, d_{i,2}, d_{i,3})$ together with $d_{i,1}, d_{i,2}$. This is captured by the *3-separability of* Init.

If P_i invokes anamorphic algorithm aInit, then P_i can prove it by embedding the PRF value $\mathsf{PRF}(s, m_{i,1})$ in $d_{i,2}$. Then the anamorphic aInit works as follows.

- $\underline{\mathsf{amsg}_i = (m_{i,1}, m_{i,2}, m_{i,3}) \leftarrow \mathsf{aInit}:}$ $\quad m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$,
  $m_{i,2} := f_{\mathsf{I},2}(d_{i,2} = \mathsf{PRF}(s, m_{i,1}))$, $(m_{i,3}, \mathsf{st}) \leftarrow \overline{\mathsf{Init}}(d_{i,1}, d_{i,2}, d_{i,3})$.

Next, upon receiving amsg_i, P_r can check whether $m_{i,2} = f_{\mathsf{I},2}(\mathsf{PRF}(s, m_{i,1}))$. If yes, P_i must have invoked anamorphic algorithm aInit, and P_r will invoke aDerR to output amsg_r and accept its double key $\mathsf{dk}_r = \mathsf{PRF}(s, (\mathsf{amsg}_i, \mathsf{amsg}_r))$, otherwise invalidate it with $\mathsf{dk}_r := \perp$. Note that in the normal mode, a uniform $d_{i,2}$ hardly collides with $\mathsf{PRF}(s, m_{i,1})$. We further require that $f_{\mathsf{I},2}$ returns different outputs on different inputs, which is captured with entropy-preserving property. Then $m_{i,2} := f_{\mathsf{I},2}(d_{i,2})$ with uniform $d_{i,2}$ hardly collides with $f_{\mathsf{I},2}(\mathsf{PRF}(s, m_{i,1}))$. So P_r can always correctly decide whether P_r invokes normal algorithm Init or anamorphic algorithm aInit, and hence achieve responder-robustness.

In the same way, we can achieve initiator-robustness by requiring that the normal algorithm DerR has 3-separability with sub-algorithms $(f_{\mathsf{R},1}, f_{\mathsf{R},2}, \overline{\mathsf{DerR}})$ computing $\mathsf{msg}_r = (m_{r,1}, m_{r,2}, m_{r,3})$ and $f_{\mathsf{R},2}$ has the property of entropy-preserving. More precisely, if P_r invokes anamorphic algorithm aDerR, then P_r

can prove this fact by embedding the PRF value $\mathsf{PRF}(s,(m_{i,1},m_{r,1}))$ in $d_{r,2}$. Consequently, the anamorphic aDerR works as follows.

- $\underline{\mathsf{amsg}_r = (m_{r,1}, m_{r,2}, m_{r,3}) \leftarrow \mathsf{aDerR}(\mathsf{amsg}_i):}$ $\quad m_{r,1} := f_{\mathsf{R},1}(d_{r,1})$, $m_{r,2} := f_{\mathsf{R},2}(d_{r,2} = \mathsf{PRF}(s,(m_{i,1},m_{r,1})))$, $(m_{r,3},\mathsf{K}_r) \leftarrow \overline{\mathsf{DerR}}(\mathsf{amsg}_i, d_{r,1}, d_{r,2}, d_{r,3})$.

Then upon receiving amsg_r, P_i can check whether $m_{r,2} = f_{\mathsf{R},2}(\mathsf{PRF}(s,(m_{i,1},m_{r,1})))$. If yes, P_r must work in anamorphic mode, and P_i will accept its double key $\mathsf{dk}_i = \mathsf{PRF}(s,(\mathsf{amsg}_i,\mathsf{amsg}_r))$, otherwise invalidate it with $\mathsf{dk}_i := \perp$. Meanwhile, P_i also computes the session key with $\mathsf{K}_i \leftarrow \mathsf{DerI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{amsg}_r, \mathsf{st})$. Here the anamorphic aDerI is exactly the normal DerI. With a similar analysis as above, we have initiator-robustness.

<u>Achieving Strong Security of IND-WM and PR-DK.</u> We note that the dictator does not know the auxiliary trapdoors $\mathsf{msk}_i, \mathsf{msk}_r$, and hence the extracted secret s is pseudo-random even if the dictator obtains the key-pairs $(\mathsf{apk}_i, \mathsf{ask}_i)$ and $(\mathsf{apk}_r, \mathsf{ask}_r)$.

Let us first consider strong IND-WM security. The difference between the normal algorithm Init and the anamorphic aInit lies in that a random $d_{i,2} \leftarrow_\$ \mathcal{D}_{\mathsf{I},2}$ is used in Init while a PRF value $d_{i,2} := \mathsf{PRF}(s, f_{\mathsf{I},1}(d_{i,1}))$ with $d_{i,1} \leftarrow_\$ \mathcal{D}_{\mathsf{I},1}$ is used in aInit.

Now we require $f_{\mathsf{I},1}$ have the property of entropy-preserving, so different inputs to $f_{\mathsf{I},1}$ will lead to different outputs overwhelmingly. Accordingly, every invocation of aInit will result in fresh $d_{i,1}$ and thus fresh $f_{\mathsf{I},1}(d_{i,1})$. Furthermore, the freshness of $f_{\mathsf{I},1}(d_{i,1})$ makes sure that $d_{i,2} := \mathsf{PRF}(s, f_{\mathsf{I},1}(d_{i,1}))$ is pseudo-random and indistinguishable to $d_{i,2} \leftarrow_\$ \mathcal{D}_{\mathsf{I},2}$ used in Init. Therefore, P_i's invoking Init or invoking aInit is indistinguishable to the dictator who knows the secret keys $\mathsf{ask}_i, \mathsf{ask}_r$ and even the randomness $(d_{i,1}, d_{i,2}, d_{i,3})$, and does active attacks with $\mathsf{ask}_i, \mathsf{ask}_r$.

By requiring entropy-preserving property for $f_{\mathsf{R},1}$, we have a similar argument showing that P_r's invoking DerR or invoking aDerR is indistinguishable to the dictator. We stress that the extracted secret s is pseudo-random to the dictator and the dictator's active attacks with message m to aDerR does not help it to distinguish whether $d_{r,2} = \mathsf{PRF}(s, f_{\mathsf{R},1}(d_{r,1}))$ or $d_{r,2} \leftarrow_\$ \mathcal{D}_{\mathsf{R},2}$ due to the freshness of $f_{\mathsf{I},1}(d_{r,1})$ and the security of PRF.

Together with the fact that $\mathsf{DerI} = \mathsf{aDerI}$, we know that the AM-AKE has strong indistinguishability of working mode (strong IND-WM) against the dictator. Here "strong" is reflected in that the dictator is able to implement active attacks with secret keys $\mathsf{ask}_i, \mathsf{ask}_r$ and also able to obtain the randomness like $(d_{i,1}, d_{i,2}, d_{i,3})$ and $(d_{r,1}, d_{r,2}, d_{r,3})$.

As for strong PR-DK security, we first consider the dictator's passive attacks, the pseudo-randomness $\mathsf{dk} = \mathsf{PRF}(s,(\mathsf{amsg}_i,\mathsf{amsg}_r))$ is indistinguishable to a random key $\mathsf{dk} \leftarrow_\$ \mathcal{DK}$, thanks to the freshness of $(\mathsf{amsg}_i,\mathsf{amsg}_r)$ from the entropy-preserving property of $f_{\mathsf{I},1}, f_{\mathsf{I},2}, f_{\mathsf{R},1}, f_{\mathsf{R},2}$. Next we consider the dictator's active attacks with message m. There are two cases.

(1) This m leads to an invalid double key $\mathsf{dk} = \perp$ (but without s, the dictator does not realize $\mathsf{dk} = \perp$) due to $d_{i,2} \neq \mathsf{PRF}(s, m_{i,1})$ or $d_{r,2} \neq \mathsf{PRF}(s,(m_{i,1},m_{r,1}))$.

(2) If $d_{i,2} = \mathsf{PRF}(s, m_{i,1})$, then $\mathsf{dk} = \mathsf{PRF}(s, (m, \mathsf{amsg}_r))$ is a valid one, but is still pseudo-random due to the freshness of amsg_r generated by aDerR. Similarly, if $d_{r,2} = \mathsf{PRF}(s, (m_{i,1}, m_{r,1}))$, then $\mathsf{dk} = \mathsf{PRF}(s, (\mathsf{amsg}_i, m))$ is a valid one, but is still pseudo-random due to the freshness of amsg_i generated by aInit.

Clearly, the pseudo-randomness of valid dk holds even if the dictator additionally knows the randomness like $(d_{i,1}, d_{i,2}, d_{i,3})$ and $(d_{r,1}, d_{r,2}, d_{r,3})$. This yields strong $\mathsf{PR\text{-}DK}$ security.

1.3 Related Works

Anamorphic Cryptography. The notion of anamorphic encryption was proposed in [22]. Later works in [15,1,16,26,5] improved and extended this notion in different aspects. To be specific, more approaches to receiver-AME are provided in [16,1]. The work in [1] decouples the generation of the anamorphic key-pair and the double key, and also proposes the notion of robustness for AME. Sender-AME was considered and specific constructions of robust sender-AME were presented in [26]. In [5], anamorphism is associated to homomorphic encryption, and the double key is dismantled with a public part and a secret part. In [15], anamorphism algorithms were extended to anamorphic signature.

Steganographic Key Exchange. Steganographic key exchange was firstly proposed in [25]. It aims to share a pseudo-random covert key by exchanging a sequence of seemingly normal messages. However, it only considered weak security where the adversary only implements passive attacks. Later, [12] proposed stronger requirement that permits the adversary to obtain the secret keys of parties. Nevertheless, steganographic key exchange does not allow active attacks in the security model, and hence much weaker than the security notions of AM-AKE defined in our paper.

2 Preliminary

Let $\kappa \in \mathbb{N}$ denote the security parameter and let pp denote the public parameter throughout the paper, and all algorithms, distributions, functions and adversaries take 1^κ and pp as implicit inputs. For $N \in \mathbb{N}$, define $[N] = \{1, 2, \ldots, N\}$. If x is defined by y or the value of y is assigned to x, we write $x := y$. For a set $\mathcal{X}$, denote by $|\mathcal{X}|$ the number of elements in $\mathcal{X}$, and denote by $x \leftarrow_\$ \mathcal{X}$ the procedure of sampling x from $\mathcal{X}$ uniformly at random. If $\mathcal{D}$ is distribution, $x \leftarrow_\$ \mathcal{D}$ means that x is sampled according to $\mathcal{D}$. For an algorithm $\mathcal{A}$, let $y \leftarrow \mathcal{A}(x; r)$ or simply $y \leftarrow \mathcal{A}(x)$ denote running $\mathcal{A}$ with input x and randomness r and assigning the output to y. "PPT" abbreviates probabilistic polynomial-time. Denote by poly some polynomial function and negl some negligible function in κ. Let $\perp$ denote the empty string/set, and all variables in our experiments are initialized to $\perp$.

Due to space limitations, we present the definitions of pseudo-random function (PRF), digital signature (SIG) and its $\mathsf{EUF\text{-}CMA}$ security, key encapsulation mechanism (KEM) and its $\mathsf{IND\text{-}CPA}$ security, two-pass authenticated key exchange (AKE) and the DDH assumption in Appendix A.

3 Anamorphic Authenticated Key Exchange

In this section, we present the syntax of anamorphic authenticated key exchange (AM-AKE), propose its robustness requirements, and define its security models. We also establish three impossibility results for plain AM-AKE, and define its relaxed security models.

3.1 Syntax of AM-AKE

Definition 1 ((Plain) Anamorphic Authenticated Key Exchange). *A two-pass authenticated key exchange scheme* $\mathsf{AKE} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI})$ *is called an AM-AKE scheme if there exists a corresponding anamorphic version of algorithms* $(\mathsf{aGen}, \mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI})$ *with syntax defined below.*

- $(\mathsf{apk}, \mathsf{ask}, \mathsf{aux}) \leftarrow \mathsf{aGen}$*: The anamorphic key generation algorithm generates a pair of anamorphic public/secret keys* $(\mathsf{apk}, \mathsf{ask})$ *as well as an auxiliary message* aux *for storing extra secret information.*
- $(\mathsf{amsg}_i, \mathsf{st}, \mathsf{aux}'_i) \leftarrow \mathsf{aInit}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}_i)$*: The anamorphic initialization algorithm takes an anamorphic public key* apk_r *of a responder (say* P_r*), an anamorphic secret key* ask_i *and an initiated auxiliary message* aux_i *of an initiator (say* P_i*) as input, and outputs a message* amsg_i*, a state* st *and an updated auxiliary message* aux'_i *for* P_i*.*
- $(\mathsf{amsg}_r, \mathsf{K}_r, \mathsf{dk}_r) \leftarrow \mathsf{aDerR}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{aux}_r, \mathsf{amsg}_i)$*: The derivation algorithm for the responder takes an anamorphic public key* apk_i *of the initiator* P_i*, an anamorphic secret key* ask_r *and an initiated auxiliary message* aux_r *of the responder* P_r*, and a message* amsg_i *as input, and outputs a message* amsg_r*, a session key* K_r *and a double key* dk_r *for* P_r*.*
- $(\mathsf{K}_i, \mathsf{dk}_i) \leftarrow \mathsf{aDerI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}'_i, \mathsf{amsg}_r, \mathsf{st})$*: The deterministic derivation algorithm for the initiator takes an anamorphic public key* apk_r *of the responder* P_r*, an anamorphic secret key* ask_i *and an updated auxiliary message* aux'_i *of the initiator* P_i*, a message* amsg_r *and a state* st *as input, and outputs a session key* K_i *and a double key* dk_i *for* P_i*.*

Then the AM-AKE scheme is denoted by $\mathsf{AM\text{-}AKE} = ((\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI}), (\mathsf{aGen}, \mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}))$ *where* $(\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI})$ *are called normal algorithms while* $(\mathsf{aGen}, \mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI})$ *are called anamorphic algorithms.*

If $\mathsf{aux} = \perp$ *or* aux *is generated independent of* $(\mathsf{apk}, \mathsf{ask})$ *in* $(\mathsf{apk}, \mathsf{ask}, \mathsf{aux}) \leftarrow \mathsf{aGen}$*, we call* $\mathsf{AM\text{-}AKE}$ *is a plain AM-AKE.*

An execution of an AM-AKE scheme $\mathsf{AM\text{-}AKE}$ is shown in Fig. 1. Any party can choose normal or anamorphic algorithms to run the AKE protocol, resulting in different working modes.

Working Modes of AM-AKE and Correctness Requirements. $\mathsf{AM\text{-}AKE}$ may work in the following three modes.

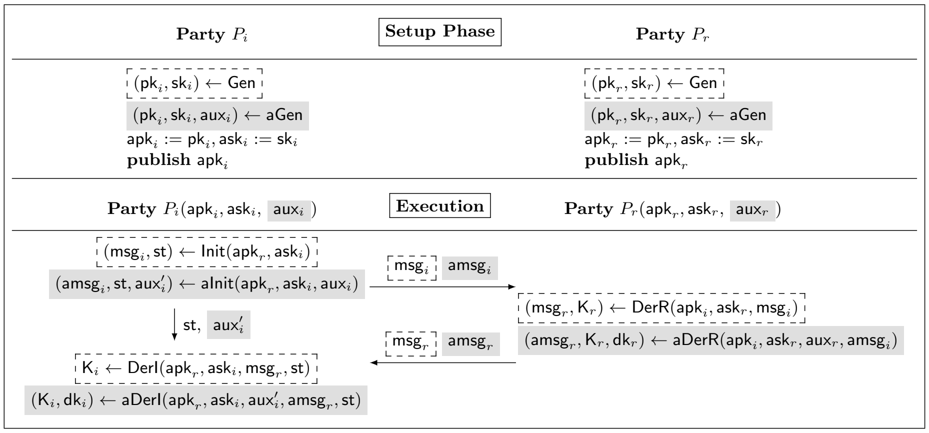

Fig. 1. The normal algorithms (with dotted boxes) and the anamorphic algorithms (with gray boxes) of AM-AKE.

- **Normal Mode.** Both P_i and P_r invoke normal algorithms, i.e., executing the AKE protocol with $(\mathsf{Init}, \mathsf{DerR}, \mathsf{DerI})$. But in the protocol execution, P_i may use either a normal key-pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ generated by Gen or an anamorphic key-pair $(\mathsf{apk}_i, \mathsf{ask}_i)$ generated by aGen, and so does P_r.
- **Anamorphic Mode.** Both P_i and P_r invoke anamorphic algorithms, i.e., executing the protocol with $(\mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI})$, where both P_i and P_r have anamorphic keys $(\mathsf{apk}_i, \mathsf{ask}_i, \mathsf{aux}_i)$, $(\mathsf{apk}_r, \mathsf{ask}_r, \mathsf{aux}_r)$ generated by aGen.
- **Half Mode.** One party invokes normal algorithms while the other invokes anamorphic algorithms. There are two cases described below.
  - **Case I.** P_i invokes anamorphic algorithms $(\mathsf{aInit}, \mathsf{aDerI})$ with its anamorphic keys $(\mathsf{apk}_i, \mathsf{ask}_i, \mathsf{aux}_i)$, while P_r invokes normal algorithm DerR with either normal key-pair $(\mathsf{pk}_r, \mathsf{sk}_r)$ or anamorphic key-pair $(\mathsf{apk}_r, \mathsf{ask}_r)$. In this case, P_i and P_r execute the protocol with $(\mathsf{aInit}, \mathsf{DerR}, \mathsf{aDerI})$.
  - **Case II.** P_i invokes normal algorithms $(\mathsf{Init}, \mathsf{DerI})$ with either normal key-pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ or anamorphic key-pair $(\mathsf{apk}_i, \mathsf{ask}_i)$, while P_r invokes anamorphic algorithm aDerR with its anamorphic keys $(\mathsf{apk}_r, \mathsf{ask}_r, \mathsf{aux}_r)$. In this case, P_i and P_r execute the protocol with $(\mathsf{Init}, \mathsf{aDerR}, \mathsf{DerI})$.

For each of the above three working modes, P_i and P_r should derive the same session key $\mathsf{K}_i = \mathsf{K}_r$. Meanwhile, in the anamorphic mode, they should also derive the same double key $\mathsf{dk}_i = \mathsf{dk}_r$ besides the same session key.

Moreover, AM-AKE always considers adversaries(dictators) who has already obtained secret keys $\mathsf{sk}_i/\mathsf{ask}_i$ and $\mathsf{sk}_r/\mathsf{ask}_r$ from users, the state st from the initiator, and the derived session keys $\mathsf{K}_i, \mathsf{K}_r$ from both initiator and responder. Therefore, an adversary can always invoke DerI to obtain a session key K_i'. To avoid the detection of using anamorphic algorithms in AM-AKE, a basic requirement is that DerI and aDerI results in the same session key $\mathsf{K}_i' = \mathsf{K}_i$. These capture the correctness of AM-AKE. For completeness, we provide the formal definition of correctness in Appendix B.1. We also refer to Table 1 for a summary of correctness requirements in different working modes.

3.2 Robustness of AM-AKE

In practice, it is hard for P_i and P_r to agree on the working mode beforehand. So it happens AM-AKE works in *half mode*: one party invokes normal algorithms while the other invokes anamorphic algorithms. Accordingly, P_i and P_r can hardly agree on consistent double keys, so it is desirable for a party P invoking anamorphic algorithms to detect this issue and invalidate its double key by setting $\mathsf{dk} = \perp$. This is captured by *robustness* of $\mathsf{AM\text{-}AKE}$.

Roughly speaking, *robustness* of $\mathsf{AM\text{-}AKE}$ requires that in the half mode, except for the correctness of $\mathsf{K}_i = \mathsf{K}_r$, the party invoking anamorphic algorithms can detect the half mode of AKE and hence set its double key $\mathsf{dk} := \perp$. According to whether the party is the initiator or the responder, we respectively define *initiator-robustness* and *responder-robustness* as follows.

Definition 2 (Initiator-Robustness). $\mathsf{AM\text{-}AKE}$ *is called initiator-robust, if for any* $(\mathsf{apk}_i, \mathsf{ask}_i, \mathsf{aux}_i) \leftarrow \mathsf{aGen}$*, and for any* $(\overline{pk}_r, \overline{sk}_r) := (\mathsf{pk}_r, \mathsf{sk}_r)$ *generated by* Gen *or* $(\overline{pk}_r, \overline{sk}_r) := (\mathsf{apk}_r, \mathsf{ask}_r)$ *generated by* aGen*, we have*

$$\Pr\left[\mathsf{dk}_i = \perp \middle| \begin{array}{c} (\mathsf{amsg}_i, \mathsf{st}, \mathsf{aux}'_i) \leftarrow \mathsf{aInit}(\overline{pk}_r, \mathsf{ask}_i, \mathsf{aux}_i) \\ (\mathsf{msg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}(\mathsf{apk}_i, \overline{sk}_r, \mathsf{amsg}_i) \\ (\mathsf{K}_i, \mathsf{dk}_i) \leftarrow \mathsf{aDerI}(\overline{pk}_r, \mathsf{ask}_i, \mathsf{aux}'_i, \mathsf{msg}_r, \mathsf{st}) \end{array}\right] \geq 1 - \mathsf{negl}(\kappa).$$

Definition 3 (Responder-Robustness). $\mathsf{AM\text{-}AKE}$ *is called responder-robust, if for any* $(\mathsf{apk}_r, \mathsf{ask}_r, \mathsf{aux}_r) \leftarrow \mathsf{aGen}$*, and for any* $(\overline{pk}_i, \overline{sk}_i) := (\mathsf{pk}_i, \mathsf{sk}_i)$ *generated by* Gen *or* $(\overline{pk}_i, \overline{sk}_i) := (\mathsf{apk}_i, \mathsf{ask}_i)$ *generated by* aGen*, we have*

$$\Pr\left[\mathsf{dk}_r = \perp \middle| \begin{array}{c} (\mathsf{msg}_i, \mathsf{st}) \leftarrow \mathsf{Init}(\mathsf{apk}_r, \overline{sk}_i) \\ (\mathsf{amsg}_r, \mathsf{K}_r, \mathsf{dk}_r) \leftarrow \mathsf{aDerR}(\overline{pk}_i, \mathsf{ask}_r, \mathsf{aux}_r, \mathsf{msg}_i) \\ \mathsf{K}_i \leftarrow \mathsf{DerI}(\mathsf{apk}_r, \overline{sk}_i, \mathsf{amsg}_r, \mathsf{st}) \end{array}\right] \geq 1 - \mathsf{negl}(\kappa).$$

We stress that robustness is important for an AM-AKE scheme, because it's meaningless for a party to derive an un-agreed double key without the other party realizing it. Indeed, using un-agreed double key in the later anamorphic encryption/signature schemes has no effect at all.

For better illustration, we list all working modes of AM-AKE and the corresponding correctness and robustness requirements in Table 1.

3.3 Security Model for AM-AKE

In this subsection, we introduce the security models for AM-AKE. To this end, we need to capture the dictator(government)'s demands and behaviors to formalize the adversary. We consider the setting of multiple parties. In practice, the dictator may force every party involved in AM-AKE to surrender their secret

Working Mode of AM-AKE	Algorithms invoked by P_i	Algorithms invoked by P_r	Correctness	Robustness Init-Rob.	Robustness Resp-Rob.
Normal	Normal	Normal	$\mathsf{K}_i = \mathsf{K}_r$	$-$	$-$
Half	Normal	Anamorphic	$\mathsf{K}_i = \mathsf{K}_r$	$-$	$\mathsf{dk}_r = \perp$
	Anamorphic	Normal	$\mathsf{K}_i = \mathsf{K}_r = \mathsf{K}'_i$	$\mathsf{dk}_i = \perp$	$-$
Anamorphic	Anamorphic	Anamorphic	$\mathsf{K}_i = \mathsf{K}_r = \mathsf{K}'_i \wedge \mathsf{dk}_i = \mathsf{dk}_r$	$-$	$-$

Table 1. Working modes of AM-AKE and the corresponding correctness and robustness requirements. In column **Algorithms invoked by**, it indicates the type of algorithms invoked by P_i and P_r. In column **Correctness**, it shows the correctness requirements, where K_i and dk_i (resp., K_r and dk_r) denote the session key and double key derived by P_i (resp., P_r), and K'_i denotes the session key derived from DerI when P_i invokes anamorphic algorithms. In column **Robustness**, it shows the robustness requirements, where **Init-Rob./Resp-Rob.** denotes Initiator-Robustness/Responder-Robustness and "$-$" means no requirement.

keys, and reveal the session keys along with the state of the initiator in any completed AM-AKE session. Moreover, the dictator may impersonate any party and conduct active attacks because it owns the secret keys of all parties.

Intuitively, the security for AM-AKE requires that such a dictator cannot tell whether AM-AKE is working in the normal mode or in other modes. This is called Indistinguishability of Working Modes ($\mathsf{IND\text{-}WM}$). Moreover, the double keys dk derived from the anamorphic mode will be used later by the anamorphic public-key primitives. To guarantee the security of the anamorphic public-key primitives, we have to require Pseudo-Randomness of Double Keys ($\mathsf{PR\text{-}DK}$).

We also define the corresponding *strong* version of $\mathsf{IND\text{-}WM}$ and $\mathsf{PR\text{-}DK}$ by allowing the dictator additionally receive the internal randomness that all parties used in the seemingly benign AKE sessions, i.e., receiving the true randomness when normal algorithms are invoked while receiving simulated randomness when anamorphic algorithms are used. Especially, we require that there exists PPT simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$, where SimI can explain a randomness R'_i used by aInit as a randomness R_i of Init, and similarly, SimR can explain a randomness R'_r used by aDerR as a randomness R_r of DerR.[4] These result in strong $\mathsf{IND\text{-}WM}$ and strong $\mathsf{PR\text{-}DK}$, denoted by $\mathsf{sIND\text{-}WM}$ and $\mathsf{sPR\text{-}DK}$ respectively.

More precisely, we define the formal security models with the $\mathsf{IND\text{-}WM}/\mathsf{sIND\text{-}WM}$ experiments $\mathsf{Exp}^{\mathsf{IND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},N}/\mathsf{Exp}^{\mathsf{sIND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}$ in Fig. 2 and the $\mathsf{PR\text{-}DK}/\mathsf{sPR\text{-}DK}$ experiments $\mathsf{Exp}^{\mathsf{PR\text{-}DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},N}/\mathsf{Exp}^{\mathsf{sPR\text{-}DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}$ in Fig. 3. To be clearer, we explain the local variables used in these security experiments.

- sID : The identifier of a specific AKE session.
- $\mathrm{init}[\mathsf{sID}]$: The initiator of session sID.
- $\mathrm{resp}[\mathsf{sID}]$: The responder of session sID.
- $\mathrm{mode}_{\mathsf{I}}[\mathsf{sID}]$: The working mode of the initiator in session sID.
- $M^{\mathsf{out}}_{\mathsf{I}}[\mathsf{sID}]/M^{\mathsf{in}}_{\mathsf{I}}[\mathsf{sID}]$: The message sent and received by the initiator in session sID.
- $M^{\mathsf{out}}_{\mathsf{R}}[\mathsf{sID}]/M^{\mathsf{in}}_{\mathsf{R}}[\mathsf{sID}]$: The message sent and received by the responder in session sID.
- $S[\mathsf{sID}]$: The state of the initiator in session sID.

[4] Note that the (anamorphic) derivation algorithms DerI and aDerI for the initiator are typically deterministic without using any randomness.

- Aux[sID]: The *updated* auxiliary message generated by the initiator in session sID.
- K_{I}[sID] (resp., K_{R}[sID]): The session key generated by the initiator (resp., responder) in session sID.
- $DK[\mathsf{sID}, \mathsf{P} \in \{\mathsf{I}, \mathsf{R}\}]$: The double key generated by the initiator when $\mathsf{P} = \mathsf{I}$ or by the responder when $\mathsf{P} = \mathsf{R}$ in session sID.
- $\mathcal{DK}$: The key space of double keys.
- $\mathcal{O}_{\mathsf{New}}(i, r)$: The oracle establishes a new session for initiator P_i and responder P_r.
- $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID})$: The oracle invokes the initialization algorithm for session sID.
- $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$: The oracle invokes the derivation algorithm with input message m for the responder of session sID .
- $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$: The oracle invokes the derivation algorithm with input message m for the initiator of session sID.
- $\mathcal{O}_{\mathsf{TestDK}}(\mathsf{sID}, \mathsf{P} \in \{\mathsf{I}, \mathsf{R}\})$: The oracle provides either the double key generated by P of session sID or a random string. Note that this oracle can be invoked only once for each session to avoid trivial attack.

$\mathsf{Exp}^{\mathsf{IND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},N}$ / $\mathsf{Exp}^{\mathsf{sIND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}$ /
$\mathsf{Exp}^{\mathsf{relaxed\text{-}IND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}$ / $\mathsf{Exp}^{\mathsf{relaxed\text{-}sIND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}$:

$b \leftarrow_{\$} \{0,1\}$
$cnt := 0$ //session counter
for $n \in [N]$: $(\mathsf{apk}_n, \mathsf{ask}_n, \mathsf{aux}_n) \leftarrow \mathsf{aGen}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{WM}}}(\mathsf{apk}_1, \ldots, \mathsf{apk}_N, \mathsf{ask}_1, \ldots, \mathsf{ask}_N)$
return $b' = b$

$\mathcal{O}_{\mathsf{New}}(i, r)$:
if $i \notin [N]$ **or** $r \notin [N]$ **or** $i = r$: **return** $\perp$
$cnt{+}{+}$
$\mathsf{sID} := cnt$
$\mathrm{init}[\mathsf{sID}] := i$; $\mathrm{resp}[\mathsf{sID}] := r$
return sID

$\mathcal{O}_{\mathsf{Init}}(\mathsf{sID}, \mathsf{w_I} \in \{\mathbf{N}, \mathbf{A}\})$:
if $\mathrm{init}[\mathsf{sID}] = \perp$: **return** $\perp$ //session not established
if $M_{\mathsf{I}}^{\mathsf{out}}[\mathsf{sID}] \neq \perp$: **return** $\perp$ //no re-use
$(i, r) := (\mathrm{init}[\mathsf{sID}], \mathrm{resp}[\mathsf{sID}])$
if $b = 0$: //normal working mode
  $(\mathsf{msg}_i, \mathsf{st}) \leftarrow \mathsf{Init}(\mathsf{apk}_r, \mathsf{ask}_i;\ R_i)$
if $b = 1$: //working mode specified by $\mathcal{A}$
  if $\mathsf{w_I} = \mathbf{N}$:
    $(\mathsf{msg}_i, \mathsf{st}) \leftarrow \mathsf{Init}(\mathsf{apk}_r, \mathsf{ask}_i;\ R_i)$
  if $\mathsf{w_I} = \mathbf{A}$:
    $(\mathsf{msg}_i, \mathsf{st}, \mathsf{aux}'_i) \leftarrow \mathsf{aInit}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}_i;\ R'_i)$
    $Aux[\mathsf{sID}] := \mathsf{aux}'_i$
    $R_i \leftarrow \mathsf{SimI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}_i, R'_i)$
$\mathrm{mode_I}[\mathsf{sID}] := \mathsf{w_I}$
$M_{\mathsf{I}}^{\mathsf{out}}[\mathsf{sID}] := \mathsf{msg}_i$
$S[\mathsf{sID}] := \mathsf{st}$
return $(\mathsf{msg}_i, \mathsf{st}, R_i)$

$\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m, \mathsf{w_R} \in \{\mathbf{N}, \mathbf{A}\})$:
if $M_{\mathsf{I}}^{\mathsf{out}}[\mathsf{sID}] = \perp$: **return** $\perp$ //initiator not invoked
if $M_{\mathsf{R}}^{\mathsf{out}}[\mathsf{sID}] \neq \perp$: **return** $\perp$ //no re-use
if $m \neq M_{\mathsf{I}}^{\mathsf{out}}[\mathsf{sID}] \wedge \mathsf{w_R} = \mathbf{A}$: **return** $\perp$ //active attack
$(i, r) := (\mathrm{init}[\mathsf{sID}], \mathrm{resp}[\mathsf{sID}])$
if $b = 0$: //normal working mode
  $(\mathsf{msg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}(\mathsf{apk}_i, \mathsf{ask}_r, m;\ R_r)$
if $b = 1$: //working mode specified by $\mathcal{A}$
  if $\mathsf{w_R} = \mathbf{N}$:
    $(\mathsf{msg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}(\mathsf{apk}_i, \mathsf{ask}_r, m;\ R_r)$
  if $\mathsf{w_R} = \mathbf{A}$:
    $(\mathsf{msg}_r, \mathsf{K}_r, \mathsf{dk}_r) \leftarrow \mathsf{aDerR}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{aux}_r, m;\ R'_r)$
    $R_r \leftarrow \mathsf{SimR}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{aux}_r, m, R'_r)$
$K_{\mathsf{R}}[\mathsf{sID}] := \mathsf{K}_r$
$M_{\mathsf{R}}^{\mathsf{out}}[\mathsf{sID}] := \mathsf{msg}_r$
return $(\mathsf{msg}_r, \mathsf{K}_r, R_r)$

$\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$:
if $M_{\mathsf{R}}^{\mathsf{out}}[\mathsf{sID}] = \perp$: **return** $\perp$ //responder not invoked
if $K_{\mathsf{I}}[\mathsf{sID}] \neq \perp$: **return** $\perp$ //no re-use
$(i, r) := (\mathrm{init}[\mathsf{sID}], \mathrm{resp}[\mathsf{sID}])$
$\mathsf{w_I} := \mathrm{mode_I}[\mathsf{sID}]$
$\mathsf{st} := S[\mathsf{sID}]$
if $b = 0$: //normal working mode
  $\mathsf{K}_i \leftarrow \mathsf{DerI}(\mathsf{apk}_r, \mathsf{ask}_i, m, \mathsf{st})$
if $b = 1$: //working mode specified by $\mathcal{A}$
  if $\mathsf{w_I} = \mathbf{N}$:
    $\mathsf{K}_i \leftarrow \mathsf{DerI}(\mathsf{apk}_r, \mathsf{ask}_i, m, \mathsf{st})$
  if $\mathsf{w_I} = \mathbf{A}$:
    $\mathsf{aux}'_i := Aux[\mathsf{sID}]$
    $(\mathsf{K}_i, \mathsf{dk}_i) \leftarrow \mathsf{aDerI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}'_i, m, \mathsf{st})$
$K_{\mathsf{I}}[\mathsf{sID}] := \mathsf{K}_i$
return K_i

Fig. 2. Security experiments for defining IND-WM (without gray and dotted boxes) and sIND-WM (with gray boxes) of AM-AKE, and experiments for defining relaxed IND-WM (with dotted boxes) and relaxed sIND-WM (with both gray and dotted boxes) of plain AM-AKE, where $\mathcal{O}_{\mathsf{WM}} := \{\mathcal{O}_{\mathsf{New}}, \mathcal{O}_{\mathsf{Init}}, \mathcal{O}_{\mathsf{DerR}}, \mathcal{O}_{\mathsf{DerI}}\}$. Here R_i, R'_i, R_r and R'_r are uniformly sampled from the corresponding randomness spaces.

