Asymmetric Message Franking in the Plain Model: Generic and Efficient Constructions

Milan Gonzalez-Thauvin[1⋆] and Keitaro Hashimoto[2]

[1] ETH Zurich, Switzerland
milan.gonzalez-thauvin@inf.ethz.ch
[2] National Institute of Advanced Industrial Science and Technology (AIST), Japan
keitaro.hashimoto@aist.go.jp

Abstract. Asymmetric Message Franking (AMF), proposed by Tyagi et al. at Crypto'19, is a sort of signature scheme that aims to provide privacy-preserving content moderation in secure messaging applications. In this work, we present the first generic construction of AMF using a public-key encryption scheme, a signature scheme, and a ZAP proof system for NP languages. This construction yields the first AMF scheme provably secure in the plain model from standard assumptions, and has tight security. To improve the efficiency of AMF in the plain model, we build a concrete scheme from asymmetric pairing groups based on our idea for the generic construction. It achieves a signature size of 47 group elements, which is significantly smaller than an instantiation of the generic construction. Also, we provide a variant of the generic construction that yields a post-quantum secure AMF scheme in the plain model from a polynomially hard LWE assumption, demonstrating its feasibility.