$\mathsf{Exp}^{\mathsf{PR\text{-}DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},N}$ / $\boxed{\mathsf{Exp}^{\mathsf{sPR\text{-}DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}}$ /

$\boxed{\mathsf{Exp}^{\mathsf{relaxed\text{-}PR\text{-}DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}}$ / $\boxed{\mathsf{Exp}^{\mathsf{relaxed\text{-}sPR\text{-}DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}}$

$b \leftarrow_\$ \{0,1\}$
$cnt := 0$ //session counter
for $n \in [N]$:
  $(\mathsf{apk}_n, \mathsf{ask}_n, \mathsf{aux}_n) \leftarrow \mathsf{aGen}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{PRD}}}(\mathsf{apk}_1, \ldots, \mathsf{apk}_N, \mathsf{ask}_1, \ldots, \mathsf{ask}_N)$
return $b' = b$

$\mathcal{O}_{\mathsf{New}}(i, r)$:
if $i \notin [N]$ **or** $r \notin [N]$ **or** $i = r$:
  return $\perp$
cnt++
$\mathsf{sID} := cnt$
$\mathrm{init}[\mathsf{sID}] := i$
$\mathrm{resp}[\mathsf{sID}] := r$
return sID

$\mathcal{O}_{\mathsf{Init}}(\mathsf{sID})$:
if $\mathrm{init}[\mathsf{sID}] = \perp$: //session not established
  return $\perp$
if $M^{\mathsf{out}}_{\mathsf{I}}[\mathsf{sID}] \neq \perp$: //no re-use
  return $\perp$
$(i, r) := (\mathrm{init}[\mathsf{sID}], \mathrm{resp}[\mathsf{sID}])$
$(\mathsf{amsg}_i, \mathsf{st}, \mathsf{aux}'_i) \leftarrow \mathsf{aInit}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}_i; \boxed{R'_i})$
$\boxed{R_i \leftarrow \mathsf{SimI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}_i, R'_i)}$
$M^{\mathsf{out}}_{\mathsf{I}}[\mathsf{sID}] := \mathsf{amsg}_i$
$S[\mathsf{sID}] := \mathsf{st}$
$Aux[\mathsf{sID}] := \mathsf{aux}'_i$
return $(\mathsf{amsg}_i, \mathsf{st}, \boxed{R_i})$

$\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$:
if $M^{\mathsf{out}}_{\mathsf{I}}[\mathsf{sID}] = \perp$: **return** $\perp$ //initiator not invoked
if $M^{\mathsf{out}}_{\mathsf{R}}[\mathsf{sID}] \neq \perp$: **return** $\perp$ //no re-use
$\boxed{\textbf{if } m \neq M^{\mathsf{out}}_{\mathsf{I}}[\mathsf{sID}] : \textbf{ return } \perp}$ //active attack
$(i, r) := (\mathrm{init}[\mathsf{sID}], \mathrm{resp}[\mathsf{sID}])$
$(\mathsf{amsg}_r, \mathsf{K}_r, \mathsf{dk}_r) \leftarrow \mathsf{aDerR}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{aux}_r, m; \boxed{R'_r})$
$\boxed{R_r \leftarrow \mathsf{SimR}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{aux}_r, m, R'_r)}$
$M^{\mathsf{in}}_{\mathsf{R}}[\mathsf{sID}] := m$; $M^{\mathsf{out}}_{\mathsf{R}}[\mathsf{sID}] := \mathsf{amsg}_r$
$DK[\mathsf{sID}, \mathsf{R}] := \mathsf{dk}_r$
return $(\mathsf{amsg}_r, \mathsf{K}_r, \boxed{R_r})$

$\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$:
if $M^{\mathsf{out}}_{\mathsf{R}}[\mathsf{sID}] = \perp$: **return** $\perp$ //responder not invoked
if $K_{\mathsf{I}}[\mathsf{sID}] \neq \perp$: **return** $\perp$ //no re-use
$\boxed{\textbf{if } m \neq M^{\mathsf{out}}_{\mathsf{R}}[\mathsf{sID}] : \textbf{ return } \perp}$ //active attack
$(i, r) := (\mathrm{init}[\mathsf{sID}], \mathrm{resp}[\mathsf{sID}])$
$\mathsf{st} := S[\mathsf{sID}]$
$\mathsf{aux}'_i := Aux[\mathsf{sID}]$
$(\mathsf{K}_i, \mathsf{dk}_i) \leftarrow \mathsf{aDerI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}'_i, m, \mathsf{st})$
$M^{\mathsf{in}}_{\mathsf{I}}[\mathsf{sID}] := m$
$DK[\mathsf{sID}, \mathsf{I}] := \mathsf{dk}_i$
return K_i

$\mathcal{O}_{\mathsf{TestDK}}(\mathsf{sID}, \mathsf{P} \in \{\mathsf{I}, \mathsf{R}\})$:
if $DK[\mathsf{sID}, \mathsf{P}] = \perp$: **return** $\perp$ //dk not generated or invalid
if $\exists (\mathsf{sID}^*, \overline{\mathsf{P}}) \in \mathfrak{M}[\mathsf{sID}, \mathsf{P}]$ **and** $\mathrm{test}[\mathsf{sID}^*, \overline{\mathsf{P}}] = 1$
  return $\perp$ //trivial attack
$\mathrm{test}[\mathsf{sID}, \mathsf{P}] := 1$
if $b = 1$:
  return $DK[\mathsf{sID}, \mathsf{P}]$
else :
  return $d \leftarrow_\$ \mathcal{DK}$

Fig. 3. Security experiments for defining PR-DK (without gray and dotted boxes) and sPR-DK (with gray boxes) of AM-AKE, and experiments for defining relaxed PR-DK (with dotted boxes) and relaxed sPR-DK (with both gray and dotted boxes) of plain AM-AKE, where $\mathcal{O}_{\mathsf{PRD}} := \{\mathcal{O}_{\mathsf{New}}, \mathcal{O}_{\mathsf{Init}}, \mathcal{O}_{\mathsf{DerR}}, \mathcal{O}_{\mathsf{DerR}}, \mathcal{O}_{\mathsf{TestDK}}\}$. Here R'_i and R'_r are uniformly sampled from the corresponding randomness spaces.

Especially, to formalize the IND-WM/sIND-WM security, we first require that the normal key-pair $(\mathsf{pk}, \mathsf{sk})$ generated by Gen and the anamorphic key-pair $(\mathsf{apk}, \mathsf{ask})$ generated by aGen are computationally indistinguishable, and then we can choose the keys of all parties via aGen. During the experiments (cf. Fig. 2), the adversary is allowed to designate the working modes of the initiator and the responder by providing additionally variables $\mathsf{w}_{\mathsf{I}}, \mathsf{w}_{\mathsf{R}} \in \{\mathbf{N}, \mathbf{A}\}$ to oracles $\mathcal{O}_{\mathsf{Init}}$ and $\mathcal{O}_{\mathsf{DerR}}$, respectively. The adversary is asked to tell whether the oracles run the protocols in the normal modes or in the modes specified by the adversary.

As for the PR-DK/sPR-DK security (cf. Fig. 3), all parties work in the anamorphic modes, and the adversary is asked to distinguish real double keys dk from uniformly chosen keys via a $\mathcal{O}_{\mathsf{TestDK}}$ oracle. To avoid trivial attacks, we define the notion of matching sessions as follows, and we require that the adversary cannot test the double keys of matching sessions.

Definition 4 (Matching Sessions). *For two sessions* $\mathsf{sID}, \mathsf{sID}^*$ *and two parties* $\mathsf{P}, \overline{\mathsf{P}} \in \{\mathsf{I}, \mathsf{R}\}$*, we say* $(\mathsf{sID}, \mathsf{P})$ *and* $(\mathsf{sID}^*, \overline{\mathsf{P}})$ *match, if the same parties are* *involved (i.e.,* $(\mathrm{init}[\mathsf{sID}], \mathrm{resp}[\mathsf{sID}]) = (\mathrm{init}[\mathsf{sID}^*], \mathrm{resp}[\mathsf{sID}^*])$*), the messages sent and received are the same (i.e.,* $(M_{\mathsf{P}}^{\mathsf{in}}[\mathsf{sID}], M_{\mathsf{P}}^{\mathsf{out}}[\mathsf{sID}]) = (M_{\overline{\mathsf{P}}}^{\mathsf{out}}[\mathsf{sID}^*], M_{\overline{\mathsf{P}}}^{\mathsf{in}}[\mathsf{sID}^*])$*), and the parties are of different type (i.e.,* $\overline{\mathsf{P}} = \{\mathsf{I}, \mathsf{R}\} \setminus \mathsf{P}$*). In particular, we define*

$$\mathfrak{M}[\mathsf{sID}, \mathsf{P}] := \left\{ (\mathsf{sID}^*, \overline{\mathsf{P}}) \,\middle|\, \begin{array}{c} (\mathrm{init}[\mathsf{sID}], \mathrm{resp}[\mathsf{sID}]) = (\mathrm{init}[\mathsf{sID}^*], \mathrm{resp}[\mathsf{sID}^*]) \ \wedge \ \overline{\mathsf{P}} = \{\mathsf{I}, \mathsf{R}\} \setminus \mathsf{P} \\ \wedge \ (M_{\mathsf{P}}^{\mathsf{in}}[\mathsf{sID}], M_{\mathsf{P}}^{\mathsf{out}}[\mathsf{sID}]) = (M_{\overline{\mathsf{P}}}^{\mathsf{out}}[\mathsf{sID}^*], M_{\overline{\mathsf{P}}}^{\mathsf{in}}[\mathsf{sID}^*]) \end{array} \right\}$$

as the set of matching sessions with $(\mathsf{sID}, \mathsf{P})$*.*

Now we are ready to present the formal definition of the security of AM-AKE.

Definition 5 (Security of AM-AKE). *The security of* AM-AKE *contains indistinguishability of working modes (*IND-WM*) and pseudo-randomness of double keys (*PR-DK*).*

- **Indistinguishability of Working Modes (IND-WM).** *For any PPT adversary* $\mathcal{A}$ *and any* $N = \mathsf{poly}(\kappa)$*, it holds that* $\big|\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{sk}) = 1] - \Pr[\mathcal{A}(\mathsf{apk}, \mathsf{ask}) = 1]\big| \le \mathsf{negl}(\kappa)$*, where* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$ *and* $(\mathsf{apk}, \mathsf{ask}, \mathsf{aux}) \leftarrow \mathsf{aGen}$*, and*
$$\left|\Pr\left[\mathsf{Exp}_{\mathsf{AM\text{-}AKE},\mathcal{A},N}^{\mathsf{IND\text{-}WM}} = 1\right] - \tfrac{1}{2}\right| \le \mathsf{negl}(\kappa). \tag{1}$$
- **Pseudo-Randomness of Double Keys (PR-DK).** *For any PPT adversary* $\mathcal{A}$ *and any* $N = \mathsf{poly}(\kappa)$*, it holds that*
$$\left|\Pr\left[\mathsf{Exp}_{\mathsf{AM\text{-}AKE},\mathcal{A},N}^{\mathsf{PR\text{-}DK}} = 1\right] - \tfrac{1}{2}\right| \le \mathsf{negl}(\kappa). \tag{2}$$

The strong security of AM-AKE *includes strong indistinguishability of working modes (*sIND-WM*) and strong pseudo-randomness of double keys (*sPR-DK*), which require that there exists PPT simulator* $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$ *such that the above (1) and (2) hold for* $\mathsf{Exp}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}^{\mathsf{sIND\text{-}WM}}$ *and* $\mathsf{Exp}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}^{\mathsf{sPR\text{-}DK}}$ *experiments.*

3.4 Impossibility Results and Relaxed Security for Plain AM-AKE

In this subsection, we show three impossibility results for (two-pass) plain AM-AKE, and then define proper *relaxed* security to circumvent the impossibility results. More precisely, we show the impossibility results via the following three theorems, whose formal proofs are postponed to Appendix B.2, and we refer to Subsect. 1.2 for a high-level overview of the proofs. Roughly speaking, for a (two-pass) plain AM-AKE, the adversary $\mathcal{A}$ holds both $(\mathsf{apk}_i, \mathsf{ask}_i)$ and $(\mathsf{apk}_r, \mathsf{ask}_r)$, and thus a null $\mathsf{aux} = \perp$ or independent aux does not offer any advantage to P_i or P_r over $\mathcal{A}$, and $\mathcal{A}$ is capable of doing whatever P_i or P_r can do.

Theorem 1. *It is impossible for a two-pass plain AM-AKE scheme* AM-AKE *to achieve responder-robustness.*

Theorem 2. *If a plain AM-AKE scheme* AM-AKE *is initiator-robust, then it is impossible for* AM-AKE *to achieve the* IND-WM/sIND-WM *security.*

Theorem 3. *It is impossible for a plain AM-AKE scheme* $\mathsf{AM\text{-}AKE}$ *to achieve the* $\mathsf{PR\text{-}DK}/\mathsf{sPR\text{-}DK}$ *security.*

To circumvent the above impossibility results for plain AM-AKE, we weaken the $\mathsf{IND\text{-}WM}/\mathsf{sIND\text{-}WM}$ security and $\mathsf{PR\text{-}DK}/\mathsf{sPR\text{-}DK}$ security, by restricting the active attacks by adversary. More precisely, we disallow the adversary to query $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m, \mathsf{w_R} = \mathbf{A})$ with its own messages m when $\mathsf{w_R} = \mathbf{A}$ in the $\mathsf{IND\text{-}WM}/\mathsf{sIND\text{-}WM}$ experiments, and disallow the adversary to query $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$ and $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ with its own messages m in the $\mathsf{PR\text{-}DK}/\mathsf{sPR\text{-}DK}$ experiments, respectively. These yield *relaxed* $\mathsf{IND\text{-}WM}/\mathsf{sIND\text{-}WM}$ and *relaxed* $\mathsf{PR\text{-}DK}/\mathsf{sPR\text{-}DK}$ securities, with experiments shown in Fig. 2 and Fig. 3 with dashed boxes.

Definition 6 (Relaxed Security of Plain AM-AKE). *The relaxed security of plain* $\mathsf{AM\text{-}AKE}$ *contains relaxed* $\mathsf{IND\text{-}WM}/\mathsf{sIND\text{-}WM}$ *and relaxed* $\mathsf{PR\text{-}DK}/\mathsf{sPR\text{-}DK}$*, which are defined the same as those (non-relaxed versions) of* $\mathsf{AM\text{-}AKE}$ *in Def. 5, except that the experiments are replaced by the* $\mathsf{Exp}^{\text{relaxed-}\mathsf{IND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}$ / $\mathsf{Exp}^{\text{relaxed-}\mathsf{sIND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}$ / $\mathsf{Exp}^{\text{relaxed-}\mathsf{PR\text{-}DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}$ / $\mathsf{Exp}^{\text{relaxed-}\mathsf{sPR\text{-}DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}$ *in Fig. 2 and Fig. 3, respectively.*

In Appendix E, we present a generic construction of plain AM-AKE with relaxed security, which not only achieves relaxed $\mathsf{sIND\text{-}WM}$ and relaxed $\mathsf{sPR\text{-}DK}$ security, but also enjoys initiator-robustness. We also discuss how to achieve responder-robustness by relying on more passes to evade the first impossibility result. (See Subsect. 1.2 for a high-level overview of this plain AM-AKE construction and its security analysis.) Then in Appendix F, we show how to instantiate the generic construction from the popular SIG+KEM and three-KEM paradigms for constructing AKE and get the corresponding plain AM-AKE schemes.

4 Generic Construction of Robust & Strongly-Secure AM-AKE from AKE

In this section, we present a generic construction of robust and strongly-secure AM-AKE from a basic AKE with the help of a PRF. To make the construction possible, the underlying AKE should be equipped with some new properties, which are defined in Subsect. 4.1. We call such AKE as *qualified AKE*. Then we show the generic construction in Subsect. 4.2 and present its security proof in Subsect. 4.3.

4.1 New Properties for Functions and Algorithms

To characterize the conditions on the basic AKE scheme, in this subsection, we first define three new properties for general functions and algorithms. Roughly speaking, the *entropy-preserving* property of a function asks the function output to have negligible guessing probability on uniformly random input. The η-*separable* property of an algorithm means that the first $\eta - 1$ parts of the output can be computed publicly and in a way independent of the input. The *secret extractability of a key generation algorithm* Gen requires that the key-pair $(\mathsf{pk}, \mathsf{sk})$ from Gen can be perfectly simulated by an algorithm SimGen which additionally outputs a master key msk, and it enables the extraction of a pseudo-random secret s from msk and pk' of another party via an algorithm $\mathsf{Extract}$.

Definition 7 (Entropy-Preserving Function). *A function $f : \mathcal{X} \to \mathcal{Y}$ is entropy-preserving, if for any $y \in \mathcal{Y}$, it holds* $\Pr[f(x) = y \,|\, x \leftarrow_{\$} \mathcal{X}] \leq \mathsf{negl}(\kappa)$.

Definition 8 (η-Separable Algorithm). *Let $\eta \in \mathbb{N}$, and let $(y, z) \leftarrow \mathsf{Alg}(x)$ be a PPT algorithm which inputs x and outputs (y, z). We say that Alg is η-separable for generating y if Alg can be implemented with $(f_1, \ldots, f_{\eta-1}, \overline{\mathsf{Alg}})$ as follows, where $f_j : \mathcal{D}_j \to \{0,1\}^*$ is a publicly and efficiently computable function for $j \in [\eta - 1]$, and $\overline{\mathsf{Alg}}$ is a PPT algorithm.*

- $(y, z) \leftarrow \mathsf{Alg}(x)$**:** *For $j \in [\eta - 1]$, sample $d_j \leftarrow_{\$} \mathcal{D}_j$ and compute $m_j := f_j(d_j)$; invoke $(m_\eta, z) \leftarrow \overline{\mathsf{Alg}}(x, d_1, \ldots, d_{\eta-1})$; output $y := (m_1, \ldots, m_\eta)$ and z.*

Definition 9 (Secret Extractability of Gen). *Let Gen be a key generation algorithm that outputs $(\mathsf{pk}, \mathsf{sk})$.*[5] *We say Gen supports secret extractability if there exist two PPT algorithms SimGen and $\mathsf{Extract}$ satisfying the following properties.*

- $(\mathsf{pk}, \mathsf{sk}, \mathsf{msk}) \leftarrow \mathsf{SimGen}$: *it is a simulated key generation algorithm that outputs a simulated key-pair $(\mathsf{pk}, \mathsf{sk})$ together with a master key msk.*
- $s \leftarrow \mathsf{Extract}(\mathsf{msk}_i, \mathsf{pk}_r)$: *it is a deterministic extracting algorithm that takes a master key msk_i and a public key pk_r as input, and outputs a secret $s \in \mathcal{D}_{\mathsf{E}}$.*

Identically Distributed Key-Pairs. *The simulated key-pair has the same distribution as the normal pair, i.e., the following two distributions are identical:*

$$\{(\mathsf{pk}, \mathsf{sk}) \mid (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}\} \quad \equiv \quad \{(\mathsf{pk}, \mathsf{sk}) \mid (\mathsf{pk}, \mathsf{sk}, \mathsf{msk}) \leftarrow \mathsf{SimGen}\}.$$

Extracting Correctness. *For any $(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{msk}_i) \leftarrow \mathsf{SimGen}$ and $(\mathsf{pk}_r, \mathsf{sk}_r, \mathsf{msk}_r) \leftarrow \mathsf{SimGen}$, it holds that $\mathsf{Extract}(\mathsf{msk}_i, \mathsf{pk}_r) = \mathsf{Extract}(\mathsf{msk}_r, \mathsf{pk}_i)$.*

Pseudo-Randomness of the Extracting. *For any PPT adversary $\mathcal{A}$, we have*

$$\mathsf{Adv}^{\mathsf{PR\text{-}Ext}}_{\mathsf{Gen},\mathcal{A}}(\kappa) := \left| \Pr\left[\mathcal{A}(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{sk}_i, \mathsf{sk}_r, s_0) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{sk}_i, \mathsf{sk}_r, s_1) = 1\right] \right| \leq \mathsf{negl}(\kappa),$$

where $(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{msk}_i) \leftarrow \mathsf{SimGen}$, $(\mathsf{pk}_r, \mathsf{sk}_r, \mathsf{msk}_r) \leftarrow \mathsf{SimGen}$, $s_0 := \mathsf{Extract}(\mathsf{msk}_i, \mathsf{pk}_r)$, and $s_1 \leftarrow_{\$} \mathcal{D}_{\mathsf{E}}$.

Based on the three new properties, we are ready to describe the requirements on the basic AKE and present the generic construction of AM-AKE from it.

[5] Gen can be the key generation algorithm of any public-key primitive, like AKE, SIG, KEM, etc.

4.2 Construction of AM-AKE from AKE and PRF

Let $\mathsf{AKE} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI})$ be a two-pass AKE scheme that satisfies:

- Gen has *secret extractability*, supported by algorithms $(\mathsf{SimGen}, \mathsf{Extract})$ and secret space $\mathcal{D}_\mathsf{E}$ as per Def. 9;
- Init is 3-*separable for generating* msg_i, supported by $(f_{\mathsf{I},1}, f_{\mathsf{I},2}, \overline{\mathsf{Init}})$ as per Def. 8, i.e., $\mathsf{Init}(\mathsf{pk}_r, \mathsf{sk}_i)$ generates $(\mathsf{msg}_i, \mathsf{st})$ by sampling $d_{i,1} \leftarrow_\$ \mathcal{D}_{\mathsf{I},1}$, $d_{i,2} \leftarrow_\$ \mathcal{D}_{\mathsf{I},2}$, computing $m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$, $m_{i,2} := f_{\mathsf{I},2}(d_{i,2})$, invoking $(m_{i,3}, \mathsf{st}) \leftarrow \overline{\mathsf{Init}}(\mathsf{pk}_r, \mathsf{sk}_i, d_{i,1}, d_{i,2})$, and setting $\mathsf{msg}_i := (m_{i,1}, m_{i,2}, m_{i,3})$;
- DerR is 3-*separable for generating* msg_r, supported by $(f_{\mathsf{R},1}, f_{\mathsf{R},2}, \overline{\mathsf{DerR}})$ as per Def. 8, i.e., $\mathsf{DerR}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i)$ generates $(\mathsf{msg}_r, \mathsf{K}_r)$ by sampling $d_{r,1} \leftarrow_\$ \mathcal{D}_{\mathsf{R},1}$,$d_{r,2} \leftarrow_\$ \mathcal{D}_{\mathsf{R},2}$, computing $m_{r,1} := f_{\mathsf{R},1}(d_{r,1})$, $m_{r,2} := f_{\mathsf{R},2}(d_{r,2})$, invoking $(m_{r,3}, \mathsf{K}_r) \leftarrow \overline{\mathsf{DerR}}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i, d_{r,1}, d_{r,2})$, and setting $\mathsf{msg}_r := (m_{r,1}, m_{r,2}, m_{r,3})$;
- The functions $f_{\mathsf{I},1}, f_{\mathsf{I},2}, f_{\mathsf{R},1}, f_{\mathsf{R},2}$ associated with Init and DerR are *entropy-preserving* as per Def. 7.

We call such AKE as *qualified AKE*, with requirements summarized in Table 2. Moreover, let $\mathsf{PRF} : \mathcal{D}_\mathsf{E} \times \{0,1\}^* \longrightarrow \mathcal{D}_{\mathsf{I},2} \times \mathcal{D}_{\mathsf{R},2} \times \{0,1\}^\kappa$ be a pseudo-random function. For ease of exposition, we parse the output of PRF as three parts, i.e., $\mathsf{PRF_I}/\mathsf{PRF_R}/\mathsf{PRF_D} : \mathcal{D}_\mathsf{E} \times \{0,1\}^* \longrightarrow \mathcal{D}_{\mathsf{I},2}/\mathcal{D}_{\mathsf{R},2}/\{0,1\}^\kappa$, such that $\mathsf{PRF}(s, m) = (\mathsf{PRF_I}(s, m), \mathsf{PRF_R}(s, m), \mathsf{PRF_D}(s, m))$ for all $s \in \mathcal{D}_\mathsf{E}$, $m \in \{0,1\}^*$.

Qualified AKE	Gen	Init	DerR
Requirements	*secret extractability*	*3-separable for* msg_i with *entropy-preserving* $f_{\mathsf{I},1}, f_{\mathsf{I},2}$	*3-separable for* msg_r with *entropy-preserving* $f_{\mathsf{R},1}, f_{\mathsf{R},2}$
Supportive Func./Alg.	$(\mathsf{SimGen}, \mathsf{Extract})$	$(f_{\mathsf{I},1}, f_{\mathsf{I},2}, \overline{\mathsf{Init}})$	$(f_{\mathsf{R},1}, f_{\mathsf{R},2}, \overline{\mathsf{DerR}})$

Table 2. Requirements for $\mathsf{AKE} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI})$ to be *qualified* for constructing AM-AKE.

Now we convert AKE to an AM-AKE scheme $\mathsf{AM\text{-}AKE} = ((\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI}), (\mathsf{aGen}, \mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}))$ with the help of PRF, where the anamorphic algorithms are described below. (See also Fig. 4 for an illustration of $\mathsf{AM\text{-}AKE}$.)

- $\underline{(\mathsf{apk}, \mathsf{ask}, \mathsf{aux}) \leftarrow \mathsf{aGen}:}$ it invokes the simulated key generation algorithm $(\mathsf{pk}, \mathsf{sk}, \mathsf{msk}) \leftarrow \mathsf{SimGen}$, and sets $(\mathsf{apk}, \mathsf{ask}) := (\mathsf{pk}, \mathsf{sk})$ and $\mathsf{aux} := \mathsf{msk}$.
- $\underline{(\mathsf{amsg}_i, \mathsf{st}, \mathsf{aux}'_i) \leftarrow \mathsf{aInit}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}_i = \mathsf{msk}_i):}$ it first extracts a secret $s_i := \mathsf{Extract}(\mathsf{msk}_i, \mathsf{apk}_r)$. Next it randomly chooses $d_{i,1} \leftarrow_\$ \mathcal{D}_{\mathsf{I},1}$ and computes $m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$. Then it computes $d_{i,2} := \mathsf{PRF_I}(s_i, m_{i,1}) \in \mathcal{D}_{\mathsf{I},2}$, $m_{i,2} := f_{\mathsf{I},2}(d_{i,2})$, and invokes $(m_{i,3}, \mathsf{st}) \leftarrow \overline{\mathsf{Init}}(\mathsf{apk}_r, \mathsf{ask}_i, d_{i,1}, d_{i,2})$. Finally, it returns $(\mathsf{amsg}_i := (m_{i,1}, m_{i,2}, m_{i,3}), \mathsf{st}, \mathsf{aux}'_i := (s_i, \mathsf{amsg}_i))$.
- $\underline{(\mathsf{amsg}_r, \mathsf{K}_r, \mathsf{dk}_r) \leftarrow \mathsf{aDerR}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{aux}_r = \mathsf{msk}_r, \mathsf{amsg}_i = (m_{i,1}, m_{i,2}, m_{i,3})):}$ it first randomly chooses $d_{r,1} \leftarrow_\$ \mathcal{D}_{\mathsf{R},1}$ and computes $m_{r,1} := f_{\mathsf{R},1}(d_{r,1})$. Next it extracts a secret $s_r := \mathsf{Extract}(\mathsf{msk}_r, \mathsf{apk}_i)$, and computes $d_{r,2} := \mathsf{PRF_R}(s_r, (m_{i,1}, m_{r,1})) \in \mathcal{D}_{\mathsf{R},2}$, $m_{r,2} := f_{\mathsf{R},2}(d_{r,2})$, invokes $(m_{r,3}, \mathsf{K}_r) \leftarrow \overline{\mathsf{DerR}}$

$(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{amsg}_i, d_{r,1}, d_{r,2})$, and sets $\mathsf{amsg}_r := (m_{r,1}, m_{r,2}, m_{r,3})$. Afterwards, it checks whether $m_{i,2} = f_{\mathsf{I},2}(\mathsf{PRF}_\mathsf{I}(s_r, m_{i,1}))$ holds. If the check passes, then it sets $\mathsf{dk}_r := \mathsf{PRF}_\mathsf{D}(s_r, (\mathsf{amsg}_i, \mathsf{amsg}_r)) \in \{0,1\}^\kappa$ as the double key; otherwise, it sets $\mathsf{dk}_r := \perp$. Finally, it returns $(\mathsf{amsg}_r, \mathsf{K}_r, \mathsf{dk}_r)$.

- $\underline{(\mathsf{K}_i, \mathsf{dk}_i) \leftarrow \mathsf{aDerI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}'_i = (s_i, \mathsf{amsg}_i), \mathsf{amsg}_r = (m_{r,1}, m_{r,2}, m_{r,3}), \mathsf{st})}$: it first checks whether $m_{r,2} = f_{\mathsf{R},2}(\mathsf{PRF}_\mathsf{R}(s_i, (m_{i,1}, m_{r,1})))$ holds. If yes, it sets $\mathsf{dk}_i := \mathsf{PRF}_\mathsf{D}(s_i, (\mathsf{amsg}_i, \mathsf{amsg}_r)) \in \{0,1\}^\kappa$ as the double key; else, $\mathsf{dk}_i := \perp$. Finally, it invokes $\mathsf{K}_i \leftarrow \mathsf{DerI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{amsg}_r, \mathsf{st})$, and returns $(\mathsf{K}_i, \mathsf{dk}_i)$.

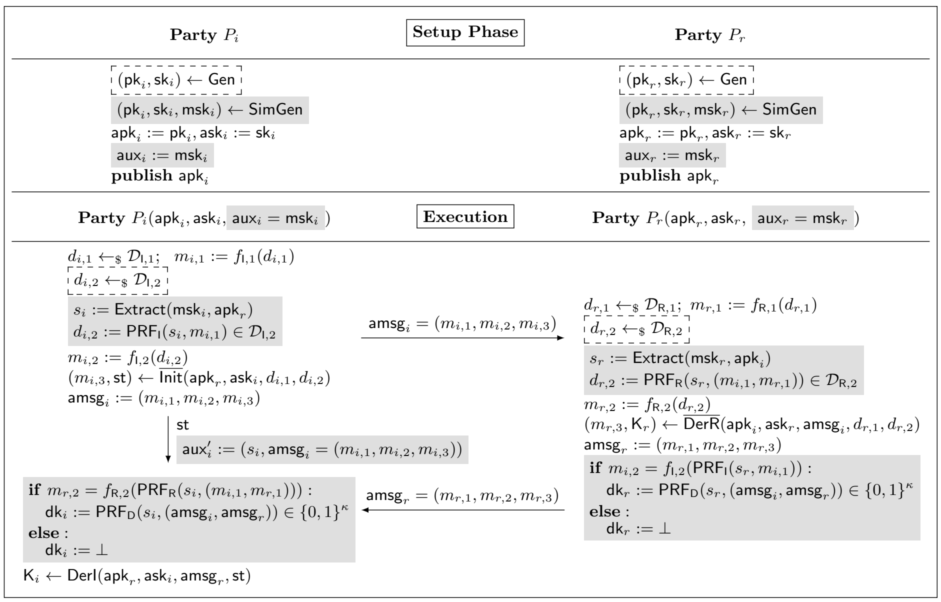

Fig. 4. Generic construction of the AM-AKE scheme AM-AKE based on AKE and PRF, where dotted boxes appear only in normal algorithms (Gen, Init, DerR, DerI), and gray boxes appear only in anamorphic algorithms (aGen, aInit, aDerR, aDerI).

Let us compare the normal algorithms and the anamorphic ones.

- The anamorphic algorithm aGen invokes SimGen to produce a simulated key-pair $(\mathsf{apk}, \mathsf{ask}) := (\mathsf{pk}, \mathsf{sk})$ as well as a master secret $\mathsf{aux} := \mathsf{msk}$. By the property of secret extractability of the normal algorithm Gen, the anamorphic key-pair has the same distribution as the normal key-pair generated by Gen.
- The normal algorithm Init makes use of random coins $d_{i,1}$ and $d_{i,2}$ for the generation of msg_i. The anamorphic algorithm aInit can be regarded as the normal Init taking random coins $d_{i,1}$ and specific coins $d_{i,2} = \mathsf{PRF}_\mathsf{I}(s_i, m_{i,1})$, with s_i a secret extracted from the master secret msk_i of P_i and apk_r of P_r.
- The normal algorithm DerR makes use of random coins $d_{r,1}, d_{r,2}$ for the generation of msg_r and the session key K_r. The anamorphic algorithm aDerR

has two parts: one part can be regarded as the normal DerR taking random coins $d_{r,1}$ and specific coins $d_{r,2} = \mathsf{PRF_R}(s_r, (m_{i,1}, m_{r,1}))$ to output msg_r and key K_r; the other part is in charge of generating the double key $\mathsf{dk}_r := \mathsf{PRF_D}(s_r, (\mathsf{amsg}_i, \mathsf{amsg}_r))$ or $\mathsf{dk}_r := \perp$ depending on whether $m_{i,2} = f_{\mathsf{I},2}(\mathsf{PRF_I}(s_r, m_{i,1}))$ holds, with s_r a secret derived from the master secret msk_r of P_r and apk_i of P_i.

– The normal algorithm DerI is deterministic and outputs the session key K_i. The anamorphic algorithm aDerI functions identically as DerI for the generation of key K_i, but it is also in charge of generating the double key $\mathsf{dk}_i := \mathsf{PRF_D}(s_i, (\mathsf{amsg}_i, \mathsf{amsg}_r))$ or $\mathsf{dk}_i := \perp$ depending on whether $m_{r,2} = f_{\mathsf{R},2}(\mathsf{PRF_R}(s_i, (m_{i,1}, m_{r,1})))$ holds.

Note that the correctness of the underlying AKE guarantees that $\mathsf{K}_i = \mathsf{K}_r$ for every possible choices of $d_{i,1}, d_{i,2}, d_{r,1}, d_{r,2}$. Thus even using specific coins in the anamorphic algorithms, we also have $\mathsf{K}_i = \mathsf{K}_r$. This shows the correctness of $\mathsf{K}_i = \mathsf{K}_r$ in all working modes. Moreover, in the anamorphic mode, we have $\mathsf{dk}_i = \mathsf{PRF_D}(s_i, (\mathsf{amsg}_i, \mathsf{amsg}_r)) = \mathsf{PRF_D}(s_r, (\mathsf{amsg}_i, \mathsf{amsg}_r)) = \mathsf{dk}_r$ since $s_i = \mathsf{Extract}(\mathsf{msk}_i, \mathsf{apk}_r) = \mathsf{Extract}(\mathsf{msk}_r, \mathsf{apk}_i) = s_r$ holds by the extracting correctness of Gen's secret extractability, and thus the correctness of double key holds.

Below we analyze the robustness of our $\mathsf{AM\text{-}AKE}$.

Initiator-Robustness. Suppose that P_i invokes anamorphic algorithms aInit and aDerI while P_r invokes normal algorithm DerR, then P_r computes $m_{r,2} := f_{\mathsf{R},2}(d_{r,2})$ by using a uniformly chosen $d_{r,2} \leftarrow_\$ \mathcal{D}_{\mathsf{R},2}$. When P_i invokes the anamorphic algorithm aDerI to check whether $m_{r,2} = f_{\mathsf{R},2}(\mathsf{PRF_R}(s_i, (m_{i,1}, m_{r,1})))$ holds, we know that here $f_{\mathsf{R},2}(\mathsf{PRF_R}(s_i, (m_{i,1}, m_{r,1})))$ is independent of $m_{r,2} := f_{\mathsf{R},2}(d_{r,2})$ since $d_{r,2} \leftarrow_\$ \mathcal{D}_{\mathsf{R},2}$ is chosen independently of $s_i, m_{i,1}, m_{r,1}$ by P_r. Thus for every possible value of $f_{\mathsf{R},2}(\mathsf{PRF_R}(s_i, (m_{i,1}, m_{r,1})))$, the check $m_{r,2} := f_{\mathsf{R},2}(d_{r,2}) = f_{\mathsf{R},2}(\mathsf{PRF_R}(s_i, (m_{i,1}, m_{r,1})))$ can pass with only a negligible probability by the entropy-preserving property of $f_{\mathsf{R},2}$ and due to the randomness of $d_{r,2} \leftarrow_\$ \mathcal{D}_{\mathsf{R},2}$, and consequently, P_i will set $\mathsf{dk}_i := \perp$ with overwhelming probability.

Responder-Robustness. Suppose that P_i invokes normal algorithms Init and DerI while P_r invokes anamorphic algorithm aDerR, then P_i computes $m_{i,2} := f_{\mathsf{I},2}(d_{i,2})$ by using a uniformly chosen $d_{i,2} \leftarrow_\$ \mathcal{D}_{\mathsf{I},2}$. When P_r invokes the anamorphic algorithm aDerR to check whether $m_{i,2} = f_{\mathsf{I},2}(\mathsf{PRF_I}(s_r, m_{i,1}))$ holds, we know that here $f_{\mathsf{I},2}(\mathsf{PRF_I}(s_r, m_{i,1}))$ is independent of $m_{i,2} := f_{\mathsf{I},2}(d_{i,2})$ since $d_{i,2} \leftarrow_\$ \mathcal{D}_{\mathsf{I},2}$ is chosen independently of $s_r, m_{i,1}$ by P_i. Thus for every possible value of $f_{\mathsf{I},2}(\mathsf{PRF_I}(s_r, m_{i,1}))$, the check $m_{i,2} := f_{\mathsf{I},2}(d_{i,2}) = f_{\mathsf{I},2}(\mathsf{PRF_I}(s_r, m_{i,1}))$ can pass with only a negligible probability by the entropy-preserving property of $f_{\mathsf{I},2}$ and due to the randomness of $d_{i,2} \leftarrow_\$ \mathcal{D}_{\mathsf{I},2}$, and consequently, P_r will set $\mathsf{dk}_r := \perp$ overwhelmingly.

4.3 Security Proofs

We show the strong security of the $\mathsf{AM\text{-}AKE}$ proposed in Subsect. 4.2.

Theorem 4 (Strong Security of $\mathsf{AM\text{-}AKE}$**).** *Let* AKE *be a qualified two-pass AKE scheme satisfying the requirements listed in Table 2, and let* PRF *be a pseudo-random function. Then the* $\mathsf{AM\text{-}AKE}$ *constructed in Subsect. 4.2 achieves both the* $\mathsf{sIND\text{-}WM}$ *and* $\mathsf{sPR\text{-}DK}$ *security.*

The proof of Theorem 4 consists of two parts: the $\mathsf{sIND\text{-}WM}$ security follows from Lemma 1 and Lemma 2, and the $\mathsf{sPR\text{-}DK}$ security follows from Lemma 3.

Lemma 1. *For any adversary* $\mathcal{A}$*, it holds that* $\left|\Pr\left[\mathcal{A}(\mathsf{pk},\mathsf{sk})=1\right]-\Pr\left[\mathcal{A}(\mathsf{apk},\mathsf{ask})=1\right]\right|=0$*, where* $(\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{Gen}$ *and* $(\mathsf{apk},\mathsf{ask},\mathsf{aux})\leftarrow\mathsf{aGen}$.

Proof of Lemma 1. In $\mathsf{AM\text{-}AKE}$, the anamorphic key-pair $(\mathsf{apk},\mathsf{ask})$ is generated by SimGen, and thus has the same distribution as the norm pair $(\mathsf{pk},\mathsf{sk})$ generated by Gen, according to the secret extractability of Gen. □

Lemma 2. *There exists PPT simulator* $\mathsf{Sim}=(\mathsf{SimI},\mathsf{SimR})$*, such that for any PPT adversary* $\mathcal{A}$ *and* $N=\mathsf{poly}(\kappa)$*,* $\left|\Pr\left[\mathsf{Exp}^{\mathsf{sIND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}=1\right]-\frac{1}{2}\right|\le\mathsf{negl}(\kappa)$.

Lemma 3. *There exists PPT simulator* $\mathsf{Sim}=(\mathsf{SimI},\mathsf{SimR})$*, such that for any PPT adversary* $\mathcal{A}$ *and* $N=\mathsf{poly}(\kappa)$*,* $\left|\Pr\left[\mathsf{Exp}^{\mathsf{sPR\text{-}DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}=1\right]-\frac{1}{2}\right|\le\mathsf{negl}(\kappa)$.

Due to space limitations, the proofs of Lemma 2 and Lemma 3 are postponed to Appendix C.1 and Appendix C.3, respectively. Here we only present the description of the simulator $\mathsf{Sim}=(\mathsf{SimI},\mathsf{SimR})$ used in these proofs, and we refer to Subsect. 1.2 for an overview of the proofs.

- $\underline{R_i\leftarrow\mathsf{SimI}(\mathsf{apk}_r,\mathsf{ask}_i,\mathsf{aux}_i=\mathsf{msk}_i,R'_i)}$: Here R'_i is an internal randomness used in aInit, and thus includes $d_{i,1}$ as well as the randomness used in $\overline{\mathsf{Init}}$, denoted by $d_{i,3}$, i.e., $R'_i=(d_{i,1},d_{i,3})$. This algorithm aims to explain R'_i as a randomness R_i for Init. To this end, it computes $s_i:=\mathsf{Extract}(\mathsf{msk}_i,\mathsf{apk}_r)$, $m_{i,1}:=f_{\mathsf{I},1}(d_{i,1})$, $d_{i,2}:=\mathsf{PRF}_{\mathsf{I}}(s_i,m_{i,1})$, and outputs $R_i:=(d_{i,1},d_{i,2},d_{i,3})$.
- $\underline{R_r\leftarrow\mathsf{SimR}(\mathsf{apk}_i,\mathsf{ask}_r,\mathsf{aux}_r=\mathsf{msk}_r,m,R'_r)}$: Here R'_r is an internal randomness used in aDerR, and thus includes $d_{r,1}$ as well as the randomness used in $\overline{\mathsf{DerR}}$, denoted by $d_{r,3}$, i.e., $R'_r=(d_{r,1},d_{r,3})$. This algorithm aims to explain R'_r as a randomness R_r for DerR. To this end, it parses $m=(m_{i,1},m_{i,2},m_{i,3})$, computes $s_r:=\mathsf{Extract}(\mathsf{msk}_r,\mathsf{apk}_i)$, $m_{r,1}:=f_{\mathsf{R},1}(d_{r,1})$, $d_{r,2}:=\mathsf{PRF}_{\mathsf{R}}(s_r,(m_{i,1},m_{r,1}))$ and outputs $R_r:=(d_{r,1},d_{r,2},d_{r3})$.

5 Instantiations of Robust and Strongly-Secure AM-AKE

To instantiate the AM-AKE generic construction proposed in Sect. 4, we can employ any pseudo-random function PRF, and thus we only need to instantiate the underlying *qualified* AKE, i.e., AKE satisfying the requirements in Table 2.

In this section, we will show that the popular SIG+KEM paradigm [19] and three-KEM paradigm [20] for constructing AKE yield *qualified* AKE schemes, as long as the underlying SIG and/or KEM satisfy certain conditions. Then by plugging them into the generic construction in Sect. 4, we immediately obtain concrete AM-AKE schemes achieving initiator-robustness, responder-robustness and strong security. More precisely, in Subsect. 5.1, we show how to obtain qualified AKE and AM-AKE via the SIG+KEM paradigm, and in Subsect. 5.2, we show how to obtain them via the three-KEM paradigm.

5.1 Instantiation from The SIG+KEM Paradigm

Qualified AKE via The SIG+KEM Paradigm. We first recall the SIG+KEM paradigm of constructing two-pass AKE according to [19]. Let $\mathsf{KEM} = (\mathsf{Gen}_{\mathsf{KEM}}, \mathsf{Encap}, \mathsf{Decap})$ be a KEM scheme, $\mathsf{SIG} = (\mathsf{Gen}_{\mathsf{SIG}}, \mathsf{Sign}, \mathsf{Vrfy})$ a signature scheme and H a suitable hash function. The resulting $\mathsf{AKE}_{\mathsf{KS}} = (\mathsf{Gen}_{\mathsf{KS}}, \mathsf{Init}_{\mathsf{KS}}, \mathsf{DerR}_{\mathsf{KS}}, \mathsf{DerI}_{\mathsf{KS}})$ is described as follows (see also Fig. 5 with dotted boxes for the paradigm).

- $\underline{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{KS}}}$: Invoke $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{SIG}}$ and return $(\mathsf{pk}, \mathsf{sk})$.
- $\underline{(\mathsf{msg}_i, \mathsf{st}) \leftarrow \mathsf{Init}_{\mathsf{KS}}(\mathsf{pk}_r, \mathsf{sk}_i)}$: Invoke $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$, $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, \widetilde{\mathsf{pk}})$, and output $\mathsf{msg}_i := (\widetilde{\mathsf{pk}}, \sigma_i)$ and the state $\mathsf{st} := (\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}})$.
- $\underline{(\mathsf{msg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}_{\mathsf{KS}}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i = (\widetilde{\mathsf{pk}}, \sigma_i))}$: If $\mathsf{Vrfy}(\mathsf{pk}_i, \widetilde{\mathsf{pk}}, \sigma_i) = 0$: output $\perp$; if $\mathsf{Vrfy}(\mathsf{pk}_i, \widetilde{\mathsf{pk}}, \sigma_i) = 1$, invoke $(K, \psi) \leftarrow \mathsf{Encap}(\widetilde{\mathsf{pk}})$, $\sigma_r \leftarrow \mathsf{Sign}(\mathsf{sk}_r, (\widetilde{\mathsf{pk}}, \psi))$, and output $\mathsf{msg}_r := (\psi, \sigma_r)$ and session key $\mathsf{K}_r := \mathsf{H}(K, \mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r)$.
- $\underline{\mathsf{K}_i \leftarrow \mathsf{DerI}_{\mathsf{KS}}(\mathsf{pk}_r, \mathsf{sk}_i, \mathsf{msg}_r = (\psi, \sigma_r), \mathsf{st} = (\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}))}$: If $\mathsf{Vrfy}(\mathsf{pk}_r, (\widetilde{\mathsf{pk}}, \psi), \sigma_r) = 0$: output $\perp$; if $\mathsf{Vrfy}(\mathsf{pk}_r, (\widetilde{\mathsf{pk}}, \psi), \sigma_r) = 1$: invoke $K \leftarrow \mathsf{Decap}(\widetilde{\mathsf{sk}}, \psi)$ and output $\mathsf{K}_i := \mathsf{H}(K, \mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r)$.

Qualified $\mathsf{AKE}_{\mathsf{KS}}$	SIG		KEM	
	$\mathsf{Gen}_{\mathsf{SIG}}$	Sign	$\mathsf{Gen}_{\mathsf{KEM}}$	Encap
Requirements	*secret extract.*	*2-separable* with *entropy-preserv.* f_{S}	*entropy-preserv.*	*entropy-preserv.*
Supportive Func./Alg.	$(\mathsf{SimGen}, \mathsf{Extract})$	$(f_{\mathsf{S}}, \overline{\mathsf{Sign}})$	$\mathsf{pk} := \overline{\mathsf{Gen}}_{\mathsf{KEM}}(d_{\mathsf{G}})$	$\psi := \overline{\mathsf{Encap}}(d_{\mathsf{K}})$

Table 3. Requirements for the building blocks $\mathsf{SIG} = (\mathsf{Gen}_{\mathsf{SIG}}, \mathsf{Sign}, \mathsf{Vrfy})$ and $\mathsf{KEM} = (\mathsf{Gen}_{\mathsf{KEM}}, \mathsf{Encap}, \mathsf{Decap})$ of the KEM-SIG paradigm in order to get a qualified $\mathsf{AKE}_{\mathsf{KS}}$.

Below we will show that the $\mathsf{AKE}_{\mathsf{KS}}$ is *qualified* for constructing AM-AKE, if the underlying SIG and KEM satisfy the following requirements (see also Table 3).

<u>Requirements for $\mathsf{SIG} = (\mathsf{Gen}_{\mathsf{SIG}}, \mathsf{Sign}, \mathsf{Vrfy})$:</u>

- $\mathsf{Gen}_{\mathsf{SIG}}$ has secret extractability, supported by $(\mathsf{SimGen}, \mathsf{Extract})$ as per Def. 9;
- Sign is 2-separable for generating σ, supported by $(f_{\mathsf{S}}, \overline{\mathsf{Sign}})$ as per Def. 8, i.e., $\underline{\mathsf{Sign}(\mathsf{sk}, m)}$ generates σ by sampling $d_{\mathsf{S}} \leftarrow_{\$} \mathcal{D}_{\mathsf{S}}$, computing $\sigma_1 := f_{\mathsf{S}}(d_{\mathsf{S}})$, invoking $\sigma_2 \leftarrow \overline{\mathsf{Sign}}(\mathsf{sk}, m, d_{\mathsf{S}})$, and setting $\sigma := (\sigma_1, \sigma_2)$;
- The function f_{S} is entropy-preserving as per Def. 7.

<u>Requirements for $\mathsf{KEM} = (\mathsf{Gen}_{\mathsf{KEM}}, \mathsf{Encap}, \mathsf{Decap})$:</u>

Party $P_i(\mathsf{pk}_i, \mathsf{sk}_i, \boxed{\mathsf{aux}_i = \mathsf{msk}_i})$

$(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$
$\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, \widetilde{\mathsf{pk}})$

$d_{\mathsf{G}} \leftarrow_\$ \mathcal{D}_{\mathsf{G}};\ (\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) := \mathsf{Gen}_{\mathsf{KEM}}(d_{\mathsf{G}})$
$s_i := \mathsf{Extract}(\mathsf{msk}_i, \mathsf{pk}_r)$
$d_{\mathsf{S},i} := \mathsf{PRF}_{\mathsf{I}}(s_i, \widetilde{\mathsf{pk}})$
$\sigma_{i,1} := f_{\mathsf{S}}(d_{\mathsf{S},i})$
$\sigma_{i,2} \leftarrow \overline{\mathsf{Sign}}(\mathsf{sk}_i, \widetilde{\mathsf{pk}}, d_{\mathsf{S},i})$
$\sigma_i := (\sigma_{i,1}, \sigma_{i,2})$

$\mathsf{msg}_i := (\widetilde{\mathsf{pk}}, \sigma_i) \longrightarrow$

$\mathsf{st} := (\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}})$
$\mathsf{aux}'_i := (s_i, \sigma_i)$

Party $P_r(\mathsf{pk}_r, \mathsf{sk}_r, \boxed{\mathsf{aux}_r = \mathsf{msk}_r})$

$\mathbf{if}\ \mathsf{Vrfy}(\mathsf{pk}_i, \widetilde{\mathsf{pk}}, \sigma_i) = 0:$
  $\mathbf{return}\ \perp$

$(K, \psi) \leftarrow \mathsf{Encap}(\widetilde{\mathsf{pk}})$
$\sigma_r \leftarrow \mathsf{Sign}(\mathsf{sk}_r, (\widetilde{\mathsf{pk}}, \psi))$

$d_{\mathsf{K}} \leftarrow_\$ \mathcal{D}_{\mathsf{K}};\ (K, \psi) := \mathsf{Encap}(\widetilde{\mathsf{pk}}; d_{\mathsf{K}})$
$s_r := \mathsf{Extract}(\mathsf{msk}_r, \mathsf{pk}_i)$
$d_{\mathsf{S},r} := \mathsf{PRF}_{\mathsf{R}}(s_r, (\widetilde{\mathsf{pk}}, \psi))$
$\sigma_{r,1} := f_{\mathsf{S}}(d_{\mathsf{S},r})$
$\sigma_{r,2} \leftarrow \overline{\mathsf{Sign}}(\mathsf{sk}_r, (\widetilde{\mathsf{pk}}, \psi), d_{\mathsf{S},r})$
$\sigma_r := (\sigma_{r,1}, \sigma_{r,2})$
parse $\sigma_i = (\sigma_{i,1}, \sigma_{i,2})$
$\mathbf{if}\ \sigma_{i,1} = f_{\mathsf{S}}(\mathsf{PRF}_{\mathsf{I}}(s_r, \widetilde{\mathsf{pk}})):$
  $\mathsf{dk}_r := \mathsf{PRF}_{\mathsf{D}}(s_r, (\widetilde{\mathsf{pk}}, \sigma_i, \psi, \sigma_r))$
$\mathbf{else}:$
  $\mathsf{dk}_r := \perp$
$\mathsf{K}_r := \mathsf{H}(K, \mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r)$

$\longleftarrow \mathsf{msg}_r := (\psi, \sigma_r)$

Party P_i (continued):

$\mathbf{if}\ \mathsf{Vrfy}(\mathsf{pk}_r, (\widetilde{\mathsf{pk}}, \psi), \sigma_r) = 0:$
  $\mathbf{return}\ \perp$

parse $\sigma_r = (\sigma_{r,1}, \sigma_{r,2})$
$\mathbf{if}\ \sigma_{r,1} = f_{\mathsf{S}}(\mathsf{PRF}_{\mathsf{R}}(s_i, (\widetilde{\mathsf{pk}}, \psi))):$
  $\mathsf{dk}_i := \mathsf{PRF}_{\mathsf{D}}(s_i, (\widetilde{\mathsf{pk}}, \sigma_i, \psi, \sigma_r))$
$\mathbf{else}:$
  $\mathsf{dk}_i := \perp$
$K \leftarrow \mathsf{Decap}(\widetilde{\mathsf{sk}}, \psi)$
$\mathsf{K}_i := \mathsf{H}(K, \mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r)$

Fig. 5. The SIG+KEM paradigm for AKE (with dotted boxes) and the resulting robust and strongly-secure AM-AKE via our generic construction in Sect. 4 (with normal algorithms in dotted boxes and anamorphic ones in gray boxes).

- The function $\overline{\mathsf{Gen}}_{\mathsf{KEM}}(\cdot) : \mathcal{D}_{\mathsf{G}} \longrightarrow \{0,1\}^*$ is entropy-preserving, where $\overline{\mathsf{Gen}}_{\mathsf{KEM}}$ functions the same as $\mathsf{Gen}_{\mathsf{KEM}}$ that takes a randomness $d_{\mathsf{G}} \in \mathcal{D}_{\mathsf{G}}$ as input but outputs only pk (and does not output sk).
- For any public key pk, the function $\overline{\mathsf{Encap}}(\mathsf{pk}; \cdot) : \mathcal{D}_{\mathsf{K}} \longrightarrow \{0,1\}^*$ is entropy-preserving, where $\overline{\mathsf{Encap}}(\mathsf{pk}; \cdot)$ functions the same as $\mathsf{Encap}(\mathsf{pk})$ that takes a randomness $d_{\mathsf{K}} \in \mathcal{D}_{\mathsf{K}}$ as input but outputs only ψ (and does not output K).

With such SIG and KEM, we prove that the resulting $\mathsf{AKE}_{\mathsf{KS}}$ is a qualified AKE via the following Lemma 4. Then by plugging the qualified $\mathsf{AKE}_{\mathsf{KS}}$ into our generic construction in Sect. 4, we immediately get a robust and strongly-secure two-pass AM-AKE scheme, as shown in Fig. 5 with gray boxes.

Lemma 4. *If* SIG *and* KEM *meet the above requirements, then the* $\mathsf{AKE}_{\mathsf{KS}}$ *yielded by the SIG+KEM paradigm is a qualified AKE for constructing AM-AKE.*

Proof. To prove that $\mathsf{AKE}_{\mathsf{KS}} = (\mathsf{Gen}_{\mathsf{KS}}, \mathsf{Init}_{\mathsf{KS}}, \mathsf{DerR}_{\mathsf{KS}}, \mathsf{DerI}_{\mathsf{KS}})$ is a qualified one, we show that all requirements listed in Table 2 are satisfied, i.e., $\mathsf{Gen}_{\mathsf{KS}}$ has secret extractability, $\mathsf{Init}_{\mathsf{KS}}$ is 3-separable with entropy-preserving functions $(f_{\mathsf{I},1}, f_{\mathsf{I},2})$, and $\mathsf{DerR}_{\mathsf{KS}}$ is 3-separable with entropy-preserving functions $(f_{\mathsf{R},1}, f_{\mathsf{R},2})$.

- Since $\mathsf{Gen}_{\mathsf{KS}} = \mathsf{Gen}_{\mathsf{SIG}}$, the secret extract. of $\mathsf{Gen}_{\mathsf{KS}}$ follows from that of $\mathsf{Gen}_{\mathsf{SIG}}$.
- The process of $\mathsf{Init}_{\mathsf{KS}}(\mathsf{pk}_r, \mathsf{sk}_i)$ for generating $(\mathsf{msg}_i = (\widetilde{\mathsf{pk}}, \sigma_i = (\sigma_{i,1}, \sigma_{i,2})), \mathsf{st} = (\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}))$ can be decomposed into three steps, since Sign is 2-separable:

1. $d_{\mathsf{G}} \leftarrow_{\$} \mathcal{D}_{\mathsf{G}}$ and $\widetilde{\mathsf{pk}} := \overline{\mathsf{Gen}}_{\mathsf{KEM}}(d_{\mathsf{G}})$. So we can define $f_{\mathsf{I},1} := \overline{\mathsf{Gen}}_{\mathsf{KEM}}$, and then the entropy-preserving property of $f_{\mathsf{I},1}$ follows from that of $\overline{\mathsf{Gen}}_{\mathsf{KEM}}$.
2. $d_{\mathsf{S},i} \leftarrow_{\$} \mathcal{D}_{\mathsf{S}}$ and $\sigma_{i,1} := f_{\mathsf{S}}(d_{\mathsf{S},i})$. So we can define $f_{\mathsf{I},2} := f_{\mathsf{S}}$, and then the entropy-preserving property of $f_{\mathsf{I},2}$ follows from that of f_{S}.
3. $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) := \mathsf{Gen}_{\mathsf{KEM}}(d_{\mathsf{G}})$, $\sigma_{i,2} \leftarrow \overline{\mathsf{Sign}}(\mathsf{sk}_i, \widetilde{\mathsf{pk}}, d_{\mathsf{S},i})$, and set $\mathsf{st} := (\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}})$. This process can be defined as $(\sigma_{i,2}, \mathsf{st}) \leftarrow \overline{\mathsf{Init}}_{\mathsf{KS}}(\mathsf{pk}_r, \mathsf{sk}_i, d_{\mathsf{G}}, d_{\mathsf{S},i})$.

Consequently, $\mathsf{Init}_{\mathsf{KS}}$ is 3-separable with two entropy-preserving functions $(f_{\mathsf{I},1} = \overline{\mathsf{Gen}}_{\mathsf{KEM}}, f_{\mathsf{I},2} = f_{\mathsf{S}})$ and an algorithm $\overline{\mathsf{Init}}_{\mathsf{KS}}$.

- Similarly, the process of $\mathsf{DerR}_{\mathsf{KS}}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i = (\widetilde{\mathsf{pk}}, \sigma_i))$ for generating $(\mathsf{msg}_r = (\psi, \sigma_r = (\sigma_{r,1}, \sigma_{r,2})), \mathsf{K}_r)$ can be decomposed into three steps:

1. $d_{\mathsf{K}} \leftarrow_{\$} \mathcal{D}_{\mathsf{K}}$ and $\psi := \overline{\mathsf{Encap}}(\widetilde{\mathsf{pk}}; d_{\mathsf{K}})$. So we can define $f_{\mathsf{R},1} := \overline{\mathsf{Encap}}(\widetilde{\mathsf{pk}}; \cdot)$, and then the entropy-preserving of $f_{\mathsf{R},1}$ follows from that of $\overline{\mathsf{Encap}}(\widetilde{\mathsf{pk}}; \cdot)$.
2. $d_{\mathsf{S},r} \leftarrow_{\$} \mathcal{D}_{\mathsf{S}}$ and $\sigma_{r,1} := f_{\mathsf{S}}(d_{\mathsf{S},r})$. So we can define $f_{\mathsf{R},2} := f_{\mathsf{S}}$, and then the entropy-preserving property of $f_{\mathsf{R},2}$ follows from that of f_{S}.
3. If $\mathsf{Vrfy}(\mathsf{pk}_i, \widetilde{\mathsf{pk}}, \sigma_i) = 1$: $(K, \psi) := \mathsf{Encap}(\widetilde{\mathsf{pk}}; d_{\mathsf{K}})$, $\sigma_{r,2} \leftarrow \overline{\mathsf{Sign}}(\mathsf{sk}_r, (\widetilde{\mathsf{pk}}, \psi), d_{\mathsf{S},r})$, and set $\mathsf{K}_r := \mathsf{H}(K, \mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r)$. Otherwise output $\perp$. This process can be defined as $(\sigma_{r,2}, \mathsf{K}_r) \leftarrow \overline{\mathsf{DerR}}_{\mathsf{KS}}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i = (\widetilde{\mathsf{pk}}, \sigma_i), d_{\mathsf{K}}, d_{\mathsf{S},r})$.

Consequently, $\mathsf{DerR}_{\mathsf{KS}}$ is 3-separable with two entropy-preserving functions $(f_{\mathsf{R},1} = \overline{\mathsf{Encap}}(\widetilde{\mathsf{pk}}; \cdot), f_{\mathsf{R},2} = f_{\mathsf{S}})$ and an algorithm $\overline{\mathsf{DerR}}_{\mathsf{KS}}$. $\square$

Concrete Instantiations. To obtain concrete qualified AKE scheme via the SIG+KEM paradigm, it remains to present concrete SIG and KEM schemes satisfying the requirements described above (cf. Table 3). More precisely, we will show that any IND-CPA secure KEM suffices, and then for SIG, we present a concrete instantiation over asymmetric pairing groups.

Concrete KEM. In fact, any IND-CPA secure KEM has entropy-preserving $\overline{\mathsf{Gen}}_{\mathsf{KEM}}$ and $\overline{\mathsf{Encap}}$, which output only pk and ψ respectively. Intuitively, if an independently generated $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen}$ or $(\widetilde{K}, \widetilde{\psi}) \leftarrow \mathsf{Encap}(\mathsf{pk})$ leads to $\widetilde{\mathsf{pk}} = \mathsf{pk}$ or $\widetilde{\psi} = \psi$ with non-negligible probability for a target pk or ψ, an adversary can use the accompanying $\widetilde{\mathsf{sk}}$ or $\widetilde{K}$ to break the IND-CPA security of KEM easily. More precisely, we have the following lemma with proof postponed to Appendix D.1.

Lemma 5 (Any IND-CPA Secure KEM has Entropy-Preserving $\overline{\mathsf{Gen}}_{\mathsf{KEM}}$ and $\overline{\mathsf{Encap}}$). *If* $\mathsf{KEM} = (\mathsf{Gen}_{\mathsf{KEM}}, \mathsf{Encap}, \mathsf{Decap})$ *is a* IND-CPA *secure KEM scheme, then the function* $\overline{\mathsf{Gen}}_{\mathsf{KEM}}(\cdot)$ *that outputs only* pk *and the function* $\overline{\mathsf{Encap}}(\mathsf{pk}; \cdot)$ *that outputs only* ψ *are entropy-preserving.*

Concrete SIG. Let $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2, g_T)$ be a description of asymmetric pairing group, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerated bilinear pairing, and g_1, g_2, g_T are generators of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ respectively. Moreover, let $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$ be a hash function. We present a concrete scheme $\mathsf{SIG}_{\mathsf{DDH}} = (\mathsf{Gen}_{\mathsf{DDH}}, \mathsf{Sign}, \mathsf{Vrfy})$ as follows.

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{DDH}}$: it picks $x \leftarrow_{\$} \mathbb{Z}_p$, and sets $(\mathsf{pk} := e(g_1, g_2)^x, \mathsf{sk} := g_2^x)$.

- $\underline{\sigma \leftarrow \mathsf{Sign}(\mathsf{sk} = g_2^x, m):}$ it chooses $r \leftarrow_\$ \mathbb{Z}_p$ randomly, then computes $\sigma_1 := g_1^r$, $d := \mathsf{H}(m, \sigma_1) \in \mathbb{Z}_p$, $\sigma_2 := {g_2}^{x \cdot d + r}$, and outputs $\sigma := (\sigma_1, \sigma_2)$.
- $\underline{0/1 \leftarrow \mathsf{Vrfy}(\mathsf{pk} = e(g_1, g_2)^x, m, \sigma = (\sigma_1, \sigma_2)):}$ it computes $d := \mathsf{H}(m, \sigma_1) \in \mathbb{Z}_p$, and outputs 1 if and only if $e(g_1, \sigma_2) = e(g_1, g_2)^{x \cdot d} \cdot e(\sigma_1, g_2)$ holds.

Intuitively, the scheme $\mathsf{SIG}_{\mathsf{DDH}}$ can be viewed as a variant of the Schnorr signature scheme [24], by lifting it from $(\mathbb{Z}_p, \mathbb{G})$ of a cyclic group to $(\mathbb{G}_2, \mathbb{G}_T)$ of the asymmetric pairing group. It is routine to check the correctness of $\mathsf{SIG}_{\mathsf{DDH}}$. Next we show its EUF-CMA security with proof appeared in Appendix D.2, since the proof is essentially the same as that for the Schnorr scheme.

Theorem 5 (Security of $\mathsf{SIG}_{\mathsf{DDH}}$). *If the DDH assumption holds over $\mathbb{G}_1$ and* H *is a random oracle, then the proposed* $\mathsf{SIG}_{\mathsf{DDH}}$ *achieves* EUF-CMA *security.*

Below we show that $\mathsf{SIG}_{\mathsf{DDH}}$ satisfies the requirements listed in Table 3 via the following two lemmas.

Lemma 6 (Secret Extractability of $\mathsf{Gen}_{\mathsf{DDH}}$). *The key generation algorithm* $\mathsf{Gen}_{\mathsf{DDH}}$ *has secret extractability based on the DDH assumption over* $\mathbb{G}_2$*.*

Proof. We first describe the supportive algorithms $\mathsf{SimGen}_{\mathsf{DDH}}$ and $\mathsf{Extract}_{\mathsf{DDH}}$ as follows, which take pp as an implicit input, the same as $\mathsf{Gen}_{\mathsf{DDH}}$.

- $\underline{(\mathsf{pk}, \mathsf{sk}, \mathsf{msk}) \leftarrow \mathsf{SimGen}_{\mathsf{DDH}}}$: it picks $x \leftarrow_\$ \mathbb{Z}_p$, and sets $(\mathsf{pk} := e(g_1, g_2)^x, \mathsf{sk} := g_2^x, \mathsf{msk} := x)$.
- $\underline{s \leftarrow \mathsf{Extract}_{\mathsf{DDH}}(\mathsf{msk}_i = x_i, \mathsf{pk}_r = e(g_1, g_2)^{x_r})}$: it computes $s := \mathsf{pk}_r^{\mathsf{msk}_i} = e(g_1, g_2)^{x_r x_i}$.

Next we show that the proposed $(\mathsf{SimGen}_{\mathsf{DDH}}, \mathsf{Extract}_{\mathsf{DDH}})$ satisfy the requirements of secret extractability (cf. Def. 9). It is easy to see that the key-pair $(\mathsf{pk}, \mathsf{sk})$ generated by $\mathsf{SimGen}_{\mathsf{DDH}}$ has the same distribution as the normal pair generated by $\mathsf{Gen}_{\mathsf{DDH}}$, and check that the extraction correctness holds, i.e., $\mathsf{Extract}_{\mathsf{DDH}}(\mathsf{msk}_i, \mathsf{pk}_r) = e(g_1, g_2)^{x_i x_r} = \mathsf{Extract}_{\mathsf{DDH}}(\mathsf{msk}_r, \mathsf{pk}_i)$.

It remains to prove the pseudo-randomness of $\mathsf{Extract}_{\mathsf{DDH}}(\mathsf{msk}_i, \mathsf{pk}_r) = e(g_1, g_2)^{x_i x_r}$ conditioned on $(\mathsf{pk}_i = e(g_1, g_2)^{x_i}, \mathsf{pk}_r = e(g_1, g_2)^{x_r}, \mathsf{sk}_i = g_2^{x_i}, \mathsf{sk}_r = g_2^{x_r})$. More precisely, for any adversary $\mathcal{A}$ against the pseudo-randomness of the extracting, we construct an algorithm $\mathcal{B}$ against the DDH assumption over $\mathbb{G}_2$ as follows.

Given a DDH challenge $(\mathsf{pp}, g_2^{x_i}, g_2^{x_r}, T)$, $\mathcal{B}$ wants to distinguish $T = g_2^{x_i x_r}$ from $T \leftarrow_\$ \mathbb{G}_2$, where $x_i, x_r \leftarrow_\$ \mathbb{Z}_p$. To this end, $\mathcal{B}$ sets $\mathsf{sk}_i := g_2^{x_i}$, $\mathsf{sk}_r := g_2^{x_r}$, computes $\mathsf{pk}_i := e(g_1, g_2^{x_i}) = e(g_1, g_2)^{x_i}$, $\mathsf{pk}_r := e(g_1, g_2^{x_r}) = e(g_1, g_2)^{x_r}$, $s^* := e(g_1, T)$, gives $(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{sk}_i, \mathsf{sk}_r, s^*)$ to $\mathcal{A}$, and returns the output of $\mathcal{A}$ to its own challenger. It is easy to see that $\mathcal{B}$'s simulation of $(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{sk}_i, \mathsf{sk}_r)$ is perfect. If $T = g_2^{x_i x_r}$, then $s^* := e(g_1, T) = e(g_1, g_2)^{x_i x_r} = \mathsf{Extract}_{\mathsf{DDH}}(\mathsf{msk}_i, \mathsf{pk}_r)$; if $T \leftarrow_\$ \mathbb{G}_2$, then $s^* := e(g_1, T)$ is uniformly distributed over $\mathbb{G}_T$. Consequently, $\mathcal{B}$ is able to distinguish $T = g_2^{x_i x_r}$ from $T \leftarrow_\$ \mathbb{G}_2$, as long as $\mathcal{A}$ can distinguish $(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{sk}_i, \mathsf{sk}_r, s^* = \mathsf{Extract}_{\mathsf{DDH}}(\mathsf{msk}_i, \mathsf{pk}_r))$ from $(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{sk}_i, \mathsf{sk}_r, s^* \leftarrow_\$ \mathbb{G}_T)$, and we have $\mathsf{Adv}^{\mathsf{PR\text{-}Ext}}_{\mathsf{Gen}_{\mathsf{DDH}}, \mathcal{A}}(\kappa) \le \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2, \mathcal{B}}(\kappa)$, which is negligible under the DDH assumption over $\mathbb{G}_2$. This shows the pseudo-randomness of the extracting. $\square$

Lemma 7 **(2-Separability of $\mathsf{Sign}_{\mathsf{DDH}}$ with Entropy-Preserving f_{S}).** $\mathsf{Sign}_{\mathsf{DDH}}$ *is 2-separable for generating σ, and the supportive function f_{S} is entropy-preserving.*

Proof. It is easy to see that the process of $\mathsf{Sign}(\mathsf{sk} = g_2^x, m)$ generating $\sigma = (\sigma_1, \sigma_2)$ can be decomposed into two parts: the first part includes $r \leftarrow_\$ \mathbb{Z}_p$ and $\sigma_1 := g_1^r$, and the second part computes $\sigma_2 := {g_2}^{x \cdot \mathsf{H}(m,\sigma_1)+r}$. Consequently, Sign is 2-separable for generating $\sigma = (\sigma_1, \sigma_2)$, supported by $(f_{\mathsf{S}}, \overline{\mathsf{Sign}})$, where f_{S} is defined by $f_{\mathsf{S}}(r) := g_1^r$ for $r \in \mathbb{Z}_p$ and $\overline{\mathsf{Sign}}$ is defined by $\overline{\mathsf{Sign}}(\mathsf{sk} = g_2^x, m, r) := g_2^{x \cdot \mathsf{H}(m, g_1^r)+r}$. Moreover, f_{S} is entropy-preserving since for any $h \in \mathbb{G}_1$, the probability $\Pr[f_{\mathsf{S}}(r) = g_1^r = h | r \leftarrow_\$ \mathbb{Z}_p] = 1/p$ is negligible. $\square$

5.2 Instantiation from The Three-KEM Paradigm

Qualified AKE via The Three-KEM Paradigm. We first recall the three-KEM paradigm of constructing two-pass AKE according to [20]. Let $\mathsf{KEM} = (\mathsf{Gen}_{\mathsf{KEM}}, \mathsf{Encap}, \mathsf{Decap})$ and $\mathsf{KEM}_0 = (\mathsf{Gen}_{\mathsf{KEM}_0}, \mathsf{Encap}_0, \mathsf{Decap}_0)$ be two KEM schemes, and H a suitable hash function. The resulting $\mathsf{AKE}_{\mathsf{3K}} = (\mathsf{Gen}_{\mathsf{3K}}, \mathsf{Init}_{\mathsf{3K}}, \mathsf{DerR}_{\mathsf{3K}}, \mathsf{DerI}_{\mathsf{3K}})$ is described as follows (see also Fig. 6 with dotted boxes).

- $\underline{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{3K}}}$: Invoke $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$ and return $(\mathsf{pk}, \mathsf{sk})$.
- $\underline{(\mathsf{msg}_i, \mathsf{st}) \leftarrow \mathsf{Init}_{\mathsf{3K}}(\mathsf{pk}_r, \mathsf{sk}_i)}$: Invoke $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}_0}$, $(K_i, \psi_i) \leftarrow \mathsf{Encap}(\mathsf{pk}_r)$, and output $\mathsf{msg}_i := (\widetilde{\mathsf{pk}}, \psi_i)$ and the state $\mathsf{st} := (\widetilde{\mathsf{sk}}, K_i)$.
- $\underline{(\mathsf{msg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}_{\mathsf{3K}}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i = (\widetilde{\mathsf{pk}}, \psi_i))}$: Invoke $K_i \leftarrow \mathsf{Decap}(\mathsf{sk}_r, \psi_i)$, $(\widetilde{K}, \widetilde{\psi}) \leftarrow \mathsf{Encap}_0(\widetilde{\mathsf{pk}})$ and $(K_r, \psi_r) \leftarrow \mathsf{Encap}(\mathsf{pk}_i)$. Output $\mathsf{msg}_r := (\widetilde{\psi}, \psi_r)$ and session key $\mathsf{K}_r := \mathsf{H}(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r, K_i, K_r, \widetilde{K})$.
- $\underline{\mathsf{K}_i \leftarrow \mathsf{DerI}_{\mathsf{3K}}(\mathsf{pk}_r, \mathsf{sk}_i, \mathsf{msg}_r = (\widetilde{\psi}, \psi_r), \mathsf{st} = (\widetilde{\mathsf{sk}}, K_i))}$: Invoke $\widetilde{K} \leftarrow \mathsf{Decap}_0(\widetilde{\mathsf{sk}}, \widetilde{\psi})$, $K_r \leftarrow \mathsf{Decap}(\mathsf{sk}_i, \psi_r)$, and output $\mathsf{K}_i := \mathsf{H}(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r, K_i, K_r, \widetilde{K})$.

Qualified $\mathsf{AKE}_{\mathsf{3K}}$	KEM		KEM_0	
	$\mathsf{Gen}_{\mathsf{KEM}}$	Encap	$\mathsf{Gen}_{\mathsf{KEM}_0}$	Encap_0
Requirements	*secret extract.*	*entropy-preserv.*	*entropy-preserv.*	*entropy-preserv.*
Supportive Func./Alg.	$(\mathsf{SimGen}, \mathsf{Extract})$	$\psi := \overline{\mathsf{Encap}}(d_{\mathsf{K}})$	$\widetilde{\mathsf{pk}} := \overline{\mathsf{Gen}_{\mathsf{KEM}}}(d_{\mathsf{G}})$	$\widetilde{\psi} := \overline{\mathsf{Encap}}(d_{\mathsf{K}_0})$

Table 4. Requirements for the building blocks $\mathsf{KEM} = (\mathsf{Gen}_{\mathsf{KEM}}, \mathsf{Encap}, \mathsf{Decap})$ and $\mathsf{KEM}_0 = (\mathsf{Gen}_{\mathsf{KEM}_0}, \mathsf{Encap}_0, \mathsf{Decap}_0)$ of the three-KEM paradigm in order to get a qualified $\mathsf{AKE}_{\mathsf{3K}}$.

Below we will show that $\mathsf{AKE}_{\mathsf{3K}}$ is *qualified* for constructing AM-AKE, if the underlying KEM and KEM_0 satisfy the following requirements (see also Table 4).

Requirements for $\mathsf{KEM} = (\mathsf{Gen}_{\mathsf{KEM}}, \mathsf{Encap}, \mathsf{Decap})$:

- $\mathsf{Gen}_{\mathsf{KEM}}$ has secret extractability, supported by $(\mathsf{SimGen}, \mathsf{Extract})$ as per Def. 9;
- For any public key pk, the function $\overline{\mathsf{Encap}}(\mathsf{pk}; \cdot) : \mathcal{D}_{\mathsf{K}} \longrightarrow \{0,1\}^*$ is entropy-preserving, where $\overline{\mathsf{Encap}}(\mathsf{pk}; \cdot)$ functions the same as $\mathsf{Encap}(\mathsf{pk})$ that takes a randomness $d_{\mathsf{K}} \in \mathcal{D}_{\mathsf{K}}$ as input but outputs only ψ (and does not output K).

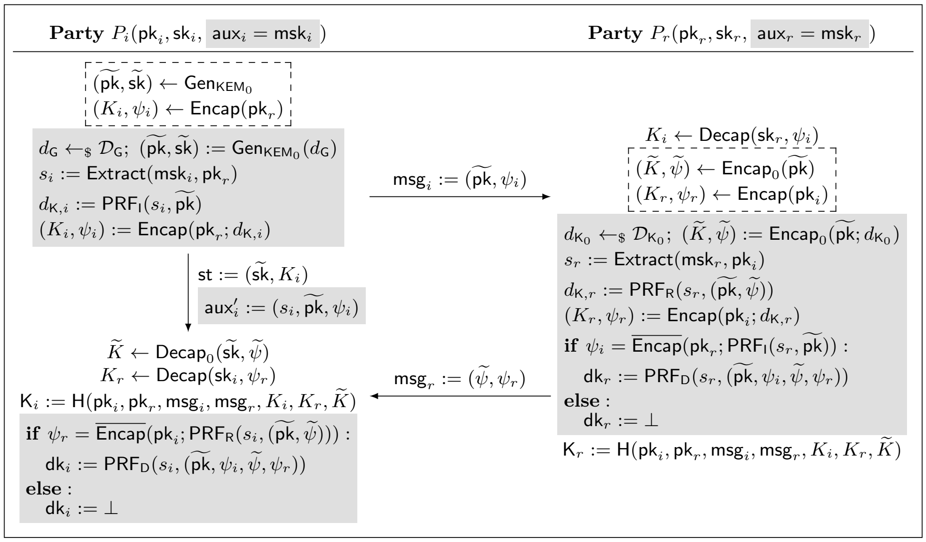

Fig. 6. The three-KEM paradigm for AKE (with dotted boxes) and the resulting robust and strongly-secure AM-AKE via our generic construction in Sect. 4 (with normal algorithms in dotted boxes and anamorphic ones in gray boxes).

Requirements for $\mathsf{KEM}_0 = (\mathsf{Gen}_{\mathsf{KEM}_0}, \mathsf{Encap}_0, \mathsf{Decap}_0)$:

- The function $\overline{\mathsf{Gen}}_{\mathsf{KEM}_0}(\cdot) : \mathcal{D}_{\mathsf{G}} \longrightarrow \{0,1\}^*$ is entropy-preserving, where $\overline{\mathsf{Gen}}_{\mathsf{KEM}_0}$ functions the same as $\mathsf{Gen}_{\mathsf{KEM}_0}$ that takes a randomness $d_{\mathsf{G}} \in \mathcal{D}_{\mathsf{G}}$ as input but outputs only $\widetilde{\mathsf{pk}}$ (and does not output $\widetilde{\mathsf{sk}}$).
- For any public key $\widetilde{\mathsf{pk}}$, the function $\overline{\mathsf{Encap}}_0(\widetilde{\mathsf{pk}}; \cdot) : \mathcal{D}_{\mathsf{K}_0} \longrightarrow \{0,1\}^*$ is entropy-preserving, where $\overline{\mathsf{Encap}}_0(\widetilde{\mathsf{pk}}; \cdot)$ is the same as $\mathsf{Encap}_0(\widetilde{\mathsf{pk}})$ that takes a randomness $d_{\mathsf{K}_0} \in \mathcal{D}_{\mathsf{K}_0}$ as input but outputs only $\widetilde{\psi}$ (and does not output $\widetilde{K}$).

With such KEM and KEM_0, we prove that the resulting $\mathsf{AKE}_{\mathsf{3K}}$ is a qualified AKE via the following Lemma 8. The proof of Lemma 8 is quite similar to that of Lemma 4 in Subsect. 5.1, and thus we postpone it to Appendix D.3.

Lemma 8. *If* KEM *and* KEM_0 *meet the above requirements, the* $\mathsf{AKE}_{\mathsf{3K}}$ *yielded by the three-KEM paradigm is a qualified AKE for constructing AM-AKE.*

Then by plugging the qualified $\mathsf{AKE}_{\mathsf{3K}}$ into our generic construction in Sect. 4, we immediately get a robust and strongly-secure two-pass AM-AKE scheme, as shown in Fig. 6 with gray boxes.

Concrete Instantiations. To obtain concrete qualified AKE scheme via the three-KEM paradigm, it remains to present concrete KEM schemes KEM and KEM_0 satisfying the requirements described above (cf. Table 4). Specially, as shown in Lemma 5 in Subsect. 5.1, any $\mathsf{IND\text{-}CPA}$ secure KEM has entropy-preserving $\overline{\mathsf{Gen}}_{\mathsf{KEM}}$ and $\overline{\mathsf{Encap}}$, so we can instantiate KEM_0 with any $\mathsf{IND\text{-}CPA}$ secure KEM scheme. For KEM, we present a concrete instantiation over asymmetric pairing groups.

Concrete KEM scheme KEM. Let $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2, g_T)$ be a description of asymmetric pairing group. We present a concrete KEM scheme $\mathsf{KEM}_{\mathsf{DDH}} = (\mathsf{Gen}_{\mathsf{DDH}}, \mathsf{Encap}, \mathsf{Decap})$ as follows:

- $\underline{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{DDH}}}$: it picks $x \leftarrow_{\$} \mathbb{Z}_p$, and sets $(\mathsf{pk} := e(g_1, g_2)^x, \mathsf{sk} := g_2^x)$.
- $\underline{(K, \psi) \leftarrow \mathsf{Encap}(\mathsf{pk} = e(g_1, g_2)^x)}$: it chooses $r \leftarrow_{\$} \mathbb{Z}_p$ randomly, then computes $\psi := g_1^r$, $K := (e(g_1, g_2)^x)^r = e(g_1, g_2)^{xr}$, and outputs (K, ψ).
- $\underline{K \leftarrow \mathsf{Decap}(\mathsf{sk} = g_2^x, \psi = g_1^r)}$: it computes $\mathsf{K} := e(g_1^r, g_2^x)$ and outputs K.

It is routine to check the correctness of $\mathsf{KEM}_{\mathsf{DDH}}$. Next we show its $\mathsf{IND\text{-}CPA}$ security based on the DDH assumption over $\mathbb{G}_1$ via the following theorem. The proof is quite straightforward and thus we postpone it to Appendix D.4.

Theorem 6 (Security of $\mathsf{KEM}_{\mathsf{DDH}}$). *If the DDH assumption holds over $\mathbb{G}_1$, then the proposed $\mathsf{KEM}_{\mathsf{DDH}}$ achieves $\mathsf{IND\text{-}CPA}$ security.*

Below we show that $\mathsf{KEM}_{\mathsf{DDH}}$ satisfies the requirements listed in Table 4, i.e., its key generation algorithm $\mathsf{Gen}_{\mathsf{DDH}}$ has secret extractability, and the function $\overline{\mathsf{Encap}}(\mathsf{pk}; \cdot)$ that outputs only ψ is entropy-preserving. Since $\mathsf{Gen}_{\mathsf{DDH}}$ is identical to that of $\mathsf{SIG}_{\mathsf{DDH}}$ in Subsect. 5.1, as shown in Lemma 6, $\mathsf{Gen}_{\mathsf{DDH}}$ has secret extractability under the DDH assumption over $\mathbb{G}_2$. Moreover, by Lemma 5, the function $\overline{\mathsf{Encap}}(\mathsf{pk}; \cdot)$ is entropy-preserving by the $\mathsf{IND\text{-}CPA}$ security of $\mathsf{KEM}_{\mathsf{DDH}}$.

Acknowledgements. We would like to thank the reviewers for their valuable comments. The authors were partially supported by National Natural Science Foundation of China (Grant Nos. 61925207, 62372292), Guangdong Major Project of Basic and Applied Basic Research (2019B030302008), the National Key R&D Program of China under Grant 2022YFB2701500, and Young Elite Scientists Sponsorship Program by China Association for Science and Technology (YESS20200185).

References

1. Banfi, F., Gegier, K., Hirt, M., Maurer, U., Rito, G.: Anamorphic encryption, revisited. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024. pp. 3–32. Springer Nature Switzerland, Cham (2024)
2. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: Santis, A.D. (ed.) EUROCRYPT'94. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (May 1995). https://doi.org/10.1007/BFb0053428
3. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_4
4. Boyd, C., Cliff, Y., González Nieto, J., Paterson, K.G.: Efficient one-round key exchange in the standard model. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 08. LNCS, vol. 5107, pp. 69–83. Springer, Heidelberg (Jul 2008)
5. Catalano, D., Giunta, E., Migliaro, F.: Anamorphic encryption: New constructions and homomorphic realizations. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024. pp. 33–62. Springer Nature Switzerland, Cham (2024)
6. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976). https://doi.org/10.1109/TIT.1976.1055638
7. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (Aug 1984)
8. Gjøsteen, K., Jager, T.: Practical and tightly-secure digital signatures and authenticated key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 95–125. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_4
9. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences **28**(2), 270–299 (1984). https://doi.org/https://doi.org/10.1016/0022-0000(84)90070-9, https://www.sciencedirect.com/science/article/pii/0022000084900709
10. Han, S., Jager, T., Kiltz, E., Liu, S., Pan, J., Riepel, D., Schäge, S.: Authenticated key exchange and signatures with tight security in the standard model. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 670–700. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84259-8_23
11. Harkins, D., Carrel, D.: The Internet Key Exchange (IKE). IETF RFC 2409 (Proposed Standard) (1998)
12. Horel, T., Park, S., Richelson, S., Vaikuntanathan, V.: How to subvert backdoored encryption: Security against adversaries that decrypt all ciphertexts. In: Blum, A. (ed.) ITCS 2019. vol. 124, pp. 42:1–42:20. LIPIcs (Jan 2019). https://doi.org/10.4230/LIPIcs.ITCS.2019.42
13. Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 117–146. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_5
14. Katz, J., Lindell, Y.: Introduction to Modern Cryptography, Second Edition. CRC Press (2014)

15. Kutylowski, M., Persiano, G., Phan, D.H., Yung, M., Zawada, M.: Anamorphic signatures: Secrecy from a dictator who only permits authentication! In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 759–790. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2_25
16. Kutylowski, M., Persiano, G., Phan, D.H., Yung, M., Zawada, M.: The self-anti-censorship nature of encryption: On the prevalence of anamorphic cryptography. Proc. Priv. Enhancing Technol. **2023**(4), 170–183 (2023). https://doi.org/10.56553/POPETS-2023-0104, https://doi.org/10.56553/popets-2023-0104
17. 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). https://doi.org/10.1007/978-3-030-64834-3_27
18. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (May 1999). https://doi.org/10.1007/3-540-48910-X_16
19. Pan, J., Qian, C., Ringerud, M.: Signed (group) Diffie-Hellman key exchange with tight security. Journal of Cryptology **35**(4), 26 (Oct 2022). https://doi.org/10.1007/s00145-022-09438-y
20. Pan, J., Wagner, B., Zeng, R.: Lattice-based authenticated key exchange with tight security. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 616–647. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_20
21. Pan, J., Wagner, B., Zeng, R.: Tighter security for generic authenticated key exchange in the QROM. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part IV. LNCS, vol. 14441, pp. 401–433. Springer, Heidelberg (Dec 2023). https://doi.org/10.1007/978-981-99-8730-6_13
22. Persiano, G., Phan, D.H., Yung, M.: Anamorphic encryption: Private communication against a dictator. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 34–63. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_2
23. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Aug 2018). https://doi.org/10.17487/RFC8446, https://www.rfc-editor.org/info/rfc8446
24. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34805-0_22
25. von Ahn, L., Hopper, N.J.: Public-key steganography. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 323–341. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_20
26. Wang, Y., Chen, R., Huang, X., Yung, M.: Sender-anamorphic encryption reformulated: Achieving robust and generic constructions. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part VI. LNCS, vol. 14443, pp. 135–167. Springer, Heidelberg (Dec 2023). https://doi.org/10.1007/978-981-99-8736-8_5
27. Xiao, Y., Zhang, R., Ma, H.: Tightly secure two-pass authenticated key exchange protocol in the CK model. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 171–198. Springer, Heidelberg (Feb 2020). https://doi.org/10.1007/978-3-030-40186-3_9

Supplementary Material

A Additional Preliminaries

Definition 10 (Pseudo-Random Function). *A function* $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{Y}$ *is a pseudo-random function, if for any PPT adversary* $\mathcal{A}$*, it holds that* $\mathsf{Adv}_{\mathsf{PRF},\mathcal{A}}(\kappa) := |\Pr[\mathcal{A}^{\mathsf{PRF}(k,\cdot)} = 1] - \Pr[\mathcal{A}^{\mathsf{TRF}(\cdot)} = 1]| \le \mathsf{negl}(\kappa)$*, where* $k \leftarrow_\$ \mathcal{K}$ *and* TRF *is truly random function from* $\mathcal{X}$ *to* $\mathcal{Y}$*.*

Definition 11 (Digital Signature). *A digital signature scheme* $\mathsf{SIG} = (\mathsf{Gen}_{\mathsf{SIG}}, \mathsf{Sign}, \mathsf{Vrfy})$ *consists of three PPT algorithms:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{SIG}}$*: The key generation algorithm generates a pair of public key and secret key* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$*: The signing algorithm takes a secret key* sk *and a message* m *as input, and outputs a signature* σ*.*
- $0/1 \leftarrow \mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$*: The verification algorithm takes a public key* pk*, a message* m *and a signature* σ *as input, and outputs* 1 *(accepted) or* 0 *(rejected) indicating whether* σ *is a valid signature for* m*.*

Correctness. *For any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{SIG}}$*, any message* m*, and any* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$*, it holds that* $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma) = 1$*.*

EUF-CMA Security. *We consider the standard security of existential unforgeability against chosen message attacks (*$\mathsf{EUF\text{-}CMA}$*) for* SIG*. More precisely, for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}_{\mathsf{SIG},\mathcal{A}}^{\mathsf{EUF\text{-}CMA}}(\kappa) := \Pr\left[\begin{array}{c} m^* \notin \mathcal{Q}_{\mathsf{sig}} \wedge \\ \mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*) = 1 \end{array} \middle| \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{SIG}}, \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}(\cdot)}(\mathsf{pk}) \end{array}\right] \le \mathsf{negl}(\kappa), \tag{3}$$

where the oracle $\mathcal{O}_{\mathsf{Sign}}(m)$ *computes* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ *and returns* σ *to* $\mathcal{A}$*, and the set* $\mathcal{Q}_{\mathsf{sig}}$ *consists of all messages* m *that* $\mathcal{A}$ *queries to* $\mathcal{O}_{\mathsf{Sign}}(\cdot)$*.*

Definition 12 (Key Encapsulation Mechanism). *A key encapsulation mechanism* $\mathsf{KEM} = (\mathsf{Gen}_{\mathsf{KEM}}, \mathsf{Encap}, \mathsf{Decap})$ *consists of three PPT algorithms:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$*: The key generation algorithm generates a pair of public key and secret key* $(\mathsf{pk}, \mathsf{sk})$*.*
- $(K, \psi) \leftarrow \mathsf{Encap}(\mathsf{pk})$*: The encapsulation algorithm takes a public key* pk *as input, and outputs a symmetric key* K *and a ciphertext* ψ*.*
- $K \leftarrow \mathsf{Decap}(\mathsf{sk}, \psi)$*: The decapsulation algorithm takes a secret key* sk *and a ciphertext* ψ *as input, and outputs a symmetric key* K*.*

Correctness. *For any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$ *and any* $(K, \psi) \leftarrow \mathsf{Encap}(\mathsf{pk})$*, it holds that* $\mathsf{Decap}(\mathsf{sk}, \psi) = K$*.*

IND-CPA Security. *We consider the standard indistinguishability under chosen-plaintext attacks (*$\mathsf{IND\text{-}CPA}$*) for* KEM*. More precisely, for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM},\mathcal{A}}(\kappa) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{c} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}, \\ (K_0^*, \psi^*) \leftarrow \mathsf{Encap}(\mathsf{pk}),\ K_1^* \leftarrow_\$ \mathcal{K} \\ b \leftarrow_\$ \{0,1\},\ b' \leftarrow \mathcal{A}(\mathsf{pk}, \psi^*, K_b^*) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\kappa),$$

where $\mathcal{K}$ *denotes the space of symmetric keys.*

Definition 13 (Authenticated Key Exchange). *A two-pass authenticated key exchange protocol* $\mathsf{AKE} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI})$ *consists of four PPT algorithms:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$*: The key generation algorithm generates a pair of public key and secret key* $(\mathsf{pk}, \mathsf{sk})$*.*
- $(\mathsf{msg}_i, \mathsf{st}) \leftarrow \mathsf{Init}(\mathsf{pk}_r, \mathsf{sk}_i)$*: The initialization algorithm takes a public key* pk_r *of a responder (say* P_r*) and a secret key* sk_i *of an initiator (say* P_i*) as input, and outputs an initiated message* msg_i *and a state* st *for* P_i*.*
- $(\mathsf{msg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i)$*: The derivation algorithm for the responder takes a public key* pk_i *of the initiator* P_i*, a secret key* sk_r *of the responder* P_r *and an initiated message* msg_i *as input, and outputs a message* msg_r *and a session key* K_r *for* P_r*.*
- $\mathsf{K}_i \leftarrow \mathsf{DerI}(\mathsf{pk}_r, \mathsf{sk}_i, \mathsf{msg}_r, \mathsf{st})$*: The deterministic derivation algorithm for the initiator takes a public key* pk_r *of the responder* P_r*, a secret key* sk_i *of the initiator* P_i*, a message* msg_r *and a state* st *as input, and outputs a session key* K_i *for* P_i*.*

We illustrate an execution of a two-pass AKE protocol in Fig. 7, where the key-pairs of P_i and P_r are generated at the beginning via $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}$ and $(\mathsf{pk}_r, \mathsf{sk}_r) \leftarrow \mathsf{Gen}$. Note that the initiator P_i invokes two algorithms Init and DerI, so P_i has to transmit a state st from Init to DerI. Correctness of AKE requires that for any distinct parties P_i and P_r, they share the same session key $\mathsf{K}_i = \mathsf{K}_r \neq \perp$ after the execution of the AKE protocol according to Fig. 7.

Definition 14 (DDH Assumption). *Let* $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2, g_T)$ *be a description of asymmetric pairing group, and let* $s \in \{1, 2\}$*. The DDH assumption holds over group* $\mathbb{G}_s$*, if for any PPT adversary* $\mathcal{A}$*, it holds that* $\mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_s,\mathcal{A}}(\kappa) := \left| \Pr[\mathcal{A}(\mathsf{pp}, g_s^x, g_s^y, g_s^{xy}) = 1] - \Pr[\mathcal{A}(\mathsf{pp}, g_s^x, g_s^y, g_s^z) = 1] \right| \leq \mathsf{negl}(\kappa)$*, where the probability is over* $x, y, z \leftarrow_\$ \mathbb{Z}_p$*.*

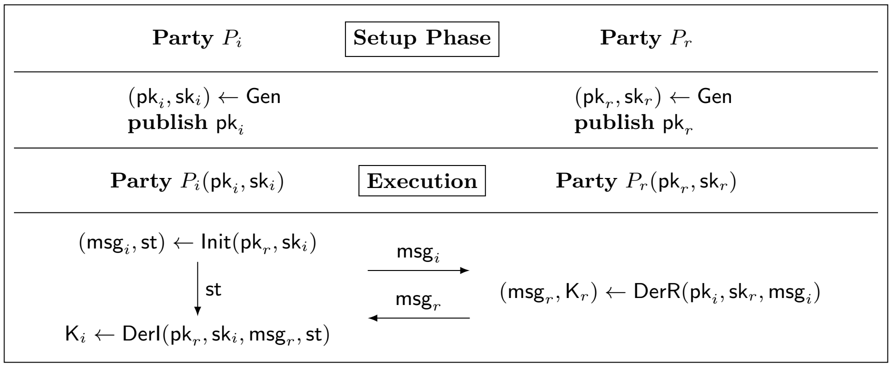

Fig. 7. An execution of two-pass AKE.

B Missing Details in Sect. 3

B.1 Correctness Requirements of AM-AKE

In this subsection, we define the correctness requirements of an AM-AKE scheme. Different requirements serve for different working modes of AM-AKE.

Definition 15 (Correctness of AM-AKE). *Let* $\mathsf{AM\text{-}AKE} = ((\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI}), (\mathsf{aGen}, \mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}))$ *be an AM-AKE scheme. We consider the correctness for its three modes.*

Correctness for the normal mode. *If both P_i and P_r invoke normal algorithms in the AKE protocol, then it results in the same session key $\mathsf{K}_i = \mathsf{K}_r$, no matter P_i (and P_r) uses normal key-pair or anamorphic key-pair. More precisely, for any $(\overline{pk}_i, \overline{sk}_i) := (\mathsf{pk}_i, \mathsf{sk}_i)$ generated by* Gen *or $(\overline{pk}_i, \overline{sk}_i) := (\mathsf{apk}_i, \mathsf{ask}_i)$ generated by* aGen*, and for any $(\overline{pk}_r, \overline{sk}_r) := (\mathsf{pk}_r, \mathsf{sk}_r)$ generated by* Gen *or $(\overline{pk}_r, \overline{sk}_r) := (\mathsf{apk}_r, \mathsf{ask}_r)$ generated by* aGen*, we have*

$$\Pr\left[\mathsf{K}_i = \mathsf{K}_r \neq \bot \;\middle|\; \begin{array}{c} (\mathsf{msg}_i, \mathsf{st}) \leftarrow \mathsf{Init}(\overline{pk}_r, \overline{sk}_i) \\ (\mathsf{msg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}(\overline{pk}_i, \overline{sk}_r, \mathsf{msg}_i) \\ \mathsf{K}_i \leftarrow \mathsf{DerI}(\overline{pk}_r, \overline{sk}_i, \mathsf{msg}_r, \mathsf{st}) \end{array}\right] = 1.$$

Correctness for the anamorphic mode. *If both P_i and P_r invoke anamorphic algorithms in the AKE protocol, then it results in the same session key $\mathsf{K}_i = \mathsf{K}_r$ and the same double key $\mathsf{dk}_i = \mathsf{dk}_r$. Meanwhile, the normal derivation by* DerI *using* ask_i *should also result in the same session key $\mathsf{K}'_i = \mathsf{K}_i = \mathsf{K}_r$. More precisely, for any $(\mathsf{apk}_i, \mathsf{ask}_i, \mathsf{aux}_i) \leftarrow \mathsf{aGen}$ and any $(\mathsf{apk}_r, \mathsf{ask}_r, \mathsf{aux}_r) \leftarrow \mathsf{aGen}$, we have*

$$\Pr\left[\begin{array}{c} \mathsf{K}_i = \mathsf{K}_r = \mathsf{K}'_i \neq \bot \\ \wedge\ \mathsf{dk}_i = \mathsf{dk}_r \neq \bot \end{array} \;\middle|\; \begin{array}{c} (\mathsf{amsg}_i, \mathsf{st}, \mathsf{aux}'_i) \leftarrow \mathsf{aInit}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}_i) \\ (\mathsf{amsg}_r, \mathsf{K}_r, \mathsf{dk}_r) \leftarrow \mathsf{aDerR}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{aux}_r, \mathsf{amsg}_i) \\ (\mathsf{K}_i, \mathsf{dk}_i) \leftarrow \mathsf{aDerI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}'_i, \mathsf{amsg}_r, \mathsf{st}) \\ \mathsf{K}'_i \leftarrow \mathsf{DerI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{amsg}_r, \mathsf{st}) \end{array}\right] = 1.$$

Correctness for the half mode. *If one party invokes normal algorithms and the other invokes anamorphic algorithms, then the half mode still results in the same session key* $\mathsf{K}_i = \mathsf{K}_r$*. Moreover, the normal derivation* DerI *using* ask_i *should also result in the same session key* $\mathsf{K}_i' = \mathsf{K}_i = \mathsf{K}_r$*. More precisely, for any* $(\mathsf{apk}_i, \mathsf{ask}_i, \mathsf{aux}_i) \leftarrow \mathsf{aGen}$*, and for any* $(\overline{pk}_r, \overline{sk}_r) := (\mathsf{pk}_r, \mathsf{sk}_r)$ *generated by* Gen *or* $(\overline{pk}_r, \overline{sk}_r) := (\mathsf{apk}_r, \mathsf{ask}_r)$ *generated by* aGen*, we have*

$$\Pr\left[\mathsf{K}_i = \mathsf{K}_r = \mathsf{K}_i' \neq \perp \,\middle|\, \begin{array}{c} (\mathsf{amsg}_i, \mathsf{st}, \mathsf{aux}_i') \leftarrow \mathsf{aInit}(\overline{pk}_r, \mathsf{ask}_i, \mathsf{aux}_i) \\ (\mathsf{msg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}(\mathsf{apk}_i, \overline{sk}_r, \mathsf{amsg}_i) \\ (\mathsf{K}_i, \mathsf{dk}_i) \leftarrow \mathsf{aDerI}(\overline{pk}_r, \mathsf{ask}_i, \mathsf{aux}_i', \mathsf{msg}_r, \mathsf{st}) \\ \mathsf{K}_i' \leftarrow \mathsf{DerI}(\overline{pk}_r, \mathsf{ask}_i, \mathsf{msg}_r, \mathsf{st}) \end{array}\right] = 1.$$

On the other hand, for any $(\mathsf{apk}_r, \mathsf{ask}_r, \mathsf{aux}_r) \leftarrow \mathsf{aGen}$*, and for any* $(\overline{pk}_i, \overline{sk}_i) := (\mathsf{pk}_i, \mathsf{sk}_i)$ *generated by* Gen *or* $(\overline{pk}_i, \overline{sk}_i) := (\mathsf{apk}_i, \mathsf{ask}_i)$ *generated by* aGen*, we have*

$$\Pr\left[\mathsf{K}_i = \mathsf{K}_r \neq \perp \,\middle|\, \begin{array}{c} (\mathsf{msg}_i, \mathsf{st}) \leftarrow \mathsf{Init}(\mathsf{apk}_r, \overline{sk}_i) \\ (\mathsf{amsg}_r, \mathsf{K}_r, \mathsf{dk}_r) \leftarrow \mathsf{aDerR}(\overline{pk}_i, \mathsf{ask}_r, \mathsf{aux}_r, \mathsf{msg}_i) \\ \mathsf{K}_i \leftarrow \mathsf{DerI}(\mathsf{apk}_r, \overline{sk}_i, \mathsf{amsg}_r, \mathsf{st}) \end{array}\right] = 1.$$

B.2 Proof of Impossibility Results for Plain AM-AKE

In this subsection, we present the formal proofs of the three impossibility results for (two-pass) plain AM-AKE.

Theorem 1 *It is impossible for a two-pass plain AM-AKE scheme* $\mathsf{AM\text{-}AKE}$ *to achieve responder-robustness.*

Proof of Theorem 1. For a two-pass plain AM-AKE scheme $\mathsf{AM\text{-}AKE}$, to achieve responder-robustness, P_r has to make the decision whether $\mathsf{dk}_r \neq \perp$ or $\mathsf{dk}_r = \perp$ upon receiving the first pass message msg_i from P_i.

We first claim that P_r cannot achieve robustness only with the help of $(\mathsf{apk}_r, \mathsf{ask}_r)$. Note that the basic requirement for $\mathsf{AM\text{-}AKE}$ is that any adversary obtaining secret keys of P_i and P_r and seeing the transcripts of $\mathsf{AM\text{-}AKE}$ cannot tell whether $\mathsf{AM\text{-}AKE}$ works in the normal mode or in the anamorphic mode or in the half mode[6].

Suppose on the contradiction, with only $(\mathsf{apk}_r, \mathsf{ask}_r)$, P_r can output $\mathsf{dk}_r \neq \perp$ when msg_i is an anamorphic first-pass message and output $\mathsf{dk}_r = \perp$ when msg_i is a normal one, with overwhelming probability. Then the adversary who obtains ask_r is also able to do that, and thus determine msg_i is a normal one if $\mathsf{dk}_r = \perp$ and an anamorphic one if $\mathsf{dk}_r \neq \perp$, breaking the security of $\mathsf{AM\text{-}AKE}$.

Therefore, P_r must also resort to aux_r for deciding $\mathsf{dk}_r \neq \perp$ or $\mathsf{dk}_r = \perp$. However, the generation of msg_i does not involve aux_r and is independent of aux_r when apk_r is independent of aux_r. If aux_r can be generated independently from $(\mathsf{apk}_r, \mathsf{ask}_r)$, then the adversary can generate an auxiliary message $\widetilde{\mathsf{aux}}_r$

[6] See the security requirements of AM-AKE formalized in Subsect. 3.3.

independently by itself and use $\widetilde{\mathsf{aux}}_r$ to decide $\mathsf{dk}_r \neq \perp$ or $\mathsf{dk}_r = \perp$, which breaks the security of $\mathsf{AM\text{-}AKE}$ as well.

In conclusion, it is impossible to achieve responder-robustness in a two-pass plain AM-AKE where $(\mathsf{apk}_r, \mathsf{ask}_r)$ and aux_r are generated independently. $\square$

Remark. Theorem 1 does *not* apply for initiator-robustness, because P_i can hide some information of aux_i into msg_i, from which msg_r is computed, and thus msg_r may carry some information of aux_i. In this way, P_i can use aux_i to judge whether msg_r is a normal message or an anamorphic one, while an adversary may not be able to use this method to detect the using of anamorphic algorithms, thus initiator-robustness can be achieved. However, our next theorem shows that it is impossible for a plain AM-AKE to achieve initiator-robustness and $\mathsf{IND\text{-}WM}$ security simultaneously.

Theorem 2 *If a plain AM-AKE scheme* $\mathsf{AM\text{-}AKE}$ *is initiator-robust, then it is impossible for* $\mathsf{AM\text{-}AKE}$ *to achieve the* $\mathsf{IND\text{-}WM}/\mathsf{sIND\text{-}WM}$ *security.*

Proof of Theorem 2. For a (two-pass) plain AM-AKE scheme $\mathsf{AM\text{-}AKE}$ that has initiator-robustness, we construct an adversary $\mathcal{A}$ to break its $\mathsf{IND\text{-}WM}/\mathsf{sIND\text{-}WM}$ security as follows. At the beginning of the experiment, $\mathcal{A}$ receives public/secret key-pairs of all parties $(\mathsf{apk}_i, \mathsf{ask}_i)$ from the challenger. Then $\mathcal{A}$ chooses an arbitrary pair of parties (P_i, P_r) and generates the auxiliary message $\widetilde{\mathsf{aux}}_i$ of party P_i by itself, since $\mathsf{AM\text{-}AKE}$ is a plain one. Next, $\mathcal{A}$ uses $(\mathsf{apk}_i, \mathsf{ask}_i, \widetilde{\mathsf{aux}}_i)$ to impersonate P_i and interacts with P_r as follows:

- Firstly, $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{New}}(i, r)$ to start a session sID.
- Then $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID}, \mathsf{w}_\mathsf{I} = \mathbf{A})$ but discards the answer, and instead, $\mathcal{A}$ generates another amsg_i itself by invoking $(\mathsf{amsg}_i, \mathsf{st}, \mathsf{aux}'_i) \leftarrow \mathsf{aInit}(\mathsf{apk}_r, \mathsf{ask}_i, \widetilde{\mathsf{aux}}_i)$.
- Next, $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, \mathsf{amsg}_i, \mathsf{w}_\mathsf{R} = \mathbf{A})$ and receives a message amsg_r.
- Finally, $\mathcal{A}$ invokes $(\mathsf{K}_i, \mathsf{dk}_i) \leftarrow \mathsf{aDerI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}'_i, \mathsf{amsg}_r, \mathsf{st})$, and outputs $b' = 1$ to its own challenger if and only if $\mathsf{dk}_i \neq \perp$ holds.

Note that in the $\mathsf{IND\text{-}WM}/\mathsf{sIND\text{-}WM}$ experiment (cf. Fig. 2), if $b = 1$, then $\mathcal{A}$ will receive an anamorphic message amsg_r generated by aDerR from its $\mathcal{O}_{\mathsf{DerR}}$ query, so both $\mathcal{A}$ (impersonating P_i) and P_r invoke anamorphic algorithms, and consequently, $\mathsf{dk}_i \neq \perp$ holds with probability 1 by the correctness of the anamorphic mode of $\mathsf{AM\text{-}AKE}$. If $b = 0$, then $\mathcal{A}$ will receive a normal message generated by DerR from the $\mathcal{O}_{\mathsf{DerR}}$ query, so $\mathcal{A}$ (impersonating P_i) invokes anamorphic algorithms while P_r invokes normal algorithm, and consequently, $\mathsf{dk}_i = \perp$ holds with overwhelming probability according to the initiator-robustness of $\mathsf{AM\text{-}AKE}$.

Overall, $\mathcal{A}$ guesses the value of b correctly with overwhelming probability by checking whether $\mathsf{dk}_i \neq \perp$, and thus breaks the $\mathsf{IND\text{-}WM}/\mathsf{sIND\text{-}WM}$ security. $\square$

Theorem 3 *It is impossible for a plain AM-AKE scheme* $\mathsf{AM\text{-}AKE}$ *to achieve the* $\mathsf{PR\text{-}DK}/\mathsf{sPR\text{-}DK}$ *security.*

Proof of Theorem 3. For a (two-pass) plain AM-AKE scheme $\mathsf{AM\text{-}AKE}$, we construct an adversary $\mathcal{A}$ to break its $\mathsf{PR\text{-}DK}/\mathsf{sPR\text{-}DK}$ security as follows. At the beginning of the experiment, $\mathcal{A}$ receives public/secret key-pairs of all parties $(\mathsf{apk}_i, \mathsf{ask}_i)$ from the challenger. Then $\mathcal{A}$ chooses an arbitrary pair of parties (P_i, P_r) and generates the auxiliary message $\widetilde{\mathsf{aux}}_i$ of party P_i by itself, since $\mathsf{AM\text{-}AKE}$ is a plain one. Next, $\mathcal{A}$ uses $(\mathsf{apk}_i, \mathsf{ask}_i, \widetilde{\mathsf{aux}}_i)$ to impersonate P_i and interacts with P_r as follows:

- Firstly, $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{New}}(i, r)$ to start a session sID.
- Then $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID})$ but discards the answer, and instead, $\mathcal{A}$ generates another amsg_i itself by invoking $(\mathsf{amsg}_i, \mathsf{st}, \mathsf{aux}'_i) \leftarrow \mathsf{aInit}(\mathsf{apk}_r, \mathsf{ask}_i, \widetilde{\mathsf{aux}}_i)$.
- Next, $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, \mathsf{amsg}_i)$ and receives a message amsg_r.
- Moreover, $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{TestDK}}(\mathsf{sID}, \mathsf{R})$ and receives a double key dk^*_r, which is either the real double key generated in the above $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, \mathsf{amsg}_i)$ oracle or a uniformly random double key.
- Finally, $\mathcal{A}$ invokes $(\mathsf{K}_i, \mathsf{dk}_i) \leftarrow \mathsf{aDerI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}'_i, \mathsf{amsg}_r, \mathsf{st})$, and outputs $b' = 1$ to its own challenger if and only if $\mathsf{dk}_i = \mathsf{dk}^*_r$ holds.

Note that in the $\mathsf{PR\text{-}DK}/\mathsf{sPR\text{-}DK}$ experiment (cf. Fig. 3), the oracle $\mathcal{O}_{\mathsf{DerR}}$ generates amsg_r by invoking aDerR, so both $\mathcal{A}$ (impersonating P_i) and P_r invoke anamorphic algorithms. If $b = 1$, the dk^*_r outputted by $\mathcal{O}_{\mathsf{TestDK}}$ is the real double key generated in the above $\mathcal{O}_{\mathsf{DerR}}$ oracle, and then the dk_i derived by $\mathcal{A}$ satisfies $\mathsf{dk}_i = \mathsf{dk}^*_r$ with probability 1 by the correctness of the anamorphic mode of $\mathsf{AM\text{-}AKE}$. If $b = 0$, dk^*_r is uniformly chosen from $\mathcal{DK}$, then $\mathsf{dk}_i = \mathsf{dk}^*_r$ holds with probability at most $1/|\mathcal{DK}|$ where $\mathcal{DK}$ is the double key space.

Overall, $\mathcal{A}$ guesses the value of b correctly with probability at least $\frac{1}{2} \cdot (1 - 1/|\mathcal{DK}|)$, which is non-negligible for $|\mathcal{DK}| \geq 2$, by checking whether $\mathsf{dk}_i = \mathsf{dk}^*_r$, and thus breaks the $\mathsf{PR\text{-}DK}/\mathsf{sPR\text{-}DK}$ security. □

C Missing Proofs in Sect. 4 (Generic Construction of AM-AKE)

C.1 Proof of Lemma 2 (sIND-WM Security of AM-AKE)

Lemma 2 *There exists PPT simulator* $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$*, such that for any PPT adversary* $\mathcal{A}$ *and* $N = \mathsf{poly}(\kappa)$*,* $\left|\Pr\left[\mathsf{Exp}^{\mathsf{sIND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N} = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\kappa)$.

Proof of Lemma 2. We first describe the simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$.

- $\underline{R_i \leftarrow \mathsf{SimI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}_i = \mathsf{msk}_i, R'_i)}$: Here R'_i is an internal randomness used in aInit, and thus includes $d_{i,1}$ as well as the randomness used in $\overline{\mathsf{Init}}$, denoted by $d_{i,3}$, i.e., $R'_i = (d_{i,1}, d_{i,3})$. This algorithm aims to explain R'_i as a randomness R_i for Init. To this end, it computes $s_i := \mathsf{Extract}(\mathsf{msk}_i, \mathsf{apk}_r)$, $m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$, $d_{i,2} := \mathsf{PRF}_{\mathsf{I}}(s_i, m_{i,1})$, and outputs $R_i := (d_{i,1}, d_{i,2}, d_{i,3})$.
- $\underline{R_r \leftarrow \mathsf{SimR}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{aux}_r = \mathsf{msk}_r, m, R'_r)}$: Here R'_r is an internal randomness used in aDerR, and thus includes $d_{r,1}$ as well as the randomness used in $\overline{\mathsf{DerR}}$, denoted by $d_{r,3}$, i.e., $R'_r = (d_{r,1}, d_{r,3})$. This algorithm aims to explain R'_r as a randomness R_r for DerR. To this end, it parses $m = (m_{i,1}, m_{i,2}, m_{i,3})$, computes $s_r := \mathsf{Extract}(\mathsf{msk}_r, \mathsf{apk}_i)$, $m_{r,1} := f_{\mathsf{R},1}(d_{r,1})$, $d_{r,2} := \mathsf{PRF}_{\mathsf{R}}(s_r, (m_{i,1}, m_{r,1}))$ and outputs $R_r := (d_{r,1}, d_{r,2}, d_{r3})$.

We prove the lemma via a sequence of games G_0-G_3, where the differences between adjacent games are highlighted in gray boxes.

Game G_0: This is the $\mathsf{Exp}^{\mathsf{sIND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}$ experiment (cf. Fig. 2). Then we have $\Pr\left[\mathsf{Exp}^{\mathsf{sIND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N} = 1\right] = \Pr[\mathsf{G}_0 = 1]$.

In this game, the challenger samples a challenge bit $b \leftarrow_\$ \{0,1\}$, and answers the $\mathcal{O}_{\mathsf{Init}}, \mathcal{O}_{\mathsf{DerR}}, \mathcal{O}_{\mathsf{DerI}}$ queries for $\mathcal{A}$ in the following way:

– If $b = 0$, the challenger invokes the normal algorithms $\mathsf{Init}, \mathsf{DerR}, \mathsf{DerI}$;
– If $b = 1$ and $\mathcal{A}$ designates normal mode (i.e., $\mathbf{N}$), the challenger also invokes the normal algorithms;
– If $b = 1$ and $\mathcal{A}$ designates anamorphic mode (i.e., $\mathbf{A}$), the challenger invokes the anamorphic algorithm $\mathsf{aInit}/\mathsf{aDerR}/\mathsf{aDerI}$ and the simulator $\mathsf{SimI}/\mathsf{SimR}$.

The adversary $\mathcal{A}$ succeeds if it guesses b correctly. Overall, there are differences between $b = 0$ and $b = 1$ only if $\mathcal{A}$ designates anamorphic mode (i.e., $\mathbf{A}$).

We note that the oracles $\mathcal{O}_{\mathsf{Init}}, \mathcal{O}_{\mathsf{DerR}}, \mathcal{O}_{\mathsf{DerI}}$ output $(\mathsf{msg}_i, \mathsf{st}, R_i)$, $(\mathsf{msg}_r, \mathsf{K}_r, R_r)$ and K_i, respectively, but do not output the double keys $\mathsf{dk}_i, \mathsf{dk}_r$. The differences between the normal algorithms and the anamorphic algorithms and simulator in generating these values only lie in the distributions of $d_{i,2}$ and $d_{r,2}$:

– The normal algorithms $\mathsf{Init}, \mathsf{DerR}, \mathsf{DerI}$ use uniformly chosen coins $d_{i,2} \leftarrow_\$ \mathcal{D}_{\mathsf{I},2}$ and $d_{r,2} \leftarrow_\$ \mathcal{D}_{\mathsf{R},2}$.
– The anamorphic algorithms $\mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}$ and simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$ involve specific coins $d_{i,2} := \mathsf{PRF}_{\mathsf{I}}(s_i, m_{i,1}) \in \mathcal{D}_{\mathsf{I},2}$ and $d_{r,2} := \mathsf{PRF}_{\mathsf{R}}(s_r, (m_{i,1}, m_{r,1})) \in \mathcal{D}_{\mathsf{R},2}$, where $s_i := \mathsf{Extract}(\mathsf{msk}_i, \mathsf{apk}_r)$, $m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$ for $d_{i,1} \leftarrow_\$ \mathcal{D}_{\mathsf{I},1}$, $s_r := \mathsf{Extract}(\mathsf{msk}_r, \mathsf{apk}_i)$, $m_{r,1} := f_{\mathsf{R},1}(d_{r,1})$ for $d_{r,1} \leftarrow_\$ \mathcal{D}_{\mathsf{R},1}$.

Game G_1: It is the same as G_0, except that at the beginning of the game, the challenger samples $\boxed{s^*_{i,r} = s^*_{r,i} \leftarrow_\$ \mathcal{D}_{\mathsf{E}}}$ for each pair of parties $(i,r) \in [N] \times [N]$ with $i \neq r$. Then in the case of $b = 1$ and the mode designated by $\mathcal{A}$ is anamorphic (i.e., $\mathbf{A}$), the challenger answers the oracle queries $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID}, \mathsf{w}_{\mathsf{I}} = \mathbf{A}), \mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m, \mathsf{w}_{\mathsf{R}} = \mathbf{A}), \mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ for $\mathcal{A}$ as follows:

– The challenger invokes anamorphic algorithms $\mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}$ and the simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$, by using specific coins $d_{i,2} := \mathsf{PRF}_{\mathsf{I}}(s_i, m_{i,1}) \in \mathcal{D}_{\mathsf{I},2}$ and $d_{r,2} := \mathsf{PRF}_{\mathsf{R}}(s_r, (m_{i,1}, m_{r,1})) \in \mathcal{D}_{\mathsf{R},2}$, where $\boxed{s_i := s^*_{i,r}}$, $m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$ for $d_{i,1} \leftarrow_\$ \mathcal{D}_{\mathsf{I},1}$, $\boxed{s_r := s^*_{r,i}}$, $m_{r,1} := f_{\mathsf{R},1}(d_{r,1})$ for $d_{r,1} \leftarrow_\$ \mathcal{D}_{\mathsf{R},1}$. Here $(i,r) := (\mathrm{init}[\mathsf{sID}], \mathrm{resp}[\mathsf{sID}])$ denote the initiator and responder of sID.

By the secret extractability of Gen (cf. Def. 9), the secrets $s_i := \mathsf{Extract}(\mathsf{msk}_i, \mathsf{apk}_r)$ and $s_r := \mathsf{Extract}(\mathsf{msk}_r, \mathsf{apk}_i)$ generated in G_0 are computationally indistinguishable from the $\boxed{s_i := s^*_{i,r}}$, $\boxed{s_r := s^*_{r,i}}$ with $\boxed{s^*_{i,r} = s^*_{r,i} \leftarrow_\$ \mathcal{D}_{\mathsf{E}}}$ in G_1. More precisely, we have the following claim via hybrid arguments over all pair of parties $(i,r) \in [N] \times [N]$ with $i \neq r$, and we postpone its formal proof to Appendix C.2.

Claim 1. By the secret extractability of Gen, $\left|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]\right| \leq \mathsf{negl}(\kappa)$.

Game G_2: It is the same as G_1, except that the challenger replaces the pseudo-random function $\mathsf{PRF} = (\mathsf{PRF_I}, \mathsf{PRF_R}, \mathsf{PRF_D})$ with truly random function $\boxed{\mathsf{TRF} = (\mathsf{TRF_I}, \mathsf{TRF_R}, \mathsf{TRF_D})}$, where $\mathsf{TRF_I}/\mathsf{TRF_R}/\mathsf{TRF_D} : \{0,1\}^* \longrightarrow \mathcal{D}_{\mathsf{I},2}/\mathcal{D}_{\mathsf{R},2}/\{0,1\}^\kappa$. More precisely, when $b = 1$ and the mode designated by $\mathcal{A}$ is anamorphic (i.e., $\mathbf{A}$), the challenger answers the oracle queries $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID}, \mathsf{w_I} = \mathbf{A}), \mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m, \mathsf{w_R} = \mathbf{A}), \mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ for $\mathcal{A}$ as follows:

– The challenger invokes anamorphic algorithms $\mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}$ and the simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$, by using specific coins $d_{i,2} := \boxed{\mathsf{TRF_I}}(s^*_{i,r}, m_{i,1}) \in \mathcal{D}_{\mathsf{I},2}$ and $d_{r,2} := \boxed{\mathsf{TRF_R}}(s^*_{r,i}, (m_{i,1}, m_{r,1})) \in \mathcal{D}_{\mathsf{R},2}$, where $m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$ for $d_{i,1} \leftarrow_\$ \mathcal{D}_{\mathsf{I},1}$, $m_{r,1} := f_{\mathsf{R},1}(d_{r,1})$ for $d_{r,1} \leftarrow_\$ \mathcal{D}_{\mathsf{R},1}$.

Since $\mathsf{PRF} = (\mathsf{PRF_I}, \mathsf{PRF_R}, \mathsf{PRF_D})$ is a pseudo-random function, its outputs are computationally indistinguishable from the outputs of truly random function $\mathsf{TRF} = (\mathsf{TRF_I}, \mathsf{TRF_R}, \mathsf{TRF_D})$. Consequently, this change is unnoticeable to $\mathcal{A}$, and by a standard hybrid argument over the PRF keys $s^*_{i,r}, s^*_{r,i}$, we have $\big|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]\big| \le \mathsf{negl}(\kappa)$.

Game G_3: It is the same as G_2, except that in the case of $b = 1$ and the mode designated by $\mathcal{A}$ is anamorphic (i.e., $\mathbf{A}$), the challenger answers the oracle queries $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID}, \mathsf{w_I} = \mathbf{A}), \mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m, \mathsf{w_R} = \mathbf{A}), \mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ for $\mathcal{A}$ as follows:

– The challenger invokes anamorphic algorithms $\mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}$ and the simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$, by using uniformly chosen coins $\boxed{d_{i,2} \leftarrow_\$ \mathcal{D}_{\mathsf{I},2}}$ and $\boxed{d_{r,2} \leftarrow_\$ \mathcal{D}_{\mathsf{R},2}}$.

Clearly, in G_3, the anamorphic algorithms $\mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}$ and the simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$ using uniformly chosen coins $\boxed{d_{i,2} \leftarrow_\$ \mathcal{D}_{\mathsf{I},2}}$ and $\boxed{d_{r,2} \leftarrow_\$ \mathcal{D}_{\mathsf{R},2}}$ are essentially the same as the normal algorithms $\mathsf{Init}, \mathsf{DerR}, \mathsf{DerI}$ in generating the responses $(\mathsf{msg}_i, \mathsf{st}, R_i)$, $(\mathsf{msg}_r, \mathsf{K}_r, R_r)$ and K_i, so the challenge bit b is perfectly hidden to $\mathcal{A}$, and we have $\Pr[\mathsf{G}_3 = 1] = 1/2$.

It remains to show that G_2 and G_3 are computationally indistinguishable for $\mathcal{A}$. Let E_I denote the event that in G_2, all invocations of $d_{i,2} := \boxed{\mathsf{TRF_I}}(s^*_{i,r}, m_{i,1}) \in \mathcal{D}_{\mathsf{I},2}$ are on different inputs $m_{i,1}$, and let E_R denote the event that all invocations of $d_{r,2} := \boxed{\mathsf{TRF_R}}(s^*_{r,i}, (m_{i,1}, m_{r,1})) \in \mathcal{D}_{\mathsf{R},2}$ are on different inputs $(m_{i,1}, m_{r,1})$. If both E_I and E_R occur, then in G_2, $\boxed{\mathsf{TRF_I}}$ and $\boxed{\mathsf{TRF_R}}$ are always computed on different inputs, so their outputs $d_{i,2}, d_{r,2}$ are uniformly and independently distributed, the same as those in G_3. Consequently, we have

$$\big|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_3 = 1]\big| \le \Pr[\neg E_\mathsf{I} \vee \neg E_\mathsf{R}] \le \Pr[\neg E_\mathsf{I}] + \Pr[\neg E_\mathsf{R}].$$

On the other hand, in G_2, each input $m_{i,1}$ of $\boxed{\mathsf{TRF_I}}$ is generated by $m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$ for $d_{i,1} \leftarrow_\$ \mathcal{D}_{\mathsf{I},1}$, so the inputs can hardly collide by the entropy-preserving property of $f_{\mathsf{I},1}$ (cf. Def. 7), and we have $\Pr[\neg E_\mathsf{I}] \le \mathsf{negl}(\kappa)$. Similarly, each input $(m_{i,1}, m_{r,1})$ of $\boxed{\mathsf{TRF_R}}$ involves $m_{r,1} := f_{\mathsf{R},1}(d_{r,1})$ for $d_{r,1} \leftarrow_\$ \mathcal{D}_{\mathsf{R},1}$, so the inputs can hardly collide by the entropy-preserving property of $f_{\mathsf{R},1}$, and we have $\Pr[\neg E_{\mathsf{R}}] \leq \mathsf{negl}(\kappa)$. This shows that $\big|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_3 = 1]\big| \leq \mathsf{negl}(\kappa)$.

Finally, by taking all things together, Lemma 2 follows. □

C.2 Proof of Claim 1

Claim 1. By the secret extractability of Gen*,* $\big|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]\big| \leq \mathsf{negl}(\kappa)$*.*

Proof. We first define an intermediate hybrid game $\mathsf{G}_{0.5}$, where it makes the same change as that from G_0 to G_1, except that the change is made only for a single pair of parties $(i, r) \in [N] \times [N]$ with $i \neq r$. We will show that $\mathsf{G}_{0.5}$ is indistinguishable from G_0, and then by a hybrid argument over all pair of parties, it follows that G_1 is also indistinguishable from G_0.

More precisely, let $(i, r) \in [N] \times [N]$ with $i \neq r$ be an arbitrary and fixed pair of parties. In $\mathsf{G}_{0.5}$, we make the following changes:

- Sample $s^*_{i,r} = s^*_{r,i} \leftarrow_{\$} \mathcal{D}_{\mathsf{E}}$ at the beginning of the game and store it.
- In the case of $b = 1$ and the mode designated by $\mathcal{A}$ is anamorphic (i.e., $\mathbf{A}$), the adversary queries the oracle $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID}, \mathsf{w_I} = \mathbf{A})$, $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m, \mathsf{w_R} = \mathbf{A})$ or $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ where $(i, r) = (\mathrm{init}[\mathsf{sID}], \mathrm{resp}[\mathsf{sID}])$, then the challenger answers as follows:
  - The challenger invokes anamorphic algorithms aInit,aDerR,aDerI and the simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$ by using specific coins $d_{i,2} := \mathsf{PRF_I}(s_i, m_{i,1}) \in \mathcal{D}_{\mathsf{I},2}$ and $d_{r,2} := \mathsf{PRF_R}(s_r, m_{i,1}, m_{r,1}) \in \mathcal{D}_{\mathsf{R},2}$, where $s_i := s^*_{i,r}$, $m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$ for $d_{i,1} \leftarrow_{\$} \mathcal{D}_{\mathsf{I},1}$, $s_r := s^*_{r,i}$, $m_{r,1} := f_{\mathsf{R},1}(d_{r,1})$ for $d_{r_k,1} \leftarrow_{\$} \mathcal{D}_{\mathsf{R},1}$.

Notice that the only difference between $\mathsf{G}_{0.5}$ and G_0 lies in the choices of s_i and s_r, so we make the following analysis centered on the generation of s_i and s_r. Suppose there exists an adversary $\mathcal{A}$ who can distinguish $\mathsf{G}_{0.5}$ from G_0, then we construct an adversary $\mathcal{B}$ to break the secret extractability (specifically, the property of the pseudo-random of the extracting) of Gen as per Definition 9.

At the beginning, $\mathcal{B}$ receives $(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{sk}_i, \mathsf{sk}_r, s^*)$ from its challenger where s^* equals to either $\mathsf{Extract}(\mathsf{msk}_i, \mathsf{pk}_r)$ or a random value from $\mathcal{D}_{\mathsf{E}}$. To simulate for $\mathcal{A}$, $\mathcal{B}$ generates $N-2$ anamorphic key-pairs itself by $(\mathsf{apk}_n, \mathsf{ask}_n, \mathsf{aux}_n = \mathsf{msk}_n) \leftarrow \mathsf{aGen}$ for $n \in [N] \setminus \{i, r\}$, and sets $(\mathsf{apk}_i, \mathsf{ask}_i) := (\mathsf{pk}_i, \mathsf{sk}_i)$, $(\mathsf{apk}_r, \mathsf{ask}_r) := (\mathsf{pk}_r, \mathsf{sk}_r)$, then sends the N anamorphic key-pairs to $\mathcal{A}$. When $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{New}}(i, r)$, $\mathcal{B}$ can perfectly follow all the steps and simulate the response to $\mathcal{A}$.

In the case of $b = 1$, suppose the mode designated by $\mathcal{A}$ is anamorphic (i.e., $\mathbf{A}$), and $\mathcal{A}$ queries the oracle $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID}, \mathsf{w_I} = \mathbf{A})$, $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m, \mathsf{w_R} = \mathbf{A})$ or $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$, then let $(i', r') := (\mathrm{init}[\mathsf{sID}], \mathrm{resp}[\mathsf{sID}])$. We have the following cases to consider.

- Suppose $\mathcal{A}$ queries oracle $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID}, \mathsf{w_I} = \mathbf{A})$ or $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$:
  1. If $i \neq i'$, then $\mathcal{B}$ can perfectly simulate the response, because it owns msk_i, and successfully generates $s_i := \mathsf{Extract}(\mathsf{msk}_i, \mathsf{apk}_r)$.

2. If $i = i'$ and $r \neq r'$, then $\mathcal{B}$ follows the process except for setting $s_i := \mathsf{Extract}(\mathsf{msk}_{r'}, \mathsf{apk}_i)$. By the extracting correctness requirement of the secret extractability of Gen, it always holds that $\mathsf{Extract}(\mathsf{msk}_{r'}, \mathsf{apk}_i) = \mathsf{Extract}(\mathsf{msk}_i, \mathsf{apk}_{r'})$, so this change is perfectly unknown to $\mathcal{A}$.
3. If $i = i'$ and $r = r'$, then $\mathcal{B}$ follows the whole process except for setting $s_i := s^*$.

– Suppose $\mathcal{A}$ queries oracle $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m, \mathsf{w_R} = \mathbf{A})$:
  1. If $r \neq r'$, then $\mathcal{B}$ can perfectly simulate the response, because it owns msk_r, and successfully generates $s_r := \mathsf{Extract}(\mathsf{msk}_r, \mathsf{apk}_i)$.
  2. If $r = r'$ and $i \neq i'$, then $\mathcal{B}$ follows the process except for setting $s_r := \mathsf{Extract}(\mathsf{msk}_{i'}, \mathsf{apk}_r)$. Still by the extracting correctness requirement of the secret extractability of Gen, it always holds that $\mathsf{Extract}(\mathsf{msk}_{i'}, \mathsf{apk}_r) = \mathsf{Extract}(\mathsf{msk}_r, \mathsf{apk}_{i'})$, so this change is perfectly unknown to $\mathcal{A}$.
  3. If $r = r'$ and $r = i'$, then $\mathcal{B}$ follows the whole process except for setting $s_r := s^*$.

Now, if s^* equals to $\mathsf{Extract}(\mathsf{msk}_i, \mathsf{pk}_r)$, by the extracting correctness requirement of the secret extractability of Gen, s^* also equals to $\mathsf{Extract}(\mathsf{msk}_r, \mathsf{pk}_i)$, then $\mathcal{B}$ simulates G_0 for $\mathcal{A}$. If s^* equals to a random string, then $\mathcal{B}$ simulates $\mathsf{G}_{0.5}$ for $\mathcal{A}$. Therefore,

$$\begin{aligned}&\Big|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_{0.5} = 1]\Big|\\ =&\Big|\Pr[\mathcal{B}(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{sk}_i, \mathsf{sk}_r, \mathsf{Extract}(\mathsf{msk}_i, \mathsf{pk}_r)) = 1] - \Pr[\mathcal{B}(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{sk}_i, \mathsf{sk}_r, s^* \leftarrow_\$ \mathcal{D}_\mathsf{E}) = 1]\Big|\\ \leq&\mathsf{negl}(\kappa). \blacksquare\end{aligned}$$

C.3 Proof of Lemma 3 (sPR-DK Security of AM-AKE)

Lemma 3 *There exists PPT simulator* $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$*, such that for any PPT adversary* $\mathcal{A}$ *and* $N = \mathsf{poly}(\kappa)$*,* $\left|\Pr\left[\mathsf{Exp}^{\mathsf{sPR\text{-}DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N} = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\kappa)$.

Proof of Lemma 3. We adopt the same simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$ defined in the proof of Lemma 2. We prove the lemma via a sequence of games G'_0-G'_3, which are defined similarly as those G_0-G_3 in the proof of Lemma 2.

Game G'_0: This is the $\mathsf{Exp}^{\mathsf{sPR\text{-}DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}$ experiment (cf. Fig. 3). Then we have $\Pr\left[\mathsf{Exp}^{\mathsf{sPR\text{-}DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N} = 1\right] = \Pr[\mathsf{G}'_0 = 1]$.

In this game, the challenger samples a challenge bit $b \leftarrow_\$ \{0, 1\}$, and answers the $\mathcal{O}_{\mathsf{Init}}, \mathcal{O}_{\mathsf{DerR}}, \mathcal{O}_{\mathsf{DerI}}$ queries for $\mathcal{A}$ by invoking the anamorphic algorithms $\mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}$ and the simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$. Moreover, the challenger answers the $\mathcal{O}_{\mathsf{TestDK}}$ queries for $\mathcal{A}$, by returning the real double keys dk_r (resp., dk_i) generated in $\mathcal{O}_{\mathsf{DerR}}$ (resp., $\mathcal{O}_{\mathsf{DerI}}$) if $b = 1$ while returning uniformly chosen $\mathsf{dk} \leftarrow_\$ \{0, 1\}^\kappa$ if $b = 0$. Note that if the dk_r (resp., dk_i) generated in $\mathcal{O}_{\mathsf{DerR}}$ (resp., $\mathcal{O}_{\mathsf{DerI}}$) is invalid (i.e., equals $\perp$), then the challenger will output $\perp$ directly for the $\mathcal{O}_{\mathsf{TestDK}}$ query regardless of the value of b. The adversary $\mathcal{A}$ succeeds if it guesses b correctly.

We note that the oracles $\mathcal{O}_{\mathsf{DerR}}$ and $\mathcal{O}_{\mathsf{DerI}}$ generate the real *valid* double keys dk_r and dk_i according to aDerR and aDerI as follows:

- $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$ generates *valid* dk_r by setting $\mathsf{dk}_r := \mathsf{PRF}_{\mathsf{D}}(s_r, (m, \mathsf{amsg}_r)) \in \{0,1\}^\kappa$, where $s_r := \mathsf{Extract}(\mathsf{msk}_r, \mathsf{apk}_i)$ and $\mathsf{amsg}_r := (m_{r,1}, m_{r,2}, m_{r,3})$ with $m_{r,1} := f_{\mathsf{R},1}(d_{r,1})$ for $d_{r,1} \leftarrow_\$ \mathcal{D}_{\mathsf{R},1}$.
- $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ generates *valid* dk_i by setting $\mathsf{dk}_i := \mathsf{PRF}_{\mathsf{D}}(s_i, (\mathsf{amsg}_i, m)) \in \{0,1\}^\kappa$, where $s_i := \mathsf{Extract}(\mathsf{msk}_i, \mathsf{apk}_r)$ and $\mathsf{amsg}_i := (m_{i,1}, m_{i,2}, m_{i,3})$ with $m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$ for $d_{i,1} \leftarrow_\$ \mathcal{D}_{\mathsf{I},1}$, which are generated during the $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID})$ query and stored in $Aux[\mathsf{sID}] = \mathsf{aux}'_i = (s_i, \mathsf{amsg}_i)$.

Here $(i, r) := (\text{init}[\mathsf{sID}], \text{resp}[\mathsf{sID}])$ denote the initiator and responder of sID.

Game G'_1: It is the same as G'_0, except that at the beginning of the game, the challenger samples $\boxed{s^*_{i,r} = s^*_{r,i} \leftarrow_\$ \mathcal{D}_{\mathsf{E}}}$ for each pair of parties $(i, r) \in [N] \times [N]$ with $i \neq r$. Then the challenger answers the oracle queries $\mathcal{O}_{\mathsf{Init}}, \mathcal{O}_{\mathsf{DerR}}, \mathcal{O}_{\mathsf{DerI}}$ by using $\boxed{s_i := s^*_{i,r}}$ and $\boxed{s_r := s^*_{r,i}}$ as the secrets, instead of $s_i := \mathsf{Extract}(\mathsf{msk}_i, \mathsf{apk}_r)$ and $s_r := \mathsf{Extract}(\mathsf{msk}_r, \mathsf{apk}_i)$. Especially, now the oracles $\mathcal{O}_{\mathsf{DerR}}$ and $\mathcal{O}_{\mathsf{DerI}}$ generate the real valid double keys dk_r and dk_i as follows:

- $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$ generates valid dk_r by setting $\mathsf{dk}_r := \mathsf{PRF}_{\mathsf{D}}(s_r, (m, \mathsf{amsg}_r))$, where $\boxed{s_r := s^*_{r,i}}$ and $\mathsf{amsg}_r := (m_{r,1}, m_{r,2}, m_{r,3})$ with $m_{r,1} := f_{\mathsf{R},1}(d_{r,1})$ for $d_{r,1} \leftarrow_\$ \mathcal{D}_{\mathsf{R},1}$.
- $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ generates valid dk_i by setting $\mathsf{dk}_i := \mathsf{PRF}_{\mathsf{D}}(s_i, (\mathsf{amsg}_i, m))$, where $\boxed{s_i := s^*_{i,r}}$ and $\mathsf{amsg}_i := (m_{i,1}, m_{i,2}, m_{i,3})$ with $m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$ for $d_{i,1} \leftarrow_\$ \mathcal{D}_{\mathsf{I},1}$.

Similar to the game transition from G_0 to G_1 in the proof of Lemma 2, G'_0 and G'_1 are indistinguishable by the secret extractability of Gen, and we have the following claim whose proof is essentially the same as that for Claim 1.

Claim 2. By the secret extractability of Gen, $\left|\Pr[\mathsf{G}'_0 = 1] - \Pr[\mathsf{G}'_1 = 1]\right| \leq \mathsf{negl}(\kappa)$.

Game G'_2: It is the same as G'_1, except that the challenger replaces the pseudo-random function $\mathsf{PRF} = (\mathsf{PRF}_{\mathsf{I}}, \mathsf{PRF}_{\mathsf{R}}, \mathsf{PRF}_{\mathsf{D}})$ with truly random function $\boxed{\mathsf{TRF} = (\mathsf{TRF}_{\mathsf{I}}, \mathsf{TRF}_{\mathsf{R}}, \mathsf{TRF}_{\mathsf{D}})}$, where $\mathsf{TRF}_{\mathsf{I}}/\mathsf{TRF}_{\mathsf{R}}/\mathsf{TRF}_{\mathsf{D}} : \{0,1\}^* \longrightarrow \mathcal{D}_{\mathsf{I},2}/\mathcal{D}_{\mathsf{R},2}/\{0,1\}^\kappa$. Especially, now the oracles $\mathcal{O}_{\mathsf{DerR}}$ and $\mathcal{O}_{\mathsf{DerI}}$ generate the real valid double keys dk_r and dk_i as follows:

- $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$ generates valid dk_r by setting $\mathsf{dk}_r := \boxed{\mathsf{TRF}_{\mathsf{D}}}(s^*_{r,i}, (m, \mathsf{amsg}_r))$, where $\mathsf{amsg}_r := (m_{r,1}, m_{r,2}, m_{r,3})$ with $m_{r,1} := f_{\mathsf{R},1}(d_{r,1})$ for $d_{r,1} \leftarrow_\$ \mathcal{D}_{\mathsf{R},1}$.
- $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ generates valid dk_i by setting $\mathsf{dk}_i := \boxed{\mathsf{TRF}_{\mathsf{D}}}(s^*_{i,r}, (\mathsf{amsg}_i, m))$, where $\mathsf{amsg}_i := (m_{i,1}, m_{i,2}, m_{i,3})$ with $m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$ for $d_{i,1} \leftarrow_\$ \mathcal{D}_{\mathsf{I},1}$.

Similar to the game transition from G_1 to G_2 in the proof of Lemma 2, G'_1 and G'_2 are computationally indistinguishable since $\mathsf{PRF} = (\mathsf{PRF}_{\mathsf{I}}, \mathsf{PRF}_{\mathsf{R}}, \mathsf{PRF}_{\mathsf{D}})$ is a pseudo-random function, and we have $\left|\Pr[\mathsf{G}'_1 = 1] - \Pr[\mathsf{G}'_2 = 1]\right| \leq \mathsf{negl}(\kappa)$.

Game G_3': It is the same as G_2', except that now the oracles $\mathcal{O}_{\mathsf{DerR}}$ and $\mathcal{O}_{\mathsf{DerI}}$ generate the real valid double keys dk_r and dk_i as follows:

- $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$ generates valid dk_r by picking $\boxed{\mathsf{dk}_r \leftarrow_\$ \{0,1\}^\kappa}$ uniformly.
- $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ generates valid dk_i by picking $\boxed{\mathsf{dk}_i \leftarrow_\$ \{0,1\}^\kappa}$ uniformly.

Clearly, in G_3', the real valid double keys dk_r (resp., dk_i) are uniformly sampled, so the challenge bit b is perfectly hidden to $\mathcal{A}$, and we have $\Pr[\mathsf{G}_3' = 1] = 1/2$.

It remains to show that G_2' and G_3' are computationally indistinguishable for $\mathcal{A}$. We define three events in G_2' as follows:

- Let E_{DR} denote the event that all invocations of $\mathsf{dk}_r := \boxed{\mathsf{TRF_D}}\,(s^*_{r,i}, (m, \mathsf{amsg}_r))$ in $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$ are on different inputs (m, amsg_r), where m is provided by $\mathcal{A}$ and $\mathsf{amsg}_r := (m_{r,1}, m_{r,2}, m_{r,3})$ has $m_{r,1} := f_{\mathsf{R},1}(d_{r,1})$ for $d_{r,1} \leftarrow_\$ \mathcal{D}_{\mathsf{R},1}$.
- Let E_{DI} denote the event that all invocations of $\mathsf{dk}_i := \boxed{\mathsf{TRF_D}}\,(s^*_{i,r}, (\mathsf{amsg}_i, m))$ in $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ are on different inputs (amsg_i, m), where m is provided by $\mathcal{A}$ and $\mathsf{amsg}_i := (m_{i,1}, m_{i,2}, m_{i,3})$ has $m_{i,1} := f_{\mathsf{I},1}(d_{i,1})$ for $d_{i,1} \leftarrow_\$ \mathcal{D}_{\mathsf{R},1}$.
- Let E_{DM} denote the event that there is no pair of invocations of $\mathsf{dk}_r := \boxed{\mathsf{TRF_D}}\,(s^*_{r,i}, (m, \mathsf{amsg}_r))$ in $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$ and $\mathsf{dk}_i := \boxed{\mathsf{TRF_D}}\,(s^*_{i,r}, (\mathsf{amsg}_i, m'))$ in $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}', m')$ on same key and same input, i.e., $(s^*_{r,i}, (m, \mathsf{amsg}_r)) = (s^*_{i,r}, (\mathsf{amsg}_i, m'))$.

If the above three events E_{DR}, E_{DI} and E_{DM} occur simultaneously, then in G_2', $\boxed{\mathsf{TRF_D}}$ is always computed on different inputs, so its outputs $\mathsf{dk}_r, \mathsf{dk}_i$ are uniformly and independently distributed, the same as those in G_3'. Besides, if E_{DM} does not happen, i.e., there exists a pair of invocations of $\mathsf{dk}_r := \boxed{\mathsf{TRF_D}}\,(s^*_{r,i}, (m, \mathsf{amsg}_r))$ and $\mathsf{dk}_i := \boxed{\mathsf{TRF_D}}\,(s^*_{i,r}, (\mathsf{amsg}_i, m'))$ with same key and same input $(s^*_{r,i}, (m, \mathsf{amsg}_r)) = (s^*_{i,r}, (\mathsf{amsg}_i, m'))$, then it implies that this pair of dk_r and dk_i are double keys of matching sessions, and thus $\mathcal{A}$ cannot test both of them (in order to avoid trivial attacks, cf. Def. 4), and consequently, the behaviour of G_2' and G_3' are also the same in this case. Overall, G_2' is identical to G_3' unless E_{DR} or E_{DI} does not happen, and we have

$$\big|\Pr[\mathsf{G}_2' = 1] - \Pr[\mathsf{G}_3' = 1]\big| \leq \Pr[\neg E_{\mathsf{DR}} \vee \neg E_{\mathsf{DI}}] \leq \Pr[\neg E_{\mathsf{DR}}] + \Pr[\neg E_{\mathsf{DI}}].$$

On the other hand, similar to the analysis in the proof of Lemma 2, we have $\Pr[\neg E_{\mathsf{DR}}] \leq \mathsf{negl}(\kappa)$ and $\Pr[\neg E_{\mathsf{DI}}] \leq \mathsf{negl}(\kappa)$ since those $m_{r,1}$ in amsg_r (resp., those $m_{i,1}$ in amsg_i) can hardly collide according to the entropy-preserving property of $f_{\mathsf{R},1}$ (resp., $f_{\mathsf{I},1}$). This shows that $\big|\Pr[\mathsf{G}_2' = 1] - \Pr[\mathsf{G}_3' = 1]\big| \leq \mathsf{negl}(\kappa)$.

Finally, by taking all things together, Lemma 3 follows. $\square$

D Missing Proofs in Sect. 5 (Instantiations of AM-AKE)

D.1 Proof of Lemma 5 (Any IND-CPA Secure KEM has Entropy-Preserving $\overline{\mathsf{Gen}_{\mathsf{KEM}}}$ and $\overline{\mathsf{Encap}}$)

Lemma 5 *If* $\mathsf{KEM} = (\mathsf{Gen}_{\mathsf{KEM}}, \mathsf{Encap}, \mathsf{Decap})$ *is a* IND-CPA *secure KEM scheme, then the function* $\overline{\mathsf{Gen}_{\mathsf{KEM}}}(\cdot)$ *that outputs only* pk *and the function* $\overline{\mathsf{Encap}}(\mathsf{pk};\cdot)$ *that outputs only* ψ *are entropy-preserving.*

Proof of Lemma 5. We will construct two PPT algorithms $\mathcal{A}$ and $\mathcal{B}$, such that for any public key pk and any ciphertext ψ^*, it holds that

$$\Pr[\overline{\mathsf{Gen}_{\mathsf{KEM}}}(d_{\mathsf{G}}) = \mathsf{pk} \mid d_{\mathsf{G}} \leftarrow_{\$} \mathcal{D}_{\mathsf{G}}] \leq 4 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM},\mathcal{A}}(\kappa), \tag{4}$$

$$\Pr[\overline{\mathsf{Encap}}(\mathsf{pk}; d_{\mathsf{K}}) = \psi^* \mid d_{\mathsf{K}} \leftarrow_{\$} \mathcal{D}_{\mathsf{K}}] \leq 4 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM},\mathcal{B}}(\kappa), \tag{5}$$

both of which are negligible under the IND-CPA security of KEM, and consequently, the entropy-preserving of $\overline{\mathsf{Gen}_{\mathsf{KEM}}}(\cdot)$ and $\overline{\mathsf{Encap}}(\mathsf{pk};\cdot)$ follows.

We first describe the construction of $\mathcal{A}$. Given a challenge $(\mathsf{pk}, \psi^*, K_b^*)$, where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$, $(K_0^*, \psi^*) \leftarrow \mathsf{Encap}(\mathsf{pk})$, $K_1^* \leftarrow_{\$} \mathcal{K}$ and $b \leftarrow_{\$} \{0,1\}$, $\mathcal{A}$ aims to guess the value of b. To this end, $\mathcal{A}$ invokes $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}(d_{\mathsf{G}})$ by itself with randomness $d_{\mathsf{G}} \leftarrow_{\$} \mathcal{D}_{\mathsf{G}}$ to generate another key-pair $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}})$, and checks whether $\widetilde{\mathsf{pk}} = \mathsf{pk}$. If the check passes, $\mathcal{A}$ uses $\widetilde{\mathsf{sk}}$ to decrypt ψ^*, i.e., $\widetilde{K} \leftarrow \mathsf{Decap}(\widetilde{\mathsf{sk}}, \psi^*)$, and returns $b' = 0$ to its own challenger if and only if $\widetilde{K} = K_b^*$. Otherwise, $\mathcal{A}$ returns a uniformly chosen $b' \leftarrow_{\$} \{0,1\}$ to its own challenger.

Let PKCol denote the event that $\widetilde{\mathsf{pk}} = \mathsf{pk}$ holds, where pk is the public key in $\mathcal{A}$'s input and $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}(d_{\mathsf{G}})$ with $d_{\mathsf{G}} \leftarrow_{\$} \mathcal{D}_{\mathsf{G}}$ is generated by $\mathcal{A}$. Since $\overline{\mathsf{Gen}_{\mathsf{KEM}}}$ is the function that only outputs $\widetilde{\mathsf{pk}}$, we have that

$$\begin{aligned}\Pr[\mathsf{PKCol}] &= \Pr[\widetilde{\mathsf{pk}} = \mathsf{pk} \mid d_{\mathsf{G}} \leftarrow_{\$} \mathcal{D}_{\mathsf{G}}, (\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}(d_{\mathsf{G}})] \\ &= \Pr[\overline{\mathsf{Gen}_{\mathsf{KEM}}}(d_{\mathsf{G}}) = \mathsf{pk} \mid d_{\mathsf{G}} \leftarrow_{\$} \mathcal{D}_{\mathsf{G}}].\end{aligned} \tag{6}$$

On the other hand, when $\widetilde{\mathsf{pk}} = \mathsf{pk}$ holds, i.e., PKCol occurs, $\widetilde{\mathsf{sk}}$ has the same functionality with sk when decrypting ψ^*, and in this case, for $\widetilde{K} \leftarrow \mathsf{Decap}(\widetilde{\mathsf{sk}}, \psi^*)$, $\widetilde{K} = K_0^*$ holds with probability 1 by the correctness of KEM, while $\widetilde{K} = K_1^*$ holds with probability $1/|\mathcal{K}|$ since $K_1^* \leftarrow_{\$} \mathcal{K}$. Consequently, we have

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM},\mathcal{A}}(\kappa) = |\Pr[b' = b] - \tfrac{1}{2}| \\ &= |\Pr[\mathsf{PKCol}] \cdot \Pr[b' = b \mid \mathsf{PKCol}] + \Pr[\neg\mathsf{PKCol}] \cdot \Pr[b' = b \mid \neg\mathsf{PKCol}] - \tfrac{1}{2}| \\ &= |\Pr[\mathsf{PKCol}] \cdot \Pr[b' = b \mid \mathsf{PKCol}] + (1 - \Pr[\mathsf{PKCol}]) \cdot \tfrac{1}{2} - \tfrac{1}{2}| \\ &= \Pr[\mathsf{PKCol}] \cdot \big|\Pr[b' = b \mid \mathsf{PKCol}] - \tfrac{1}{2}\big| \\ &= \Pr[\mathsf{PKCol}] \cdot \big|\Pr[b = 0] \cdot \Pr[b' = 0 \mid b = 0 \wedge \mathsf{PKCol}] \\ &\qquad + \Pr[b = 1] \cdot \Pr[b' = 1 \mid b = 1 \wedge \mathsf{PKCol}] - \tfrac{1}{2}\big| \\ &= \Pr[\mathsf{PKCol}] \cdot \big|\tfrac{1}{2} \cdot 1 + \tfrac{1}{2} \cdot (1 - 1/|\mathcal{K}|) - \tfrac{1}{2}\big| \\ &= \tfrac{1}{2} \cdot \Pr[\mathsf{PKCol}] \cdot (1 - 1/|\mathcal{K}|) \quad \geq \quad \tfrac{1}{4} \cdot \Pr[\mathsf{PKCol}],\end{aligned} \tag{7}$$

where the last inequality holds for $|\mathcal{K}| \geq 2$. Then (4) follows from (6) and (7).

Next we describe the construction of $\mathcal{B}$, which is similar to that of $\mathcal{A}$. Given a challenge $(\mathsf{pk}, \psi^*, K_b^*)$, where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$, $(K_0^*, \psi^*) \leftarrow \mathsf{Encap}(\mathsf{pk})$, $K_1^* \leftarrow_{\$} \mathcal{K}$ and $b \leftarrow_{\$} \{0,1\}$, $\mathcal{B}$ also aims to guess the value of b. To this end, $\mathcal{B}$ invokes $(\widetilde{K}, \widetilde{\psi}) \leftarrow \mathsf{Encap}(\mathsf{pk}; d_{\mathsf{K}})$ by itself with randomness $d_{\mathsf{K}} \leftarrow_{\$} \mathcal{D}_{\mathsf{K}}$, and checks whether $\widetilde{\psi} = \psi^*$. If the check passes, then $\mathcal{B}$ returns $b' = 0$ to its own challenger if and only if $\widetilde{K} = K_b^*$. Otherwise, $\mathcal{B}$ returns a uniformly chosen $b' \leftarrow_{\$} \{0,1\}$ to its own challenger.

Let CTCol denote the event that $\widetilde{\psi} = \psi^*$ holds, where ψ^* is the ciphertext in $\mathcal{B}$'s input and $(\widetilde{K}, \widetilde{\psi}) \leftarrow \mathsf{Encap}(\mathsf{pk}; d_{\mathsf{K}})$ with $d_{\mathsf{K}} \leftarrow_{\$} \mathcal{D}_{\mathsf{K}}$ is generated by $\mathcal{B}$. Since $\overline{\mathsf{Encap}}$ is the function that only outputs $\widetilde{\psi}$, we have that

$$\begin{aligned}\Pr[\mathsf{CTCol}] &= \Pr[\widetilde{\psi} = \psi^* \mid d_{\mathsf{K}} \leftarrow_{\$} \mathcal{D}_{\mathsf{K}}, (\widetilde{K}, \widetilde{\psi}) \leftarrow \mathsf{Encap}(\mathsf{pk}; d_{\mathsf{K}})] \\ &= \Pr[\overline{\mathsf{Encap}}(\mathsf{pk}; d_{\mathsf{K}}) = \psi^* \mid d_{\mathsf{G}} \leftarrow_{\$} \mathcal{D}_{\mathsf{G}}]. \end{aligned} \tag{8}$$

On the other hand, when $\widetilde{\psi} = \psi^*$ holds, i.e., CTCol occurs, both $\widetilde{K}$ and K_0^* are the symmetric keys encapsulated in $\widetilde{\psi} = \psi^*$, and in this case, $\widetilde{K} = K_0^*$ holds with probability 1 by the correctness of KEM, while $K = K_1^*$ holds with probability $1/|\mathcal{K}|$ since $K_1^* \leftarrow_{\$} \mathcal{K}$. With a similar analysis to (7), we can get that

$$\mathsf{Adv}_{\mathsf{KEM},\mathcal{B}}^{\mathsf{IND\text{-}CPA}}(\kappa) \quad \geq \quad \tfrac{1}{4} \cdot \Pr[\mathsf{CTCol}]. \tag{9}$$

Then (5) follows from (8) and (9).

Overall, (4) and (5) hold, and consequently, the entropy-preserving of $\overline{\mathsf{Gen}_{\mathsf{KEM}}}(\cdot)$ and $\overline{\mathsf{Encap}}(\mathsf{pk}; \cdot)$ follow from the $\mathsf{IND\text{-}CPA}$ security of KEM. □

D.2 Proof of Theorem 5 (Security of $\mathsf{SIG}_{\mathsf{DDH}}$)

Theorem 5 *If the DDH assumption holds over $\mathbb{G}_1$ and H is a random oracle, then the proposed $\mathsf{SIG}_{\mathsf{DDH}}$ achieves $\mathsf{EUF\text{-}CMA}$ security.*

Proof of Theorem 5. The proof is very similar to the security proof of the Schnorr signature scheme [24] (see, e.g., [14, Subsect. 12.5], for a proof of the Schnorr scheme). Here we provide a proof for completeness. More precisely, our proof goes with two steps. We will first describe an identification protocol (denoted by $\mathsf{IP}_{\mathsf{DDH}}$) derived from $\mathsf{SIG}_{\mathsf{DDH}}$, and prove its security based on the DDH assumption holds over $\mathbb{G}_1$. Then we will prove the $\mathsf{EUF\text{-}CMA}$ security of $\mathsf{SIG}_{\mathsf{DDH}}$ based on the security of $\mathsf{IP}_{\mathsf{DDH}}$ in the random oracle model.

The Identification Protocol $\mathsf{IP}_{\mathsf{DDH}}$ and Its Security Definition. The protocol is played between two parties, say Alice and Bob. Alice generates her own key-pair via $(\mathsf{pk} = e(g_1, g_2)^x, \mathsf{sk} = g_2^x) \leftarrow \mathsf{Gen}_{\mathsf{DDH}}$ and publishes pk. Through the protocol, Alice aims to prove to Bob that she owns the sk corresponding to pk. To this end, Alice and Bob process in three steps:

1. Alice chooses a randomness $r \leftarrow_{\$} \mathbb{Z}_p$, and sends $\sigma_1 := g_1^r$ to Bob.

2. After receiving σ_1, Bob sends a uniform $d \leftarrow_{\$} \mathbb{Z}_p$ to Alice as a challenge.
3. After getting d, Alice computes $\sigma_2 := g_2^{x \cdot d + r}$ with her $\mathsf{sk} = g_2^x$ and the randomness r chosen in step 1, and sends σ_2 to Bob.

Finally, Bob checks whether $e(g_1, \sigma_2) = e(g_1, g_2)^{x \cdot d} \cdot e(\sigma_1, g_2)$ holds, and outputs 1 if and only if the check passes.

The security of the protocol $\mathsf{IP}_{\mathsf{DDH}}$ asks the hardness to impersonate Alice, even if an adversary gets pk and many transcripts of the protocol. More precisely, it requires that for any stateful PPT adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathsf{IP}_{\mathsf{DDH}},\mathcal{A}}(\kappa) :=$

$$\Pr\left[\begin{array}{c} e(g_1, \sigma_2^*) = \\ e(g_1, g_2)^{x \cdot d^*} \cdot e(\sigma_1^*, g_2) \end{array} \middle| \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{DDH}},\ \sigma_1^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{IP}}}(\mathsf{pk}) \\ d^* \leftarrow_{\$} \mathbb{Z}_p,\ \sigma_2^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{IP}}}(d^*) \end{array}\right] \leq \mathsf{negl}(\kappa), \quad (10)$$

where the oracle $\mathcal{O}_{\mathsf{IP}}$ gets no input and returns a freshly generated transcript (σ_1, d, σ_2) of the protocol to $\mathcal{A}$, i.e., $r \leftarrow_{\$} \mathbb{Z}_p$, $\sigma_1 := g_1^r$, $d \leftarrow_{\$} \mathbb{Z}_p$, $\sigma_2 := g_2^{x \cdot d + r}$.

Next we will prove the following two claims, and then Theorem 5 directly holds.

Claim 3. If the DDH assumption holds over $\mathbb{G}_1$, then the identification protocol $\mathsf{IP}_{\mathsf{DDH}}$ is secure.

Claim 4. If the identification protocol $\mathsf{IP}_{\mathsf{DDH}}$ is secure, and H is a random oracle, then the signature scheme $\mathsf{SIG}_{\mathsf{DDH}}$ achieves $\mathsf{EUF\text{-}CMA}$ security.

Proof of Claim 3. For any adversary $\mathcal{A}$ against the security of the identification protocol $\mathsf{IP}_{\mathsf{DDH}}$, we construct an algorithm $\mathcal{B}$ against the DDH assumption over $\mathbb{G}_1$ as follows.

Given a DDH challenge $(\mathsf{pp}, g_1^x, g_1^y, T)$, $\mathcal{B}$ wants to compute distinguish $T = g_1^{xy}$ from $T \leftarrow_{\$} \mathbb{G}_1$, where $x, y \leftarrow_{\$} \mathbb{Z}_p$. To this end, $\mathcal{B}$ simulates the security experiment as described in (10) for $\mathcal{A}$ as follows as follows. $\mathcal{B}$ will sample a randomness $r_{\mathcal{A}}$ for $\mathcal{A}$, and invoke $\mathcal{A}$ twice with the same randomness $r_{\mathcal{A}}$ as follows.

– $\mathcal{B}$ computes $\mathsf{pk} := e(g_1^x, g_2) = e(g_1, g_2)^x$, sends pk to $\mathcal{A}$. Then $\mathcal{B}$ answers the $\mathcal{O}_{\mathsf{IP}}$ queries for $\mathcal{A}$ by sampling $w \leftarrow_{\$} \mathbb{Z}_p$, $d \leftarrow_{\$} \mathbb{Z}_p$, computing $\sigma_2 := g_2^w$, $\sigma_1 := g_1^w \cdot (g_1^x)^{-d} = g_1^{w - x \cdot d}$, and returning (σ_1, d, σ_2). In particular, for each $\mathcal{O}_{\mathsf{IP}}$ query made by $\mathcal{A}$, $\mathcal{B}$ will use the same randomness to answer the query for the two invocations.
– At some point $\mathcal{A}$ outputs σ_1^*.
– $\mathcal{B}$ picks $d^* \leftarrow_{\$} \mathbb{Z}_p$ and sends d^* to $\mathcal{A}$ in the first invocation of $\mathcal{A}$, while $\mathcal{B}$ picks another $d'^* \leftarrow_{\$} \mathbb{Z}_p$ and sends d'^* to $\mathcal{A}$ in the second invocation of $\mathcal{A}$.
– $\mathcal{B}$ continues to answer the $\mathcal{O}_{\mathsf{IP}}$ queries for $\mathcal{A}$, the same as the above.
– At the end of the first invocation, $\mathcal{A}$ outputs σ_2^*, while at the end of the second invocation, $\mathcal{A}$ outputs $\sigma_2'^*$.

Finally, if $d^* \neq d'^*$, then $\mathcal{B}$ computes $h := (\sigma_2^* / \sigma_2'^*)^{(d^* - d'^*)^{-1}}$, uses h to check whether $e(g_1^y, h) = e(T, g_2)$ holds, and outputs 1 if and only if the check passes; otherwise, $\mathcal{B}$ outputs 0.

Below we analyze the simulation by $\mathcal{B}$. Clearly, $\mathcal{B}$'s simulation of pk is perfect, and $\mathcal{B}$'s answers for $\mathcal{O}_{\mathsf{IP}}$ queries are also perfect, since the real transcripts ($\sigma_1 := g_1^r, d \leftarrow_\$ \mathbb{Z}_p, \sigma_2 := g_2^{x\cdot d+r}$) and the simulated transcripts ($\sigma_1 := g_1^{w-x\cdot d}, d \leftarrow_\$ \mathbb{Z}_p, \sigma_2 := g_2^w$) are identically distributed for $r, w \leftarrow_\$ \mathbb{Z}_p$.

For each $\mathcal{O}_{\mathsf{IP}}$ query made by $\mathcal{A}$, $\mathcal{B}$ will use the same randomness to answer the query for the two invocations, so that $\mathcal{A}$'s views in these two invocations are the same. Consequently, $\mathcal{A}$ will output the same σ_1^* in the two invocations.

Suppose that $d^* \neq d'^*$ and $\mathcal{A}$ succeeds in both of the two invocations, i.e., both $e(g_1, \sigma_2^*) = e(g_1, g_2)^{x\cdot d^*} \cdot e(\sigma_1^*, g_2)$ and $e(g_1, \sigma_2'^*) = e(g_1, g_2)^{x\cdot d'^*} \cdot e(\sigma_1^*, g_2)$ hold. Then by dividing these two equations, we get that $e(g_1, \sigma_2^*/\sigma_2'^*) = e(g_1, g_2)^{x\cdot(d^*-d'^*)}$, which implies that $\sigma_2^*/\sigma_2'^* = g_2^{x\cdot(d^*-d'^*)}$. Consequently, the h computed by $\mathcal{B}$ is in fact $h := (\sigma_2^*/\sigma_2'^*)^{(d^*-d'^*)^{-1}} = g_2^x$, and it is clear to see that the check of $e(g_1^y, h) = e(T, g_2)$ passes if and only if $T = g_1^{xy}$. Overall, $\mathcal{B}$ is able to distinguish $T = g_1^{xy}$ from $T \leftarrow_\$ \mathbb{G}_1$, as long as $d^* \neq d'^*$ and $\mathcal{A}$ succeeds in both of the two invocations. More precisely, let r_{exp} denote all messages $\mathcal{A}$ received in the experiment except d^* and d'^*. As we explained above, r_{exp} is the same for the two invocations since $\mathcal{B}$ uses the same randomness. Then we have

$$
\begin{aligned}
&\mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1,\mathcal{B}}(\kappa) \geq \Pr[d^* \neq d'^* \ \wedge\ \mathcal{A}(r_{\mathsf{exp}}, d^*; r_{\mathcal{A}}) \text{ succeeds} \ \wedge\ \mathcal{A}(r_{\mathsf{exp}}, d'^*; r_{\mathcal{A}}) \text{ succeeds}] \\
&\geq \Pr[\mathcal{A}(r_{\mathsf{exp}}, d^*; r_{\mathcal{A}}) \text{ succeeds} \ \wedge\ \mathcal{A}(r_{\mathsf{exp}}, d'^*; r_{\mathcal{A}}) \text{ succeeds}] - \Pr[d^* = d'^*] \\
&= \sum_{r_0}\sum_{r_1} \Pr[r_{\mathsf{exp}} = r_0] \cdot \Pr[r_{\mathcal{A}} = r_1] \cdot \Pr[\mathcal{A}(r_0, d^*; r_1) \text{ succeeds} \ \wedge\ \mathcal{A}(r_0, d'^*; r_1) \text{ succeeds}] - \tfrac{1}{p} \\
&= \sum_{r_0}\sum_{r_1} \Pr[r_{\mathsf{exp}} = r_0] \cdot \Pr[r_{\mathcal{A}} = r_1] \cdot \Pr[\mathcal{A}(r_0, d^*; r_1) \text{ succeeds}] \cdot \Pr[\mathcal{A}(r_0, d'^*; r_1) \text{ succeeds}] - \tfrac{1}{p} \\
&= \sum_{r_0}\sum_{r_1} \Pr[r_{\mathsf{exp}} = r_0] \cdot \Pr[r_{\mathcal{A}} = r_1] \cdot \Pr[\mathcal{A}(r_0, d^*; r_1) \text{ succeeds}]^2 - \tfrac{1}{p} \\
&= \mathbb{E}_{r_{\mathsf{exp}}}\mathbb{E}_{r_{\mathcal{A}}} \Pr[\mathcal{A}(r_{\mathsf{exp}}, d^*; r_{\mathcal{A}}) \text{ succeeds}]^2 - \tfrac{1}{p} \\
&\geq \big(\mathbb{E}_{r_{\mathsf{exp}}}\mathbb{E}_{r_{\mathcal{A}}} \Pr[\mathcal{A}(r_{\mathsf{exp}}, d^*; r_{\mathcal{A}}) \text{ succeeds}]\big)^2 - \tfrac{1}{p} \qquad (11) \\
&= \big(\sum_{r_0}\sum_{r_1} \Pr[r_{\mathsf{exp}} = r_0] \cdot \Pr[r_{\mathcal{A}} = r_1] \cdot \Pr[\mathcal{A}(r_0, d^*; r_1) \text{ succeeds}]\big)^2 - \tfrac{1}{p} \\
&= \Pr[\mathcal{A}(r_{\mathsf{exp}}, d^*; r_{\mathcal{A}}) \text{ succeeds}]^2 - \tfrac{1}{p} = \mathsf{Adv}_{\mathsf{IP}_{\mathsf{DDH}},\mathcal{A}}(\kappa)^2 - \tfrac{1}{p},
\end{aligned}
$$

where the summations $\sum_{r_0}$ and $\sum_{r_1}$ are over all possible values of r_{exp} and $r_{\mathcal{A}}$, respectively, $\mathbb{E}$ denotes the mathematical expectation, (11) follows from the fact that $\mathbb{E}\ X^2 \geq (\mathbb{E}\ X^2)$ holds for any variable X. Putting differently, it holds that

$$\mathsf{Adv}_{\mathsf{IP}_{\mathsf{DDH}},\mathcal{A}}(\kappa) \leq \sqrt{\mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1,\mathcal{B}}(\kappa) + \tfrac{1}{p}},$$

which is negligible under the DDH assumption over $\mathbb{G}_1$. This shows the security of the identification protocol $\mathsf{IP}_{\mathsf{DDH}}$. ▌

Proof of Claim 4. For any adversary $\mathcal{A}$ against the $\mathsf{EUF\text{-}CMA}$ security of the signature scheme $\mathsf{SIG}_{\mathsf{DDH}}$, we construct an algorithm $\mathcal{B}$ against the security of the identification protocol $\mathsf{IP}_{\mathsf{DDH}}$ as follows. The reduction is in the random oracle model.

$\mathcal{B}$ is constructed by simulating the $\mathsf{EUF\text{-}CMA}$ security experiment as described in (3) for $\mathcal{A}$. Let Q denote the number of $\mathcal{O}_{\mathsf{Sign}}$ queries made by $\mathcal{A}$.

– Firstly, $\mathcal{B}$ receives $\mathsf{pk} = e(g_1, g_2)^x$ from its own challenger, and passes pk to $\mathcal{A}$. Moreover, $\mathcal{B}$ samples an index $j^* \leftarrow_\$ [Q]$ uniformly.
– Then $\mathcal{B}$ needs to answer the $\mathcal{O}_{\mathsf{Sign}}(m)$ queries for $\mathcal{A}$. To this end, $\mathcal{B}$ asks its own $\mathcal{O}_{\mathsf{IP}}$ oracle to obtain a fresh transcript $(\sigma_1 = g_1^r, d, \sigma_2 = g_2^{x\cdot d+r})$ of the protocol $\mathsf{IP}_{\mathsf{DDH}}$, where $r, d \leftarrow_\$ \mathbb{Z}_p$, then sets the hash value $\mathsf{H}(m, \sigma_1) := d$, and returns $\sigma := (\sigma_1, \sigma_2)$ as a signature of m to $\mathcal{A}$. It is clear to see that the simulation of $\sigma = (\sigma_1, \sigma_2)$ is perfect.
– Meanwhile, $\mathcal{B}$ needs to answer the random oracle queries $\mathsf{H}(m', \sigma_1')$ for $\mathcal{A}$.
  - If this is the j^*-th random oracle query made by $\mathcal{A}$, denoted by $(m'^{(j^*)}, \sigma_1'^{(j^*)})$, then $\mathcal{B}$ returns $\sigma_1'^{(j^*)}$ to its own challenger and receives $d^* \leftarrow_\$ \mathbb{Z}_p$ from its own challenger (cf. (10) for the security experiment of $\mathsf{IP}_{\mathsf{DDH}}$). Then $\mathcal{B}$ sets the hash value $\mathsf{H}(m'^{(j^*)}, \sigma_1'^{(j^*)}) := d^*$, and returns d^* to $\mathcal{A}$.
  - If $\mathsf{H}(m', \sigma_1')$ is already defined, $\mathcal{B}$ returns the value of $\mathsf{H}(m', \sigma_1')$ to $\mathcal{A}$.
  - Otherwise, $\mathcal{B}$ samples $d' \leftarrow_\$ \mathbb{Z}_p$ uniformly, sets the hash value $\mathsf{H}(m', \sigma_1') := d'$, and returns d' to $\mathcal{A}$.
– Finally, $\mathcal{B}$ receives a forgery $(m^*, \sigma^* = (\sigma_1^*, \sigma_2^*))$ from $\mathcal{A}$, and outputs σ_2^* to its own challenger.

Clearly, $\mathcal{B}$'s simulation of pk is perfect, and $\mathcal{B}$'s answers for $\mathcal{O}_{\mathsf{Sign}}$ and H queries are perfect as well, since $\mathcal{B}$ always sets the hash values as uniformly random elements.

We note that $\mathcal{B}$ breaks the security of the identification protocol $\mathsf{IP}_{\mathsf{DDH}}$ if the following three events occur simultaneously:

– Event I: $\mathcal{A}$ made a random oracle query $\mathsf{H}(m^*, \sigma_1^*)$ to $\mathcal{B}$.
– Event II: $\mathcal{A}$'s random oracle query $\mathsf{H}(m^*, \sigma_1^*)$ happened to be the j^*-th query.
– Event III: $\mathcal{A}$ breaks the $\mathsf{EUF\text{-}CMA}$ security of $\mathsf{SIG}_{\mathsf{DDH}}$ by providing a fresh and valid forgery $(m^*, \sigma^* = (\sigma_1^*, \sigma_2^*))$, i.e., satisfying

$$e(g_1, \sigma_2^*) = e(g_1, g_2)^{x\cdot d'^*} \cdot e(\sigma_1^*, g_2) \tag{12}$$

  for $d'^* := \mathsf{H}(m^*, \sigma_1^*) \in \mathbb{Z}_p$.

This is because that when $\mathcal{A}$'s random oracle query $\mathsf{H}(m^*, \sigma_1^*)$ is the j^*-th query, i.e., $(m'^{(j^*)}, \sigma_1'^{(j^*)}) = (m^*, \sigma_1^*)$, then $\mathcal{B}$ actually returns $\sigma_1'^{(j^*)} = \sigma_1^*$ to its own challenger and sets the hash value of $\mathsf{H}(m^*, \sigma_1^*)$ as the obtained d^*, i.e., $d'^* = \mathsf{H}(m^*, \sigma_1^*) = d^*$, and thus $\mathcal{B}$'s final output σ_2^* breaks the security of $\mathsf{IP}_{\mathsf{DDH}}$ as long as $\mathcal{A}$'s forgery satisfies (12). Consequently, we get that

$$\begin{aligned}
&\mathsf{Adv}_{\mathsf{IP}_{\mathsf{DDH}},\mathcal{B}}(\kappa) \geq \Pr[\text{Event I} \wedge \text{Event II} \wedge \text{Event III}] \\
&= \Pr[\text{Event I} \wedge \text{Event III}] \cdot \Pr[\text{Event II} \mid \text{Event I} \wedge \text{Event III}] \\
&= (\Pr[\text{Event III}] - \Pr[\neg\text{Event I} \wedge \text{Event III}]) \cdot \Pr[\text{Event II} \mid \text{Event I} \wedge \text{Event III}] \\
&\geq (\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}_{\mathsf{DDH}},\mathcal{A}}(\kappa) - \tfrac{1}{p}) \cdot \tfrac{1}{Q},
\end{aligned} \tag{13}$$

where (13) follows from the three facts that $\Pr[\text{Event III}] = \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}_{\mathsf{DDH}},\mathcal{A}}(\kappa)$, $\Pr[\neg\text{Event I} \wedge \text{Event III}] \leq \frac{1}{p}$ (if $\mathcal{A}$ never queries $\mathsf{H}(m^*, \sigma_1^*)$, then the value of $d'^* := \mathsf{H}(m^*, \sigma_1^*) \in \mathbb{Z}_p$ is uniformly random to $\mathcal{A}$, and thus $\mathcal{A}$'s forgery can satisfy (12) with probability at most $\frac{1}{p}$), and $\Pr[\text{Event II} \mid \text{Event I} \wedge \text{Event III}] \geq \frac{1}{Q}$ (if $\mathcal{A}$ made a random oracle query $\mathsf{H}(m^*, \sigma_1^*)$, then for $j^* \leftarrow_\$ [Q]$, the query happened to be the j^*-th query with probability at least $\frac{1}{Q}$).

Putting differently, it holds that

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}_{\mathsf{DDH}},\mathcal{A}}(\kappa) \leq Q \cdot \mathsf{Adv}_{\mathsf{IP}_{\mathsf{DDH}},\mathcal{B}}(\kappa) + \tfrac{1}{p},$$

which is negligible assuming the security of the identification protocol $\mathsf{IP}_{\mathsf{DDH}}$. This shows the $\mathsf{EUF\text{-}CMA}$ security of the signature scheme $\mathsf{SIG}_{\mathsf{DDH}}$. ▌

Finally, by combining Claim 3 and Claim 4 together, Theorem 5 follows. □

D.3 Proof of Lemma 8 (Qualified $\mathsf{AKE}_{\mathsf{3K}}$ via The Three-KEM Paradigm)

Lemma 8 *If* KEM *and* KEM_0 *meet the requirements listed in Table 2, then the* $\mathsf{AKE}_{\mathsf{3K}}$ *yielded by the three-KEM paradigm is a qualified AKE for constructing AM-AKE.*

Proof of Lemma 8. To prove that $\mathsf{AKE}_{\mathsf{3K}} = (\mathsf{Gen}_{\mathsf{3K}}, \mathsf{Init}_{\mathsf{3K}}, \mathsf{DerR}_{\mathsf{3K}}, \mathsf{DerI}_{\mathsf{3K}})$ is a qualified one, we show that all requirements listed in Table 2 are satisfied, i.e., $\mathsf{Gen}_{\mathsf{3K}}$ has secret extractability, $\mathsf{Init}_{\mathsf{3K}}$ is 3-separable with entropy-preserving functions $(f_{\mathsf{I},1}, f_{\mathsf{I},2})$, and $\mathsf{DerR}_{\mathsf{3K}}$ is 3-separable with entropy-preserving functions $(f_{\mathsf{R},1}, f_{\mathsf{R},2})$.

- Since $\mathsf{Gen}_{\mathsf{3K}} = \mathsf{Gen}_{\mathsf{KEM}}$, the secret extractability of $\mathsf{Gen}_{\mathsf{3K}}$ follows directly from that of $\mathsf{Gen}_{\mathsf{KEM}}$.
- The process of $\mathsf{Init}_{\mathsf{3K}}(\mathsf{pk}_r, \mathsf{sk}_i)$ for generating $(\mathsf{msg}_i = (\widetilde{\mathsf{pk}}, \psi_i), \mathsf{st} = (\widetilde{\mathsf{sk}}, K_i))$ can be decomposed into three steps:
  1. $d_{\mathsf{G}} \leftarrow_\$ \mathcal{D}_{\mathsf{G}}$ and $\widetilde{\mathsf{pk}} := \overline{\mathsf{Gen}}_{\mathsf{KEM}_0}(d_{\mathsf{G}})$. So we can define $f_{\mathsf{I},1} := \overline{\mathsf{Gen}}_{\mathsf{KEM}_0}$, and then the entropy-preserving of $f_{\mathsf{I},1}$ follows from that of $\overline{\mathsf{Gen}}_{\mathsf{KEM}_0}$.
  2. $d_{\mathsf{K},i} \leftarrow_\$ \mathcal{D}_{\mathsf{K}}$ and $\psi_i := \overline{\mathsf{Encap}}(\mathsf{pk}_r; d_{\mathsf{K},i})$. So we can define $f_{\mathsf{I},2} := \overline{\mathsf{Encap}}(\mathsf{pk}_r; \cdot)$, and then the entropy-preserving of $f_{\mathsf{I},2}$ follows from that of $\overline{\mathsf{Encap}}(\mathsf{pk}_r; \cdot)$.
  3. $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) := \mathsf{Gen}_{\mathsf{KEM}_0}(d_{\mathsf{G}})$, $(K_i, \psi_i) := \mathsf{Encap}(\mathsf{pk}_r; d_{\mathsf{K},i})$, and set $\mathsf{st} := (\widetilde{\mathsf{sk}}, K_i)$. This process can be defined as $(\varepsilon, \mathsf{st}) \leftarrow \overline{\mathsf{Init}}_{\mathsf{3K}}(\mathsf{pk}_r, \mathsf{sk}_i, d_{\mathsf{G}}, d_{\mathsf{K},i})$.

  Consequently, $\mathsf{Init}_{\mathsf{3K}}$ is 3-separable with two entropy-preserving functions $(f_{\mathsf{I},1} = \overline{\mathsf{Gen}}_{\mathsf{KEM}_0}, f_{\mathsf{I},2} = \overline{\mathsf{Encap}}(\mathsf{pk}_r; \cdot))$ and an algorithm $\overline{\mathsf{Init}}_{\mathsf{3K}}$.
- Similarly, the process of $\mathsf{DerR}_{\mathsf{3K}}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i = (\widetilde{\mathsf{pk}}, \psi_i))$ for generating $(\mathsf{msg}_r = (\widetilde{\psi}, \psi_r), \mathsf{K}_r)$ can be decomposed into three steps:
  1. $d_{\mathsf{K}_0} \leftarrow_\$ \mathcal{D}_{\mathsf{K}_0}$ and $\widetilde{\psi} := \overline{\mathsf{Encap}}_0(\widetilde{\mathsf{pk}}; d_{\mathsf{K}_0})$. So we can define $f_{\mathsf{R},1} := \overline{\mathsf{Encap}}_0(\widetilde{\mathsf{pk}}; \cdot)$, and then the entropy-preserving of $f_{\mathsf{R},1}$ follows from that of $\overline{\mathsf{Encap}}_0(\widetilde{\mathsf{pk}}; \cdot)$.
  2. $d_{\mathsf{K},r} \leftarrow_\$ \mathcal{D}_{\mathsf{K}}$ and $\psi_r := \overline{\mathsf{Encap}}(\mathsf{pk}_i; d_{\mathsf{K},r})$. So we can define $f_{\mathsf{R},2} := \overline{\mathsf{Encap}}(\mathsf{pk}_i; \cdot)$, and then the entropy-preserving of $f_{\mathsf{R},2}$ follows from that of $\overline{\mathsf{Encap}}(\mathsf{pk}_i; \cdot)$.

3. $K_i \leftarrow \mathsf{Decap}(\mathsf{sk}_r, \psi_i)$, $(\widetilde{K}, \widetilde{\psi}) := \mathsf{Encap}_0(\widetilde{\mathsf{pk}}; d_{\mathsf{K}_0})$, $(K_r, \psi_r) := \mathsf{Encap}(\mathsf{pk}_i; d_{\mathsf{K},r})$, and sets $\mathsf{K}_r := \mathsf{H}(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r, K_i, K_r, \widetilde{K})$. This process can be defined as $(\varepsilon, \mathsf{K}_r) \leftarrow \overline{\mathsf{DerR}}_{\mathsf{KS}}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i = (\widetilde{\mathsf{pk}}, \psi_i), d_{\mathsf{K}_0}, d_{\mathsf{K},r})$, with ε denoting the empty string.

Consequently, $\mathsf{DerR}_{\mathsf{3K}}$ is 3-separable with two entropy-preserving functions $(f_{\mathsf{R},1} = \overline{\mathsf{Encap}}_0(\widetilde{\mathsf{pk}}; \cdot), f_{\mathsf{R},2} = \overline{\mathsf{Encap}}(\mathsf{pk}_i; \cdot))$ and an algorithm $\overline{\mathsf{DerR}}_{\mathsf{3K}}$. $\square$

D.4 Proof of Theorem 6 (Security of $\mathsf{KEM}_{\mathsf{DDH}}$)

Theorem 6 *If the DDH assumption holds over $\mathbb{G}_1$, then the proposed* $\mathsf{KEM}_{\mathsf{DDH}}$ *achieves* IND-CPA *security.*

Proof of Theorem 6. The proof is quite straightforward. For any adversary $\mathcal{A}$ against the IND-CPA security of $\mathsf{KEM}_{\mathsf{DDH}}$, we construct an algorithm $\mathcal{B}$ against the DDH assumption over $\mathbb{G}_1$ as follows.

Given a DDH challenge $(\mathsf{pp}, g_1^x, g_1^r, T)$, $\mathcal{B}$ wants to distinguish $T = g_1^{xr}$ from $T \leftarrow_\$ \mathbb{G}_1$, where $x, r \leftarrow_\$ \mathbb{Z}_p$. To this end, $\mathcal{B}$ computes $\mathsf{pk} := e(g_1^x, g_2) = e(g_1, g_2)^x$, $\psi^* := g_1^r$, $K^* := e(T, g_2)$, gives $(\mathsf{pk}, \psi^*, K^*)$ to $\mathcal{A}$, and returns the output of $\mathcal{A}$ to its own challenger. It is easy to see that $\mathcal{B}$'s simulation of (pk, ψ^*) is perfect. If $T = g_1^{xr}$, then $K^* = e(T, g_2) = e(g_1, g_2)^{xr}$, which is the real symmetric key encapsulated in $\psi^* = g_1^r$; if $T \leftarrow_\$ \mathbb{G}_1$, then $K^* = e(T, g_2)$ is uniformly distributed over $\mathbb{G}_T$. Consequently, $\mathcal{B}$ is able to distinguish $T = g_1^{xr}$ from $T \leftarrow_\$ \mathbb{G}_1$, as long as $\mathcal{A}$ can distinguish the real symmetric key $K^* = e(g_1, g_2)^{xr}$ encapsulated in ψ^* from a uniformly chosen $K^* \leftarrow_\$ \mathbb{G}_T$, and we have $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM}_{\mathsf{DDH}},\mathcal{A}}(\kappa) \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1,\mathcal{B}}(\kappa)$, which is negligible under the DDH assumption over $\mathbb{G}_1$. This shows the IND-CPA security of $\mathsf{KEM}_{\mathsf{DDH}}$. $\square$

E Generic Construction of Plain AM-AKE with Relaxed Security from AKE

As shown in Subsect. 3.4, it is impossible for a plain (two-pass) AM-AKE scheme to achieve responder-robustness, achieve IND-WM/sIND-WM if it is initiator-robust, and achieve PR-DK/sPR-DK. The best security for it is the relaxed security notions defined in Def. 6 in Subsect. 3.4.

In this section, we propose a generic construction of plain AM-AKE with relaxed security from a basic AKE, with the help of a KEM and a PRF. The resulting plain AM-AKE achieves initiator-robust, relaxed sIND-WM security and relaxed sPR-DK security, simultaneously. To make the construction possible, we require the underlying AKE and KEM to meet some new requirements, which are defined in Appendix E.1. Then we will show the generic construction in Appendix E.2, and prove its security in Appendix E.3. Finally, in Appendix E.4, we discuss how to achieve responder-robustness for plain AM-AKE by relying on more passes.

E.1 Requirements for The Underlying AKE and KEM

To construct a plain AM-AKE scheme from AKE and KEM, we require the underlying AKE have partially randomness-recoverable algorithms Init and DerR, and the underlying KEM have pseudo-random public keys, encapsulated keys and ciphertexts, which are defined as follows.

Definition 16 (AKE with Partially Randomness-Recoverable Init and DerR). *Let* $\mathsf{AKE} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI})$ *be a two-pass AKE scheme, with* $\mathcal{D}_\mathsf{I}^* \times \overline{\mathcal{D}}_\mathsf{I}$ *the randomness space of* Init *and* $\mathcal{D}_\mathsf{R}^* \times \overline{\mathcal{D}}_\mathsf{R}$ *the randomness space of* DerR. *We say that the algorithms* Init *and* DerR *are partially randomness-recoverable, if there exist PPT algorithms* $\mathsf{Rec}_\mathsf{Init}$ *and* $\mathsf{Rec}_\mathsf{DerR}$*, such that for any* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}$*,* $(\mathsf{pk}_r, \mathsf{sk}_r) \leftarrow \mathsf{Gen}$*,* $d_i^* \in \mathcal{D}_\mathsf{I}^*$*,* $\overline{d}_i \in \overline{\mathcal{D}}_\mathsf{I}$*,* $d_r^* \in \mathcal{D}_\mathsf{R}^*$*,* $\overline{d}_r \in \overline{\mathcal{D}}_\mathsf{R}$*,* $(\mathsf{msg}_i, \mathsf{st}) \leftarrow \mathsf{Init}(\mathsf{pk}_r, \mathsf{sk}_i; (d_i^*, \overline{d}_i))$*,* $(\mathsf{msg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i; (d_r^*, \overline{d}_r))$*, it holds that*

$$d_i^* = \mathsf{Rec}_\mathsf{Init}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i) \quad \textit{and} \quad d_r^* = \mathsf{Rec}_\mathsf{DerR}(\mathsf{pk}_r, \mathsf{sk}_i, \mathsf{msg}_r, \mathsf{st}).$$

Definition 17 (Fully Pseudo-Random KEM). *Let* $\mathsf{KEM} = (\mathsf{Gen}_\mathsf{KEM}, \mathsf{Encap}, \mathsf{Decap})$ *be a KEM scheme with public key space* $\mathcal{PK}$*, encapsulated key space* $\mathcal{K}$ *and ciphertext space* $\mathcal{CT}$*. We say that* KEM *is fully pseudo-random, if for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\Big| \Pr[\mathcal{A}(\mathsf{pk}, K, \psi) = 1] - \Pr[\mathcal{A}(\mathsf{pk}', K', \psi') = 1] \Big| \leq \mathsf{negl}(\kappa)$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_\mathsf{KEM}$*,* $(K, \psi) \leftarrow \mathsf{Encap}(\mathsf{pk})$*,* $\mathsf{pk}' \leftarrow_\$ \mathcal{PK}$*,* $K' \leftarrow_\$ \mathcal{K}$ *and* $\psi' \leftarrow_\$ \mathcal{CT}$*.*

Based on the properties defined above, we are ready to present the generic construction of plain AM-AKE.

E.2 Construction of Plain AM-AKE from AKE, KEM and PRF

Let $\mathsf{KEM} = (\mathsf{Gen}_\mathsf{KEM}, \mathsf{Encap}, \mathsf{Decap})$ be a fully pseudo-random KEM as per Def. 17, with public key space $\mathcal{PK}$, encapsulated key space $\mathcal{K}$ and ciphertext space $\mathcal{CT}$. Let $\mathsf{AKE} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI})$ be a two-pass AKE scheme with partially randomness-recoverable $(\mathsf{Init}, \mathsf{DerR})$ supported by $(\mathsf{Rec}_\mathsf{Init}, \mathsf{Rec}_\mathsf{DerR})$ as per Def. 16, where the randomness space of Init is $\mathcal{D}_\mathsf{I}^* \times \overline{\mathcal{D}}_\mathsf{I}$ with $\mathcal{D}_\mathsf{I}^* = \mathcal{PK}$, and the randomness space of DerR is $\mathcal{D}_\mathsf{R}^* \times \overline{\mathcal{D}}_\mathsf{R}$ with $\mathcal{D}_\mathsf{R}^* = \mathcal{CT} \times \{0,1\}^\kappa$. Moreover, let $\mathsf{PRF} : \mathcal{K} \times \{0,1\}^* \longrightarrow \{0,1\}^{2\kappa}$. For ease of exposition, we parse the output of PRF as two parts, i.e., $\mathsf{PRF}_\mathsf{R}/\mathsf{PRF}_\mathsf{D} : \mathcal{K} \times \{0,1\}^* \longrightarrow \{0,1\}^\kappa$, such that $\mathsf{PRF}(K, m) = (\mathsf{PRF}_\mathsf{R}(K, m), \mathsf{PRF}_\mathsf{D}(K, m))$ for all $K \in \mathcal{K}$ and $m \in \{0,1\}^*$.

Now we convert AKE to a plain AM-AKE scheme $\mathsf{AM\text{-}AKE} = ((\mathsf{Gen}, \mathsf{Init}, \mathsf{DerR}, \mathsf{DerI}), (\mathsf{aGen}, \mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}))$ with the help of KEM and PRF, where the anamorphic algorithms are described below. (See also Fig. 8 for an illustration of the resulting plain $\mathsf{AM\text{-}AKE}$.)

- $\underline{(\mathsf{apk}, \mathsf{ask}, \mathsf{aux}) \leftarrow \mathsf{aGen}}$: it invokes AKE's key generation algorithm $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$, and sets $(\mathsf{apk}, \mathsf{ask}, \mathsf{aux}) := (\mathsf{pk}, \mathsf{sk}, \perp)$.

- $\underline{(\mathsf{amsg}_i, \mathsf{st}, \mathsf{aux}'_i) \leftarrow \mathsf{aInit}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}_i = \bot)}$: it invokes $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$, sets $d_i^* := \widetilde{\mathsf{pk}}$, chooses $\overline{d}_i \leftarrow_\$ \overline{\mathcal{D}}_\mathsf{I}$, and invokes $(\mathsf{msg}_i, \mathsf{st}) \leftarrow \mathsf{Init}(\mathsf{apk}_r, \mathsf{ask}_i; (d_i^*, \overline{d}_i))$. Then, it returns $(\mathsf{amsg}_i := \mathsf{msg}_i, \mathsf{st}, \mathsf{aux}'_i := (\widetilde{\mathsf{sk}}, \mathsf{amsg}_i))$.
- $\underline{(\mathsf{amsg}_r, \mathsf{K}_r, \mathsf{dk}_r) \leftarrow \mathsf{aDerR}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{aux}_r = \bot, \mathsf{amsg}_i)}$: it first recovers the partial randomness used by Init via computing $\widetilde{\mathsf{pk}} := \mathsf{Rec}_{\mathsf{Init}}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{amsg}_i)$. Then it invokes $(\widetilde{K}, \widetilde{\psi}) \leftarrow \mathsf{Encap}(\widetilde{\mathsf{pk}})$ and computes $h := \mathsf{PRF}_\mathsf{R}(\widetilde{K}, \mathsf{amsg}_i)$. Next it sets $d_r^* := (\widetilde{\psi}, h)$, chooses $\overline{d}_r \leftarrow_\$ \overline{\mathcal{D}}_\mathsf{R}$, invokes $(\mathsf{msg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{amsg}_i; (d_r^*, \overline{d}_r))$, and sets $\mathsf{amsg}_r := \mathsf{msg}_r$. Finally, it computes $\mathsf{dk}_r := \mathsf{PRF}_\mathsf{D}(\widetilde{K}, (\mathsf{amsg}_i, \mathsf{amsg}_r))$ as the double key, and returns $(\mathsf{amsg}_r, \mathsf{K}_r, \mathsf{dk}_r)$.
- $\underline{(\mathsf{K}_i, \mathsf{dk}_i) \leftarrow \mathsf{aDerI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}'_i = (\widetilde{\mathsf{sk}}, \mathsf{amsg}_i), \mathsf{amsg}_r, \mathsf{st})}$: it first recovers the partial randomness used by DerR via computing $d_r^* := \mathsf{Rec}_{\mathsf{DerR}}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{amsg}_r, \mathsf{st})$, and parses $d_r^* = (\widetilde{\psi}, h)$. Then it decrypts $\widetilde{\psi}$ to obtain $\widetilde{K} \leftarrow \mathsf{Decap}(\widetilde{\mathsf{sk}}, \widetilde{\psi})$, and checks whether $h = \mathsf{PRF}_\mathsf{R}(\widetilde{K}, \mathsf{amsg}_i)$ holds. If the check passes, it sets $\mathsf{dk}_i := \mathsf{PRF}_\mathsf{D}(\widetilde{K}, (\mathsf{amsg}_i, \mathsf{amsg}_r))$ as the double key; otherwise, $\mathsf{dk}_i := \bot$. Finally, it invokes $\mathsf{K}_i \leftarrow \mathsf{DerI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{amsg}_r, \mathsf{st})$, and returns $(\mathsf{K}_i, \mathsf{dk}_i)$.

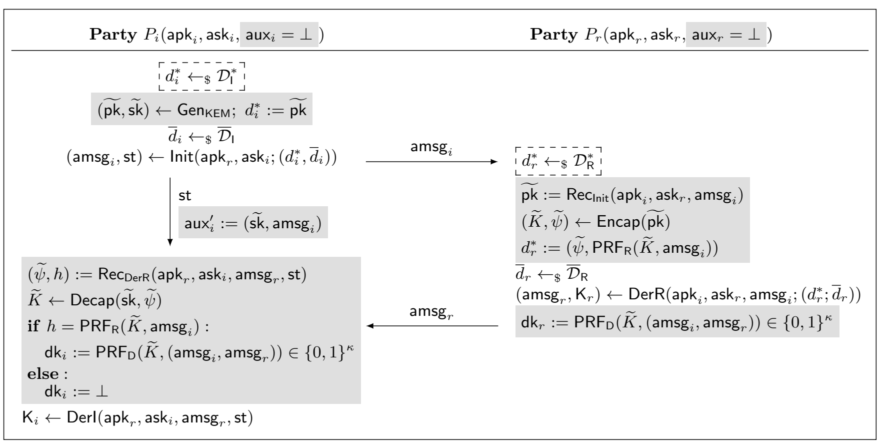

Fig. 8. Generic construction of the *plain* AM-AKE scheme AM-AKE based on AKE, KEM and PRF, where dotted boxes appear only in normal algorithms (Gen, Init, DerR, DerI), and gray boxes appear only in anamorphic algorithms (aGen, aInit, aDerR, aDerI).

Let us compare the normal algorithms and the anamorphic ones.

- The anamorphic algorithm aGen is identical to the normal algorithm Gen, so are the key-pairs $(\mathsf{apk}, \mathsf{ask})$ and $(\mathsf{pk}, \mathsf{sk})$ they generate.
- The normal algorithm Init makes use of random coins d_i^* and $\overline{d}_i$ for the generation of msg_i and st. The anamorphic algorithm aInit can be regarded as the normal Init taking specific coins $d_i^* = \widetilde{\mathsf{pk}}$ and random coins $\overline{d}_i$, with $\widetilde{\mathsf{pk}}$ an ephemeral public key of KEM.

– The normal algorithm DerR makes use of random coins d_r^* and $\overline{d}_r$ for the generation of msg_r and K_r. The anamorphic algorithm aDerR has two parts: one part can be regarded as the normal DerR taking specific coins $d_r^* = (\widetilde{\psi}, h)$ and random coins $\overline{d}_r$ to output msg_r and K_r, where $(\widetilde{K}, \widetilde{\psi}) \leftarrow \mathsf{Encap}(\widetilde{\mathsf{pk}})$ and $h = \mathsf{PRF_R}(\widetilde{K}, \mathsf{amsg}_i)$; the other part is in charge of generating the double key $\mathsf{dk}_r := \mathsf{PRF_D}(\widetilde{K}, (\mathsf{amsg}_i, \mathsf{amsg}_r))$.
– The normal algorithm DerI is deterministic and outputs K_i. The anamorphic algorithm aDerI functions identically as DerI for the generation of K_i, but it is also in charge of generating the double key $\mathsf{dk}_i := \mathsf{PRF_D}(\widetilde{K}, (\mathsf{amsg}_i, \mathsf{amsg}_r))$ or $\mathsf{dk}_i := \perp$ depending on whether $h = \mathsf{PRF_R}(\widetilde{K}, \mathsf{amsg}_i)$.

Below we analyze the correctness and robustness of our plain $\mathsf{AM\text{-}AKE}$.

Correctness. It is easy to see that the correctness of plain $\mathsf{AM\text{-}AKE}$ follows from the correctness of AKE, the partially randomness-recoverable property of $(\mathsf{Init}, \mathsf{DerR})$ and the correctness of KEM. In particular, the correctness of AKE guarantees $\mathsf{K}_i = \mathsf{K}_r$ for every possible choices of d_i^*, $\overline{d}_i$, d_r^*, $\overline{d}_r$, so even using specific coins in the anamorphic algorithms, we also have $\mathsf{K}_i = \mathsf{K}_r$. Moreover, by the correctness of KEM, we have $\mathsf{dk}_i = \mathsf{PRF_D}(\widetilde{K}, (\mathsf{amsg}_i, \mathsf{amsg}_r)) = \mathsf{dk}_r$.

Initiator-Robustness. Suppose that P_i invokes anamorphic algorithms aInit and aDerI while P_r invokes normal algorithm DerR, then the randomness $d_r^* \leftarrow_\$ \mathcal{D}_\mathsf{R}^*$ used in DerR is uniformly chosen. When P_i invokes the anamorphic algorithm aDerI to recover the partial randomness d_r^* and parse $d_r^* = (\widetilde{\psi}, h)$, we know that $\widetilde{\psi}$ and h are independently and uniformly distributed. Therefore for $\widetilde{K} \leftarrow \mathsf{Decap}(\widetilde{\mathsf{sk}}, \widetilde{\psi})$, the check $h = \mathsf{PRF_R}(\widetilde{K}, \mathsf{amsg}_i)$ can pass with only a negligible probability $1/2^\kappa$ due to the uniformity of h, and consequently, P_i will set $\mathsf{dk}_i := \perp$ with overwhelming probability.

E.3 Security Proofs

We show the relaxed security of the plain $\mathsf{AM\text{-}AKE}$ proposed in Appendix E.2.

Theorem 7 (Relaxed Security of Plain $\mathsf{AM\text{-}AKE}$). *Let* AKE *be a two-pass AKE scheme with partially randomness-recoverable algorithms* $(\mathsf{Init}, \mathsf{DerR})$*, let* KEM *be a fully pseudo-random KEM, and let* PRF *be a pseudo-random function. Then the plain AM-AKE scheme* $\mathsf{AM\text{-}AKE}$ *constructed in Appendix E.2 achieves* relaxed $\mathsf{sIND\text{-}WM}$ *and* relaxed $\mathsf{sPR\text{-}DK}$ *security.*

The proof of Theorem 7 consists of two parts: the relaxed $\mathsf{sIND\text{-}WM}$ security follows from Lemma 9 and Lemma 10, while the relaxed $\mathsf{sPR\text{-}DK}$ security follows from Lemma 11.

Lemma 9. *For any adversary* $\mathcal{A}$*, it holds that* $\left|\Pr\left[\mathcal{A}(\mathsf{pk}, \mathsf{sk}) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{apk}, \mathsf{ask}) = 1\right]\right| = 0$*, where* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$ *and* $(\mathsf{apk}, \mathsf{ask}, \mathsf{aux}) \leftarrow \mathsf{aGen}$.

Proof of Lemma 9. Since both the anamorphic key-pair $(\mathsf{apk}, \mathsf{ask})$ and the normal key-pair $(\mathsf{pk}, \mathsf{sk})$ are generated by Gen, they have the same distribution. □

Lemma 10. *There exists PPT simulator* $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$*, such that for any PPT adversary* $\mathcal{A}$ *and* $N = \mathsf{poly}(\kappa)$*,* $\left|\Pr\left[\mathsf{Exp}^{\text{relaxed-}\mathsf{sIND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N} = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\kappa)$.

Proof of Lemma 10. We first describe the simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$.

- $\underline{R_i \leftarrow \mathsf{SimI}(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}_i = \perp, R'_i)}$: Here R'_i is an internal randomness used in aInit, and thus includes the randomness used in $\mathsf{Gen}_{\mathsf{KEM}}$ which is denoted by d_{G}, as well as $\overline{d}_i$ used in Init, i.e., $R'_i = (d_{\mathsf{G}}, \overline{d}_i)$. This algorithm aims to explain R'_i as a randomness R_i for Init. To this end, it computes $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) := \mathsf{Gen}_{\mathsf{KEM}}(d_{\mathsf{G}})$, sets $d_i^* := \widetilde{\mathsf{pk}}$, and outputs $R_i := (d_i^*, \overline{d}_i)$.
- $\underline{R_r \leftarrow \mathsf{SimR}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{aux}_r = \perp, m, R'_r)}$: Here R'_r is an internal randomness used in aDerR, and thus includes the randomness used in Encap denoted by d_{K}, as well as $\overline{d}_r$ used in $\overline{\mathsf{DerR}}$, i.e., $R'_r = (d_{\mathsf{K}}, \overline{d}_r)$. This algorithm aims to explain R'_r as a randomness R_r for DerR. To this end, it computes $(\widetilde{K}, \widetilde{\psi}) := \mathsf{Encap}(\widetilde{\mathsf{pk}}; d_{\mathsf{K}})$, sets $d_r^* := (\widetilde{\psi}, \mathsf{PRF}_{\mathsf{R}}(\widetilde{K}, m))$, and outputs $R_r := (d_r^*, \overline{d}_r)$.

We prove the lemma via a sequence of games G_0-G_3, where the differences between adjacent games are highlighted in gray boxes.

$\underline{\textbf{Game } \mathsf{G}_0\textbf{:}}$ This is the $\mathsf{Exp}^{\text{relaxed-}\mathsf{sIND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}$ experiment (cf. Fig. 2). Then we have $\Pr\left[\mathsf{Exp}^{\text{relaxed-}\mathsf{sIND\text{-}WM}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N} = 1\right] = \Pr[\mathsf{G}_0 = 1]$.

In this game, the challenger samples a challenge bit $b \leftarrow_{\$} \{0, 1\}$, and answers the $\mathcal{O}_{\mathsf{Init}}, \mathcal{O}_{\mathsf{DerR}}, \mathcal{O}_{\mathsf{DerI}}$ queries for $\mathcal{A}$ in the following way:

– If $b = 0$, the challenger invokes the normal algorithms $\mathsf{Init}, \mathsf{DerR}, \mathsf{DerI}$;
– If $b = 1$ and $\mathcal{A}$ designates normal mode (i.e., $\mathbf{N}$), the challenger also invokes the normal algorithms;
– If $b = 1$ and $\mathcal{A}$ designates anamorphic mode (i.e., $\mathbf{A}$), the challenger invokes the anamorphic algorithm $\mathsf{aInit}/\mathsf{aDerR}/\mathsf{aDerI}$ and the simulator $\mathsf{SimI}/\mathsf{SimR}$.

The adversary $\mathcal{A}$ succeeds if it guesses b correctly. Overall, there are differences between $b = 0$ and $b = 1$ only if $\mathcal{A}$ designates anamorphic mode (i.e., $\mathbf{A}$).

We note that the oracles $\mathcal{O}_{\mathsf{Init}}, \mathcal{O}_{\mathsf{DerR}}, \mathcal{O}_{\mathsf{DerI}}$ output $(\mathsf{msg}_i, \mathsf{st}, R_i)$, $(\mathsf{msg}_r, \mathsf{K}_r, R_r)$ and K_i, respectively, but do not output the double keys $\mathsf{dk}_i, \mathsf{dk}_r$. The differences between the normal algorithms and the anamorphic algorithms and simulator in generating these values only lie in the distributions of d_i^* and d_r^*:

– The normal algorithms $\mathsf{Init}, \mathsf{DerR}, \mathsf{DerI}$ use uniformly chosen coins $d_i^* \leftarrow_{\$} \mathcal{D}_{\mathsf{I}}^*$ and $d_r^* \leftarrow_{\$} \mathcal{D}_{\mathsf{R}}^*$.
– The anamorphic algorithms $\mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}$ and simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$ involve specific coins $d_i^* := \widetilde{\mathsf{pk}} \in \mathcal{D}_{\mathsf{I}}^*$ and $d_r^* := (\widetilde{\psi}', \mathsf{PRF}_{\mathsf{R}}(\widetilde{K}', m)) \in \mathcal{D}_{\mathsf{R}}^*$, where $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) := \mathsf{Gen}_{\mathsf{KEM}}$, $\widetilde{\mathsf{pk}}' := \mathsf{Rec}_{\mathsf{Init}}(\mathsf{apk}_i, \mathsf{ask}_r, m)$, $(\widetilde{K}, \widetilde{\psi}) := \mathsf{Encap}(\widetilde{\mathsf{pk}}')$, and m is chosen by $\mathcal{A}$ as the input of $\mathcal{O}_{\mathsf{DerR}}$.

Notice that for any session sID, $\mathcal{A}$ always queries $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$ with $m = \mathsf{amsg}_i$ to obtain valid result where amsg_i is outputted by $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID})$, otherwise

$\mathcal{O}_{\mathsf{DerR}}$ will return $\perp$. In this case, $\widetilde{\mathsf{pk}} = \widetilde{\mathsf{pk}}'$ because Init is partially randomness-recoverable.

Game G_1: It is the same as G_0, except that at the beginning of the game, the challenger samples $\boxed{\widetilde{\mathsf{pk}}_j \leftarrow_\$ \mathcal{PK} = \mathcal{D}_{\mathsf{I}}^*}$, $\boxed{\widetilde{K}_j \leftarrow_\$ \mathcal{K}}$, $\boxed{\widetilde{\psi}_j \leftarrow_\$ \mathcal{CT}}$ for every $j \in Q_{\mathsf{New}}$ where Q_{New} is the maximum number of the queries for $\mathcal{O}_{\mathsf{New}}$ by $\mathcal{A}$. Then in the case of $b = 1$ and the mode designated by $\mathcal{A}$ is anamorphic (i.e., $\mathbf{A}$), the challenger answers the oracle queries $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID}, \mathsf{w_I} = \mathbf{A}), \mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m, \mathsf{w_R} = \mathbf{A}), \mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ for $\mathcal{A}$ as follows:

- Let $j := \mathsf{sID}$. The challenger invokes anamorphic algorithms $\mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}$ and the simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$, by using specific coins $d_i^* := \widetilde{\mathsf{pk}} \in \mathcal{D}_{\mathsf{I}}^*$ and $d_r^* := (\widetilde{\psi}, \mathsf{PRF_R}(\widetilde{K}, m)) \in \mathcal{D}_{\mathsf{R}}^*$, where $\boxed{\widetilde{\mathsf{pk}} := \widetilde{\mathsf{pk}}_j}$, $\boxed{\widetilde{K} := \widetilde{K}_j}$, $\boxed{\widetilde{\psi} := \widetilde{\psi}_j}$.

By the fully pseudo-random property of KEM (cf. Def. 17), for every $j \in Q_{\mathsf{New}}$, the j-th tuple $(\widetilde{\mathsf{pk}}, \widetilde{K}, \widetilde{\psi})$ generated in G_0 with $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen_{KEM}}$, $(\widetilde{K}, \widetilde{\psi}) \leftarrow \mathsf{Encap}(\widetilde{\mathsf{pk}})$ is computationally indistinguishable from $\boxed{(\widetilde{\mathsf{pk}}_j, \widetilde{K}_j, \widetilde{\psi}_j)}$ with $\boxed{\widetilde{\mathsf{pk}} := \widetilde{\mathsf{pk}}_j}$, $\boxed{\widetilde{K} := \widetilde{K}_j}$, $\boxed{\widetilde{\psi} := \widetilde{\psi}_j}$ in G_1, which is unknown to $\mathcal{A}$. By a standard hybrid argument, we conclude that $\Big|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_0 = 1]\Big| \le \mathsf{negl}(\kappa)$.

Game G_2: It is the same as G_1, except that the challenger replaces the pseudo-random function $\mathsf{PRF} = (\mathsf{PRF_R}, \mathsf{PRF_D})$ with truly random function $\boxed{\mathsf{TRF} = (\mathsf{TRF_R}, \mathsf{TRF_D})}$, where $\mathsf{TRF_R}/\mathsf{TRF_D} : \{0,1\}^* \longrightarrow \mathcal{D}_{\mathsf{R}}^*/\{0,1\}^\kappa$. More precisely, when $b = 1$ and the mode designated by $\mathcal{A}$ is anamorphic (i.e., $\mathbf{A}$), the challenger answers the oracle queries $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m, \mathsf{w_R} = \mathbf{A}), \mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ for $\mathcal{A}$ as follows:

- Let $j := \mathsf{sID}$. The challenger invokes anamorphic algorithms $\mathsf{aDerR}, \mathsf{aDerI}$ and the simulator SimR, by using specific coins $d_r^* := (\widetilde{\psi}_j, \boxed{\mathsf{TRF_R}}(\widetilde{K}_j, m)) \in \mathcal{D}_{\mathsf{R}}^*$.

Since $\mathsf{PRF} = (\mathsf{PRF_R}, \mathsf{PRF_D})$ is a pseudo-random function, its outputs are computationally indistinguishable from the outputs of truly random function $\mathsf{TRF} = (\mathsf{TRF_R}, \mathsf{TRF_D})$. Consequently, this change is unnoticeable to $\mathcal{A}$, and by a standard hybrid argument over the PRF keys $\widetilde{K}_j$, we have $\big|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]\big| \le \mathsf{negl}(\kappa)$.

Game G_3: It is the same as G_2, except that in the case of $b = 1$ and the mode designated by $\mathcal{A}$ is anamorphic (i.e., $\mathbf{A}$), the challenger answers the oracle queries $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID}, \mathsf{w_I} = \mathbf{A}), \mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m, \mathsf{w_R} = \mathbf{A}), \mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ for $\mathcal{A}$ as follows:

- The challenger invokes anamorphic algorithms $\mathsf{aDerR}, \mathsf{aDerI}$ and the simulator SimR, by using uniformly chosen coins $\boxed{d_r^* \leftarrow_\$ \mathcal{D}_{\mathsf{R}}^*}$.

Clearly, in G_3, the anamorphic algorithms $\mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}$ and the simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$ using uniformly chosen coins $d_i^* \leftarrow_\$ \mathcal{D}_{\mathsf{I}}^*$ and $\boxed{d_r^* \leftarrow_\$ \mathcal{D}_{\mathsf{R}}^*}$, which are essentially the same as the normal algorithms $\mathsf{Init}, \mathsf{DerR}, \mathsf{DerI}$ in generating the responses $(\mathsf{msg}_i, \mathsf{st}, R_i)$, $(\mathsf{msg}_r, \mathsf{K}_r, R_r)$ and K_i, so the challenge bit b is perfectly hidden to $\mathcal{A}$, and we have $\Pr[\mathsf{G}_3 = 1] = 1/2$.

It remains to show that G_2 and G_3 are computationally indistinguishable for $\mathcal{A}$. Notice that every invocation of $\mathsf{TRF}_\mathsf{R}(\widetilde{K}_j, \mathsf{amsg}_i)$ uses a different input instance $\widetilde{K}_j$, then $d_r^* := (\widetilde{\psi}_j, \mathsf{TRF}_\mathsf{R}(\widetilde{K}_j, \mathsf{amsg}_i)) \in \mathcal{D}_\mathsf{R}^*$ is always mapped to a uniformly random value in $\mathcal{CT} \times \{0,1\}^\kappa = \mathcal{D}_\mathsf{R}^*$. This shows that $\big|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_3 = 1]\big| \le \mathsf{negl}(\kappa)$.

Finally, by taking all things together, Lemma 10 follows. $\square$

Lemma 11. *There exists PPT simulator* $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$*, such that for any PPT adversary* $\mathcal{A}$ *and* $N = \mathsf{poly}(\kappa)$*,* $\left|\Pr\left[\mathsf{Exp}^{\text{relaxed-sPR-DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N} = 1\right] - \frac{1}{2}\right| \le \mathsf{negl}(\kappa)$.

Proof of Lemma 11. We adopt the same simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$ defined in the proof of Lemma 10. We prove the lemma via a sequence of games G'_0-G'_3, which are defined similarly as those G_0-G_3 in the proof of Lemma 10.

<u>**Game G'_0:**</u> This is the $\mathsf{Exp}^{\text{relaxed-sPR-DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N}$ experiment (cf. Fig. 3). Then we have $\Pr\left[\mathsf{Exp}^{\text{relaxed-sPR-DK}}_{\mathsf{AM\text{-}AKE},\mathcal{A},\mathsf{Sim},N} = 1\right] = \Pr[\mathsf{G}'_0 = 1]$.

In this game, the challenger samples a challenge bit $b \leftarrow_\$ \{0,1\}$, and answers the $\mathcal{O}_{\mathsf{Init}}, \mathcal{O}_{\mathsf{DerR}}, \mathcal{O}_{\mathsf{DerI}}$ queries for $\mathcal{A}$ by invoking the anamorphic algorithms $\mathsf{aInit}, \mathsf{aDerR}, \mathsf{aDerI}$ and the simulator $\mathsf{Sim} = (\mathsf{SimI}, \mathsf{SimR})$. Moreover, the challenger answers the $\mathcal{O}_{\mathsf{TestDK}}$ queries for $\mathcal{A}$, by returning the real double keys dk_r (resp., dk_i) generated in $\mathcal{O}_{\mathsf{DerR}}$ (resp., $\mathcal{O}_{\mathsf{DerI}}$) if $b = 1$ while returning uniformly chosen $\mathsf{dk} \leftarrow_\$ \{0,1\}^\kappa$ if $b = 0$. Note that if the dk_r (resp., dk_i) generated in $\mathcal{O}_{\mathsf{DerR}}$ (resp., $\mathcal{O}_{\mathsf{DerI}}$) is invalid (i.e., equals $\perp$), then the challenger will output $\perp$ directly for the $\mathcal{O}_{\mathsf{TestDK}}$ query regardless of the value of b. The adversary $\mathcal{A}$ succeeds if it guesses b correctly.

We note that the oracles $\mathcal{O}_{\mathsf{DerR}}$ and $\mathcal{O}_{\mathsf{DerI}}$ generate the real *valid* double keys dk_r and dk_i according to aDerR and aDerI as follows:

- $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$ generates *valid* dk_r by setting $\mathsf{dk}_r := \mathsf{PRF}_\mathsf{D}(\widetilde{K}, (m, \mathsf{amsg}_r)) \in \{0,1\}^\kappa$ where $\widetilde{\mathsf{pk}}' := \mathsf{Rec}_{\mathsf{Init}}(\mathsf{apk}_i, \mathsf{ask}_r, m)$, $(\widetilde{K}, \widetilde{\psi}) \leftarrow \mathsf{Encap}(\widetilde{\mathsf{pk}}')$, and $\mathsf{amsg}_r := \mathsf{DerR}(\mathsf{apk}_i, \mathsf{ask}_r, m; (d_r^*, \overline{d}_r))$ with $d_r^* := (\widetilde{\psi}, \mathsf{PRF}_\mathsf{R}(\widetilde{K}, m))$ and $\overline{d}_r \leftarrow_\$ \overline{\mathcal{D}}_\mathsf{R}$.
- $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ generates *valid* dk_i by setting $\mathsf{dk}_i := \mathsf{PRF}_\mathsf{D}(\widetilde{K}', (\mathsf{amsg}_i, m)) \in \{0,1\}^\kappa$, where $(\widetilde{\psi}', h) := \mathsf{Rec}_{\mathsf{DerR}}(\mathsf{apk}_r, \mathsf{ask}_i, m, \mathsf{st} := S[\mathsf{sID}])$, $\widetilde{K}' \leftarrow \mathsf{Decap}(\widetilde{\mathsf{sk}}, \widetilde{\psi}')$ and $\mathsf{amsg}_i := \mathsf{Init}(\mathsf{apk}_r, \mathsf{ask}_i; (d_i^*, \overline{d}_i))$ with $d_i^* := \widetilde{\mathsf{pk}}$ for $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$ and $\overline{d}_i \leftarrow_\$ \overline{\mathcal{D}}_\mathsf{I}$. Notice that $\widetilde{\mathsf{sk}}$ is generated during the $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID})$ query and stored in $Aux[\mathsf{sID}] = \mathsf{aux}'_i = (\widetilde{\mathsf{sk}}, \mathsf{amsg}_i)$.

Here $(i, r) := (\mathrm{init}[\mathsf{sID}], \mathrm{resp}[\mathsf{sID}])$ denote the initiator and responder of sID.

Notice that for any session sID, $\mathcal{A}$ always queries $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$ with $m = \mathsf{amsg}_i$ to obtain valid result where amsg_i is outputted by $\mathcal{O}_{\mathsf{Init}}(\mathsf{sID})$, otherwise $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$ will return $\perp$. Similarly, $\mathcal{A}$ always queries $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m')$ with $m' = \mathsf{amsg}_r$ to obtain valid result where amsg_r is outputted by $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$, otherwise $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m')$ will return $\perp$. In this case, we have $\widetilde{\mathsf{pk}}' = \widetilde{\mathsf{pk}}$ because Init is partially randomness-recoverable, and $(\widetilde{K}', \widetilde{\psi}') = (\widetilde{K}, \widetilde{\psi})$ because DerR is partially randomness-recoverable.

Game G'_1: It is the same as G'_0, except that at the beginning of the game, the challenger samples $\boxed{\widetilde{\mathsf{pk}}_j \leftarrow_\$ \mathcal{PK}}$, $\boxed{\widetilde{K}_j \leftarrow_\$ \mathcal{K}}$, $\boxed{\widetilde{\psi}_j \leftarrow_\$ \mathcal{CT}}$ for every $j \in Q_{\mathsf{New}}$ where Q_{New} is the maximum number of the queries for $\mathcal{O}_{\mathsf{New}}$ by $\mathcal{A}$. When $\mathcal{A}$ queries the oracles $\mathcal{O}_{\mathsf{Init}}, \mathcal{O}_{\mathsf{DerR}}, \mathcal{O}_{\mathsf{DerI}}$, the challenger answers as follows:

- For $\mathcal{O}_{\mathsf{Init}}(j := \mathsf{sID})$, use $\boxed{\widetilde{\mathsf{pk}} := \widetilde{\mathsf{pk}}_j}$ instead of $\boxed{\widetilde{\mathsf{pk}} := \widetilde{\mathsf{pk}}'}$ where $(\widetilde{\mathsf{pk}}', \widetilde{\mathsf{sk}}') \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$.
- For $\mathcal{O}_{\mathsf{DerR}}(j := \mathsf{sID}, m = M^{\mathsf{out}}_{\mathsf{I}}[\mathsf{sID}])$: Set $\boxed{\widetilde{K} := \widetilde{K}_j}$ and $\boxed{\widetilde{\psi} := \widetilde{\psi}_j}$. Especially, now $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$ generates valid dk_r by $\mathsf{dk}_r := \mathsf{PRF}_{\mathsf{D}}(\widetilde{K}_j, (m, \mathsf{amsg}_r))$ where $\mathsf{amsg}_r := (\mathsf{apk}_i, \mathsf{ask}_r, m; (d^*_r, \overline{d}_r))$ with $d^*_r := (\widetilde{\psi}_j, \mathsf{PRF}_{\mathsf{R}}(\widetilde{K}_j, m))$, $\overline{d}_r \leftarrow_\$ \overline{\mathcal{D}}_{\mathsf{R}}$. Set $M^{\mathsf{out}}_{\mathsf{R}}[\mathsf{sID}] := \mathsf{amsg}_r$ and $DK[\mathsf{sID}, \mathsf{R}] := \mathsf{dk}_r$.
- For $\mathcal{O}_{\mathsf{DerI}}(j := \mathsf{sID}, m = M^{\mathsf{out}}_{\mathsf{R}}[\mathsf{sID}])$: Set $\boxed{\widetilde{K} := \widetilde{K}_j}$ and $\boxed{\widetilde{\psi} := \widetilde{\psi}_j}$. Especially, now $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ generates valid dk_i by setting $\mathsf{dk}_i := \mathsf{PRF}_{\mathsf{D}}(\widetilde{K}_j, (\mathsf{amsg}_i, m))$ where $\mathsf{amsg}_i := \mathsf{Init}(\mathsf{apk}_r, \mathsf{ask}_i; (d^*_i, \overline{d}_i))$ with $d^*_i := \widetilde{\mathsf{pk}}_j$, $\overline{d}_i \leftarrow_\$ \overline{\mathcal{D}}_{\mathsf{I}}$. Set $DK[\mathsf{sID}, \mathsf{I}] := \mathsf{dk}_i$.

Now similar to the game transition from G_0 to G_1 in the proof of Lemma 10, G'_0 and G'_1 are computationally indistinguishable by the fully pseudo-random property of KEM, and we have $\left|\Pr[\mathsf{G}'_0 = 1] - \Pr[\mathsf{G}'_1 = 1]\right| \le \mathsf{negl}(\kappa)$.

Game G'_2: It is the same as G'_1, except that the challenger replaces the pseudo-random function $\mathsf{PRF} = (\mathsf{PRF}_{\mathsf{R}}, \mathsf{PRF}_{\mathsf{D}})$ with truly random function $\boxed{\mathsf{TRF} = (\mathsf{TRF}_{\mathsf{R}}, \mathsf{TRF}_{\mathsf{D}})}$, where $\mathsf{TRF}_{\mathsf{R}}/\mathsf{TRF}_{\mathsf{D}} : \{0,1\}^* \longrightarrow \{0,1\}^\kappa$. Especially, now the oracles $\mathcal{O}_{\mathsf{DerR}}$ and $\mathcal{O}_{\mathsf{DerI}}$ work as follows:

- $\mathcal{O}_{\mathsf{DerR}}(j := \mathsf{sID}, m)$ generates valid dk_r by setting $\mathsf{dk}_r := \boxed{\mathsf{TRF}_{\mathsf{D}}}(\widetilde{K}, (m, \mathsf{amsg}_r)) = \boxed{\mathsf{TRF}_{\mathsf{D}}}(\widetilde{K}, (\mathsf{amsg}_i, \mathsf{amsg}_r))$, where $\widetilde{K} := \widetilde{K}_j$, $\mathsf{amsg}_r := (\mathsf{apk}_i, \mathsf{ask}_r, m; (d^*_r, \overline{d}_r))$ with $d^*_r := \mathsf{TRF}_{\mathsf{R}}(\widetilde{K}_j, m)$ and $\overline{d}_r \leftarrow_\$ \overline{\mathcal{D}}_{\mathsf{R}}$. Set $M^{\mathsf{out}}_{\mathsf{R}}[\mathsf{sID}] := \mathsf{amsg}_r$ and $DK[\mathsf{sID}, \mathsf{R}] := \mathsf{dk}_r$.
- $\mathcal{O}_{\mathsf{DerI}}(j := \mathsf{sID}, m)$ generates valid dk_i by setting $\mathsf{dk}_i := \boxed{\mathsf{TRF}_{\mathsf{D}}}(\widetilde{K}, (\mathsf{amsg}_i, m)) = \boxed{\mathsf{TRF}_{\mathsf{D}}}(\widetilde{K}, (\mathsf{amsg}_i, \mathsf{amsg}_r))$, where $\widetilde{K} := \widetilde{K}_j$, $\mathsf{amsg}_i := \mathsf{Init}(\mathsf{apk}_r, \mathsf{ask}_i; (d^*_i, \overline{d}_i))$ with $d^*_i := \widetilde{\mathsf{pk}}_j$ and $\overline{d}_i \leftarrow_\$ \overline{\mathcal{D}}_{\mathsf{I}}$. Set $DK[\mathsf{sID}, \mathsf{I}] := \mathsf{dk}_i$.

Similar to the game transition from G_1 to G_2 in the proof of Lemma 10, G'_1 and G'_2 are computationally indistinguishable since $\mathsf{PRF} = (\mathsf{PRF}_{\mathsf{R}}, \mathsf{PRF}_{\mathsf{D}})$ is a pseudo-random function, and we have $\left|\Pr[\mathsf{G}'_1 = 1] - \Pr[\mathsf{G}'_2 = 1]\right| \le \mathsf{negl}(\kappa)$.

Game G'_3: It is the same as G'_2, except that now the oracles $\mathcal{O}_{\mathsf{DerR}}$ and $\mathcal{O}_{\mathsf{DerI}}$ generate the real valid double keys dk_r and dk_i as follows:

- $\mathcal{O}_{\mathsf{DerR}}(\mathsf{sID}, m)$ generates valid dk_r by picking $\boxed{\mathsf{dk}_r \leftarrow_\$ \{0,1\}^\kappa}$ uniformly. Set $DK[\mathsf{sID}, \mathsf{R}] := \mathsf{dk}_r$.
- $\mathcal{O}_{\mathsf{DerI}}(\mathsf{sID}, m)$ generates valid dk_i by picking $\boxed{\mathsf{dk}_i := DK[\mathsf{sID}, \mathsf{R}]}$ uniformly.

Clearly, in G_3', for the same session, only one of dk_r and dk_i is tested, so the real valid double keys dk_r (resp., dk_i) are uniformly sampled, and then the challenge bit b is perfectly hidden to $\mathcal{A}$, and we have $\Pr[\mathsf{G}_3' = 1] = 1/2$.

It remains to show that G_2' and G_3' are computationally indistinguishable for $\mathcal{A}$. Note that for each sID, the underlying $\widetilde{K}_{\mathsf{sID}}$ is chosen uniformly and independently. Then for all sID, $DK[\mathsf{sID}, \mathsf{R}] := \mathsf{dk}_r = \boxed{\mathsf{TRF}_{\mathsf{D}}}\,(\widetilde{K}_{\mathsf{sID}}, (\mathsf{amsg}_i, \mathsf{amsg}_r))$ is uniformly distributed and independent of each other.

This shows that $\left|\Pr[\mathsf{G}_2' = 1] - \Pr[\mathsf{G}_3' = 1]\right| = 0$.

Finally, by taking all things together, Lemma 11 follows. □

E.4 On Achieving Responder-Robustness for Plain AM-AKE

As shown by the first impossibility result (i.e., Theorem 1) in Subsect. 3.4, it is impossible for two-pass plain AM-AKE to achieve responder-robustness. To evade this impossibility result and achieve responder-robustness for the plain AM-AKE constructed in Appendix E.2, we have to rely on more passes.

For example, we can conduct an additional execution of AM-AKE, where we make some changes for the aInit and aDerR algorithms (denoted by aInit^* and aDerR^* in the following description), and the second execution of AM-AKE is conducted by $(\mathsf{aInit}^*, \mathsf{aDerR}^*, \mathsf{DerI})$.

Now for the second execution, the aInit^* and aDerR^* algorithms can take $\mathsf{aux}_i := (\mathsf{dk}_i, \mathsf{amsg}_i^{(1)}, \mathsf{amsg}_r^{(1)})$ and $\mathsf{aux}_r := (\mathsf{dk}_r, \mathsf{amsg}_i^{(1)}, \mathsf{amsg}_r^{(1)})$ as auxiliary inputs, respectively, where dk_i (resp., dk_r) is the double key computed by P_i (resp., P_r) in the first execution of AM-AKE, and $\mathsf{amsg}_i^{(1)}$ (resp., $\mathsf{amsg}_r^{(1)}$) is the message sent by P_i (resp., P_r) in the first execution of AM-AKE. Moreover, we need another pseudo-random function $\mathsf{PRF}_{\mathsf{I}} : \{0,1\}^\kappa \times \{0,1\}^* \longrightarrow \mathcal{D}_{\mathsf{I}}^*$ as building block. The algorithms aInit^* and aDerR^* are described as follows.

- $\underline{(\mathsf{amsg}_i^{(2)}, \mathsf{st}) \leftarrow \mathsf{aInit}^*(\mathsf{apk}_r, \mathsf{ask}_i, \mathsf{aux}_i = (\mathsf{dk}_i, \mathsf{amsg}_i^{(1)}, \mathsf{amsg}_r^{(1)}))}$: it first computes $d_i^* := \mathsf{PRF}_{\mathsf{I}}(\mathsf{dk}_i, (\mathsf{amsg}_i^{(1)}, \mathsf{amsg}_r^{(1)}))$, then randomly picks $\overline{d}_i \leftarrow_\$ \overline{\mathcal{D}}_{\mathsf{I}}$, and invokes $(\mathsf{msg}_i, \mathsf{st}) \leftarrow \mathsf{Init}(\mathsf{apk}_r, \mathsf{ask}_i; (d_i^*, \overline{d}_i))$. Finally, it returns $(\mathsf{amsg}_i^{(2)} := \mathsf{msg}_i, \mathsf{st})$.
- $\underline{(\mathsf{amsg}_r^{(2)}, \mathsf{K}_r, \mathsf{dk}_r') \leftarrow \mathsf{aDerR}^*(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{amsg}_i^{(2)}, \mathsf{aux}_r = (\mathsf{dk}_r, \mathsf{amsg}_i^{(1)}, \mathsf{amsg}_r^{(1)}))}$: it first computes $d := \mathsf{PRF}_{\mathsf{I}}(\mathsf{dk}_r, (\mathsf{amsg}_i^{(1)}, \mathsf{amsg}_r^{(1)}))$, and recovers the partial randomness used by Init via computing $d_i^* := \mathsf{Rec}_{\mathsf{Init}}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{amsg}_i^{(2)})$. Next, it checks whether $d_i^* = d$ holds. If the check passes, it sets $\mathsf{dk}_r' := \mathsf{dk}_r$; otherwise, it sets $\mathsf{dk}_r' := \perp$. Finally, it invokes $(\mathsf{msg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}(\mathsf{apk}_i, \mathsf{ask}_r, \mathsf{amsg}_i^{(2)})$, and returns $(\mathsf{amsg}_r^{(2)} := \mathsf{msg}_r, \mathsf{K}_r, \mathsf{dk}_r')$.

See Fig. 9 for an illustration of twice executions of the plain AM-AKE.

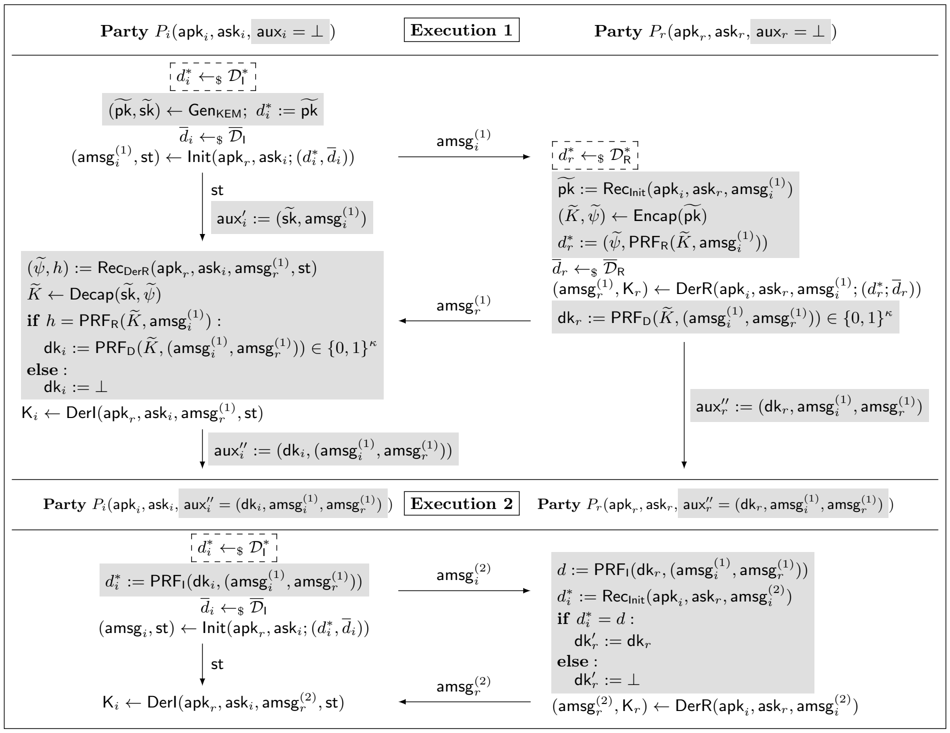

Fig. 9. Generic construction of the *plain* AM-AKE scheme AM-AKE based on AKE, KEM and PRF, where dotted boxes appear only in normal algorithms (Gen, Init, DerR, DerI), and gray boxes appear only in anamorphic algorithms (aGen, aInit, aDerR, aDerI) for the first execution and (aInit^*, aDerR^*) for the second execution.

It is easy to check the correctness for the second execution of AM-AKE.

Now we can show responder-robustness of the plain AM-AKE with twice executions. Suppose that P_i invokes normal algorithm Init in the second execution, while P_r invokes anamorphic algorithm aDerR^*. We know that P_i invokes Init with uniformly chosen randomness $(d_i^*, \overline{d}_i)$, and P_r can recover d_i^* by invoking $\mathsf{Rec}_\mathsf{Init}$. Since d_i^* is uniformly distributed over $\mathcal{D}_\mathsf{I}^*$, and in particular, it is independent of $d := \mathsf{PRF}_\mathsf{I}(\mathsf{dk}_r, (\mathsf{amsg}_i^{(1)}, \mathsf{amsg}_r^{(1)}))$, thus the check $d_i^* = d$ can pass with only a negligible probability, and consequently, P_r will set $\mathsf{dk}_r' := \perp$ with overwhelming probability.

Nevertheless, we note that the plain AM-AKE cannot achieve (non-relaxed) IND-WM/sIND-WM security or PR-DK/sPR-DK security even if it is executed twice. This is because the (initial) auxiliary message is $\mathsf{aux} = \perp$, and the adversary owning $(\mathsf{apk}_i, \mathsf{ask}_i)$ and $(\mathsf{apk}_r, \mathsf{ask}_r)$ is capable of doing whatever P_i or P_r can do. Indeed, the second and third impossibility results (i.e., Theorem 2 and Theorem 3) in Subsect. 3.4 apply to plain AM-AKE even with more passes.

F Instantiation of Plain AM-AKE with Relaxed Security

To instantiate the generic construction of plain AM-AKE proposed in Appendix E, we can employ any pseudo-random function PRF, and thus we only need to instantiate the underlying AKE and KEM, i.e., AKE with partially randomness-recoverable Init and DerR (cf. Def. 16) and fully pseudo-random KEM (cf. Def. 17).

In this section, we will show that the popular SIG+KEM paradigm [19] and three-KEM paradigm [20] for constructing AKE yield the desired AKE with partially randomness-recoverable Init and DerR, as long as the underlying SIG and/or KEM are randomness-recoverable. Moreover, the ElGamal-KEM [7] turns out to be fully pseudo-random under the DDH assumption. Then by plugging them into the generic construction in Appendix E, we immediately obtain concrete plain AM-AKE schemes with initiator-robustness and relaxed security.

More precisely, in Appendix F.1, we show that the ElGamal-KEM is fully pseudo-random. Then in Appendix F.2, we show how to instantiate AKE with partially randomness-recoverable Init and DerR via the SIG+KEM paradigm, and in Appendix F.3, we show how to instantiate it via the three-KEM paradigm.

F.1 Concrete KEM with Fully Pseudo-Random Property

In this subsection, we recall the ElGamal-KEM [7] and show that it is a fully pseudo-random KEM under the DDH assumption.

Let $\mathsf{pp} = (\mathbb{G}, p, g)$ be a description of cyclic group, where $\mathbb{G}$ is a cyclic group of prime order p and generator g. The ElGamal-KEM $\mathsf{KEM}_{\mathsf{ElG}} = (\mathsf{Gen}_{\mathsf{ElG}}, \mathsf{Encap}_{\mathsf{ElG}}, \mathsf{Decap}_{\mathsf{ElG}})$ is described as follows.

- $\underline{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{ElG}}}$: it randomly picks $x \leftarrow_{\$} \mathbb{Z}_p$, and sets $(\mathsf{pk} := g^x, \mathsf{sk} := x)$.
- $\underline{(K, \psi) \leftarrow \mathsf{Encap}_{\mathsf{ElG}}(\mathsf{pk} = g^x)}$: it randomly picks $r \leftarrow_{\$} \mathbb{Z}_p$ and outputs $(K := (g^x)^r = g^{xr}, \psi := g^r)$.
- $\underline{K \leftarrow \mathsf{Decap}_{\mathsf{ElG}}(\mathsf{sk} = x, \psi = g^r)}$: it computes $K := (g^r)^x = g^{xr}$.

It is straightforward to prove the full pseudo-randomness of $\mathsf{KEM}_{\mathsf{ElG}}$. Note that the honestly generated $(\mathsf{pk}, K, \psi) = (g^x, g^{xr}, g^r)$ with $x, r \leftarrow_{\$} \mathbb{Z}_p$ is exactly a DDH tuple, and thus it is computationally indistinguishable from a random tuple $(\mathsf{pk}', K', \psi') \leftarrow_{\$} \mathbb{G}^3$ under the DDH assumption.

F.2 Concrete AKE via The SIG+KEM Paradigm

AKE with Partially Randomness-Recoverable Init and DerR via The SIG+KEM Paradigm. We first recall the SIG+KEM paradigm of constructing two-pass AKE according to [19]. Let $\mathsf{KEM} = (\mathsf{Gen}_{\mathsf{KEM}}, \mathsf{Encap}, \mathsf{Decap})$ be a key encapsulation mechanism, $\mathsf{SIG} = (\mathsf{Gen}_{\mathsf{SIG}}, \mathsf{Sign}, \mathsf{Vrfy})$ a signature scheme and H a suitable hash function. The resulting $\mathsf{AKE}_{\mathsf{KS}} = (\mathsf{Gen}_{\mathsf{KS}}, \mathsf{Init}_{\mathsf{KS}}, \mathsf{DerR}_{\mathsf{KS}}, \mathsf{DerI}_{\mathsf{KS}})$ is described as follows (see also Fig. 10 without gray boxes).

- $\underline{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{KS}}}$: Invoke $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{SIG}}$ and return $(\mathsf{pk}, \mathsf{sk})$.

- $\underline{(\mathsf{msg}_i, \mathsf{st}) \leftarrow \mathsf{Init}_{\mathsf{KS}}(\mathsf{pk}_r, \mathsf{sk}_i)}$: Invoke $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$, $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, \widetilde{\mathsf{pk}})$, and output $\mathsf{msg}_i := (\widetilde{\mathsf{pk}}, \sigma_i)$ and the state $\mathsf{st} := (\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}})$.
- $\underline{(\mathsf{msg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}_{\mathsf{KS}}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i = (\widetilde{\mathsf{pk}}, \sigma_i))}$: If $\mathsf{Vrfy}(\mathsf{pk}_i, \widetilde{\mathsf{pk}}, \sigma_i) = 0$: output $\perp$; if $\mathsf{Vrfy}(\mathsf{pk}_i, \widetilde{\mathsf{pk}}, \sigma_i) = 1$, invoke $(K, \psi) \leftarrow \mathsf{Encap}(\widetilde{\mathsf{pk}})$, $\sigma_r \leftarrow \mathsf{Sign}(\mathsf{sk}_r, (\widetilde{\mathsf{pk}}, \psi))$, and output $\mathsf{msg}_r := (\psi, \sigma_r)$ and session key $\mathsf{K}_r := \mathsf{H}(K, \mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r)$.
- $\underline{\mathsf{K}_i \leftarrow \mathsf{DerI}_{\mathsf{KS}}(\mathsf{pk}_r, \mathsf{sk}_i, \mathsf{msg}_r = (\psi, \sigma_r), \mathsf{st} = (\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}))}$: If $\mathsf{Vrfy}(\mathsf{pk}_r, (\widetilde{\mathsf{pk}}, \psi), \sigma_r) = 0$: output $\perp$; if $\mathsf{Vrfy}(\mathsf{pk}_r, (\widetilde{\mathsf{pk}}, \psi), \sigma_r) = 1$: invoke $K \leftarrow \mathsf{Decap}(\widetilde{\mathsf{sk}}, \psi)$ and output $\mathsf{K}_i := \mathsf{H}(K, \mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r)$.

Party $P_i(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{aux}_i = \perp)$		**Party** $P_r(\mathsf{pk}_r, \mathsf{sk}_r, \mathsf{aux}_r = \perp)$
$(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$ $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}_{\mathsf{ElG}}$ $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, \widetilde{\mathsf{pk}};\ d_i^* := \mathsf{pk}^*)$	$\xrightarrow{\mathsf{msg}_i := (\widetilde{\mathsf{pk}}, \sigma_i)}$	**if** $\mathsf{Vrfy}(\mathsf{pk}_i, \widetilde{\mathsf{pk}}, \sigma_i) = 0$: **return** $\perp$ $\mathsf{pk}^* := \mathsf{Rec}_{\mathsf{SIG}}(\mathsf{pk}_i, \widetilde{\mathsf{pk}}, \sigma_i)$ $(\mathsf{K}^*, \psi^*) \leftarrow \mathsf{Encap}_{\mathsf{ElG}}(\mathsf{pk}^*)$ $d_r^* := (\psi^*, \mathsf{PRF}_{\mathsf{R}}(\mathsf{K}^*, \mathsf{msg}_i))$ $(K, \psi) \leftarrow \mathsf{Encap}(\widetilde{\mathsf{pk}})$ $\sigma_r \leftarrow \mathsf{Sign}(\mathsf{sk}_r, (\widetilde{\mathsf{pk}}, \psi);\ d_r^*)$ $\mathsf{msg}_r := (\psi, \sigma_r)$ $\mathsf{K}_r := \mathsf{H}(K, (\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r))$ $\mathsf{dk}_r := \mathsf{PRF}_{\mathsf{D}}(\mathsf{K}^*, (\mathsf{msg}_i, \mathsf{msg}_r))$
$\mathsf{st} := (\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}})$ $\mathsf{aux}_i' := (\mathsf{sk}^*, \mathsf{msg}_i)$		

if $\mathsf{Vrfy}(\mathsf{pk}_r, (\widetilde{\mathsf{pk}}, \psi), \sigma_r) = 0$:
  return $\perp$
$K \leftarrow \mathsf{Decap}(\widetilde{\mathsf{sk}}, \psi)$
$\mathsf{K}_i := \mathsf{H}(K, \mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r)$
$d_r^* := \mathsf{Rec}_{\mathsf{SIG}}(\mathsf{pk}_r, (\widetilde{\mathsf{pk}}, \psi), \sigma_r)$
parse $d_r^* = (\psi^*, h)$
$\mathsf{K}^* \leftarrow \mathsf{Decap}_{\mathsf{ElG}}(\mathsf{sk}^*, \psi^*)$
if $h = \mathsf{PRF}_{\mathsf{R}}(\mathsf{K}^*, \mathsf{msg}_i)$:
  $\mathsf{dk}_i := \mathsf{PRF}_{\mathsf{D}}(\mathsf{K}^*, (\mathsf{msg}_i, \mathsf{msg}_r))$
else :
  $\mathsf{dk}_i := \perp$
	$\xleftarrow{\mathsf{msg}_r := (\psi, \sigma_r)}$	

Fig. 10. The SIG+KEM paradigm for AKE (without gray boxes) and the resulting plain AM-AKE with initiator-robustness and relaxed security via our generic construction in Appendix E (with anamorphic algorithms in gray boxes).

Remark 1. In Fig. 10, d_i^* and d_r^* can be padded to a same length, so that they are compatible with the same space (i.e., the randomness space of the signing algorithm Sign).

Below we will show that the $\mathsf{AKE}_{\mathsf{KS}}$ has partially randomness-recoverable Init and DerR, if the underlying SIG is publicly randomness-recoverable.

Definition 18 (Publicly Randomness-Recoverable SIG). *We say that a signature scheme* $\mathsf{SIG} = (\mathsf{Gen}_{\mathsf{SIG}}, \mathsf{Sign}, \mathsf{Vrfy})$ *is publicly randomness-recoverable, if there exists a PPT algorithm* $\mathsf{Rec}_{\mathsf{SIG}}$*, such that for any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{SIG}}$*, any message* m *and any* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m; r)$*, it holds that*

$$r = \mathsf{Rec}_{\mathsf{SIG}}(\mathsf{pk}, m, \sigma).$$

With such SIG, it is easy to check that the Init and DerR algorithms of $\mathsf{AKE}_{\mathsf{KS}}$ are partially randomness-recoverable, supported by the following $\mathsf{Rec}_{\mathsf{Init}}$ and $\mathsf{Rec}_{\mathsf{DerR}}$ as per Def. 16.

- $\underline{d_i^* \leftarrow \mathsf{Rec}_{\mathsf{Init}}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i = (\widetilde{\mathsf{pk}}, \sigma_i))}$: it sets $d_i^* := \mathsf{Rec}_{\mathsf{SIG}}(\mathsf{pk}_i, \widetilde{\mathsf{pk}}, \sigma_i)$.
- $\underline{d_r^* \leftarrow \mathsf{Rec}_{\mathsf{DerR}}(\mathsf{pk}_r, \mathsf{sk}_i, \mathsf{msg}_r = (\psi, \sigma_r), \mathsf{st} = (\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}))}$: it sets $d_r^* := \mathsf{Rec}_{\mathsf{SIG}}(\mathsf{pk}_r, (\widetilde{\mathsf{pk}}, \psi), \sigma_r)$.

The correctness of $\mathsf{Rec}_{\mathsf{Init}}$ and $\mathsf{Rec}_{\mathsf{DerR}}$ directly follows from the property of publicly randomness-recoverable SIG.

Then by plugging the $\mathsf{AKE}_{\mathsf{KS}}$ and the ElGamal-KEM $\mathsf{KEM}_{\mathsf{ElG}}$ into our generic construction in Appendix E, we immediately get a plain AM-AKE scheme with initiator-robustness and relaxed security, as shown in Fig. 10 with gray boxes.

Concrete SIG. Finally, it remains to present concrete publicly randomness-recoverable SIG scheme. Here we use the Boneh-Boyen signature scheme [3], which is publicly randomness-recoverable as noted in [15].

Let $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2, g_T)$ be a description of asymmetric pairing group, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p, $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a non-degenerated bilinear pairing, and g_1, g_2, g_T are generators of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ respectively. The Boneh-Boyen signature scheme $\mathsf{SIG}_{\mathsf{BB}} = (\mathsf{Gen}_{\mathsf{BB}}, \mathsf{Sign}, \mathsf{Vrfy})$ is recalled as follows.

– $\underline{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{BB}}}$: it randomly picks $x \leftarrow_\$ \mathbb{Z}_p, y \leftarrow_\$ \mathbb{Z}_p$, sets $\mathsf{sk} := (x, y)$, $\mathsf{pk} := (g_2^x, g_2^y)$, and outputs $(\mathsf{pk}, \mathsf{sk})$.
– $\underline{\sigma \leftarrow \mathsf{Sign}(\mathsf{sk} = (x, y), m \in \mathbb{Z}_p)}$: it randomly selects $r \leftarrow_\$ \mathbb{Z}_p$. If $r = -(x + m)/y$, then it re-samples r; otherwise, it computes $s := g_1^{1/(x+yr+m)}$, and outputs $\sigma := (r, s)$.
– $\underline{0/1 \leftarrow \mathsf{Vrfy}(\mathsf{pk} = (g_2^x, g_2^y), m, \sigma = (r, s))}$: it checks whether $e(s, g_2^x \cdot g_2^m \cdot (g_2^y)^r) = e(g_1, g_2)$ holds. If the check passes, then return 1; otherwise return 0.

It is easy to check that the Boneh-Boyen scheme is publicly randomness-recoverable, because the signature σ already contains the randomness r, which is trivially recoverable.

F.3 Concrete AKE via The Three-KEM Paradigm

AKE with Partially Randomness-Recoverable Init and DerR via The Three-KEM Paradigm. We first recall the three-KEM paradigm of constructing two-pass AKE according to [20]. Let $\mathsf{KEM} = (\mathsf{Gen}_{\mathsf{KEM}}, \mathsf{Encap}, \mathsf{Decap})$ and $\mathsf{KEM}_0 = (\mathsf{Gen}_{\mathsf{KEM}_0}, \mathsf{Encap}_0, \mathsf{Decap}_0)$ be two KEM schemes, and H a suitable hash function. The resulting $\mathsf{AKE}_{\mathsf{3K}} = (\mathsf{Gen}_{\mathsf{3K}}, \mathsf{Init}_{\mathsf{3K}}, \mathsf{DerR}_{\mathsf{3K}}, \mathsf{DerI}_{\mathsf{3K}})$ is described as follows (see also Fig. 11 without gray boxes).

- $\underline{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{3K}}}$: Invoke $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$ and return $(\mathsf{pk}, \mathsf{sk})$.
- $\underline{(\mathsf{msg}_i, \mathsf{st}) \leftarrow \mathsf{Init}_{\mathsf{3K}}(\mathsf{pk}_r, \mathsf{sk}_i)}$: Invoke $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}_0}$, $(K_i, \psi_i) \leftarrow \mathsf{Encap}(\mathsf{pk}_r)$, and output $\mathsf{msg}_i := (\widetilde{\mathsf{pk}}, \psi_i)$ and the state $\mathsf{st} := (\widetilde{\mathsf{sk}}, K_i)$.

- $\underline{(\mathsf{msg}_r, \mathsf{K}_r) \leftarrow \mathsf{DerR}_{\mathsf{3K}}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i = (\widetilde{\mathsf{pk}}, \psi_i))}$: Invoke $K_i \leftarrow \mathsf{Decap}(\mathsf{sk}_r, \psi_i)$, $(\widetilde{K}, \widetilde{\psi}) \leftarrow \mathsf{Encap}_0(\widetilde{\mathsf{pk}})$ and $(K_r, \psi_r) \leftarrow \mathsf{Encap}(\mathsf{pk}_i)$. Output $\mathsf{msg}_r := (\widetilde{\psi}, \psi_r)$ and session key $\mathsf{K}_r := \mathsf{H}(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r, K_i, K_r, \widetilde{K})$.
- $\underline{\mathsf{K}_i \leftarrow \mathsf{DerI}_{\mathsf{3K}}(\mathsf{pk}_r, \mathsf{sk}_i, \mathsf{msg}_r = (\widetilde{\psi}, \psi_r), \mathsf{st} = (\widetilde{\mathsf{sk}}, K_i))}$: Invoke $\widetilde{K} \leftarrow \mathsf{Decap}_0(\widetilde{\mathsf{sk}}, \widetilde{\psi})$, $K_r \leftarrow \mathsf{Decap}(\mathsf{sk}_i, \psi_r)$, and output $\mathsf{K}_i := \mathsf{H}(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r, K_i, K_r, \widetilde{K})$.

Party $P_i(\mathsf{pk}_i, \mathsf{sk}_i, \boxed{\mathsf{aux}_i = \bot})$		**Party** $P_r(\mathsf{pk}_r, \mathsf{sk}_r, \boxed{\mathsf{aux}_r = \bot})$
$(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}_0}$ $\boxed{(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}_{\mathsf{EIG}}}$ $(K_i, \psi_i) \leftarrow \mathsf{Encap}(\mathsf{pk}_r; \boxed{\mathsf{pk}^*})$	$\xrightarrow{\mathsf{msg}_i := (\widetilde{\mathsf{pk}}, \psi_i)}$	$\boxed{\begin{array}{l}\mathsf{pk}^* := \mathsf{Rec}_{\mathsf{KEM}}(\mathsf{sk}_r, \psi_i)\\ (\mathsf{K}^*, \psi^*) \leftarrow \mathsf{Encap}_{\mathsf{EIG}}(\mathsf{pk}^*)\\ d_r^* := (\psi^*, \mathsf{PRF}_{\mathsf{R}}(\mathsf{K}^*, \mathsf{msg}_i))\end{array}}$ $K_i \leftarrow \mathsf{Decap}(\mathsf{sk}_r, \psi_i)$ $(\widetilde{K}, \widetilde{\psi}) \leftarrow \mathsf{Encap}_0(\widetilde{\mathsf{pk}})$ $(K_r, \psi_r) \leftarrow \mathsf{Encap}(\mathsf{pk}_i; \boxed{d_r^*})$ $\mathsf{K}_r := \mathsf{H}(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r, K_i, K_r, \widetilde{K})$ $\boxed{\mathsf{dk}_r := \mathsf{PRF}_{\mathsf{D}}(\mathsf{K}^*, (\mathsf{msg}_i, \mathsf{msg}_r))}$
$\mathsf{st} := (\widetilde{\mathsf{sk}}, K_i)$ $\boxed{\mathsf{aux}_i' := (\mathsf{sk}^*, \mathsf{msg}_i)}$		
$\widetilde{K} \leftarrow \mathsf{Decap}_0(\widetilde{\mathsf{sk}}, \widetilde{\psi})$ $K_r \leftarrow \mathsf{Decap}(\mathsf{sk}_i, \psi_r)$ $\mathsf{K}_i := \mathsf{H}(\mathsf{pk}_i, \mathsf{pk}_r, \mathsf{msg}_i, \mathsf{msg}_r, K_i, K_r, \widetilde{K})$ $\boxed{\begin{array}{l}d_r^* := \mathsf{Rec}_{\mathsf{KEM}}(\mathsf{sk}_i, \psi_r)\\ \text{parse } d_r^* = (\psi^*, h)\\ \mathsf{K}^* \leftarrow \mathsf{Decap}_{\mathsf{EIG}}(\mathsf{sk}^*, \psi^*)\\ \textbf{if } h = \mathsf{PRF}_{\mathsf{R}}(\mathsf{K}^*, \mathsf{msg}_i):\\ \quad \mathsf{dk}_i := \mathsf{PRF}_{\mathsf{D}}(\mathsf{K}^*, (\mathsf{msg}_i, \mathsf{msg}_r))\\ \textbf{else}:\\ \quad \mathsf{dk}_i := \bot\end{array}}$	$\xleftarrow{\mathsf{msg}_r := (\widetilde{\psi}, \psi_r)}$	

Fig. 11. The three-KEM paradigm for AKE (without gray boxes) and the resulting plain AM-AKE with initiator-robustness and relaxed security via our generic construction in Appendix E (with anamorphic algorithms in gray boxes).

Below we will show that the $\mathsf{AKE}_{\mathsf{3K}}$ has partially randomness-recoverable Init and DerR, if the underlying KEM is randomness-recoverable.

Definition 19 (Randomness-Recoverable KEM). *We say that a KEM scheme* $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$ *is randomness-recoverable, if there exists a PPT algorithm* $\mathsf{Rec}_{\mathsf{KEM}}$*, such that for any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{KEM}}$ *and* $(K, \psi) \leftarrow \mathsf{Encap}(\mathsf{pk}; r)$*, it holds that*

$$r = \mathsf{Rec}_{\mathsf{KEM}}(\mathsf{sk}, \psi).$$

With such KEM, it is easy to check that the Init and DerR algorithms of $\mathsf{AKE}_{\mathsf{3K}}$ are partially randomness-recoverable, supported by the following $\mathsf{Rec}_{\mathsf{Init}}$ and $\mathsf{Rec}_{\mathsf{DerR}}$ as per Def. 16.

- $\underline{d_i^* \leftarrow \mathsf{Rec}_{\mathsf{Init}}(\mathsf{pk}_i, \mathsf{sk}_r, \mathsf{msg}_i = (\widetilde{\mathsf{pk}}, \psi_i))}$: it sets $d_i^* := \mathsf{Rec}_{\mathsf{KEM}}(\mathsf{sk}_r, \psi_i)$.
- $\underline{d_r^* \leftarrow \mathsf{Rec}_{\mathsf{DerR}}(\mathsf{pk}_r, \mathsf{sk}_i, \mathsf{msg}_r = (\widetilde{\psi}, \psi_r), \mathsf{st} = (\widetilde{\mathsf{sk}}, K_i))}$: it sets $d_r^* := \mathsf{Rec}_{\mathsf{KEM}}(\mathsf{sk}_i, \psi_r)$.

The correctness of $\mathsf{Rec}_{\mathsf{Init}}$ and $\mathsf{Rec}_{\mathsf{DerR}}$ directly follows from the property of randomness-recoverable KEM.

Then by plugging the $\mathsf{AKE}_{\mathsf{3K}}$ and the ElGamal-KEM $\mathsf{KEM}_{\mathsf{ElG}}$ into our generic construction in Appendix E, we immediately get a plain AM-AKE scheme with initiator-robustness and relaxed security, as shown in Fig. 11 with gray boxes.

Concrete KEM. Finally, it remains to present concrete randomness-recoverable KEM scheme. Randomness-recoverable KEM can be constructed from randomness-recoverable PKE by setting the message to be uniformly chosen encapsulated key. In fact, there are many existing randomness-recoverable PKE schemes [2,18,9], where the randomness-recoverable property are proved in [16].

Table of Contents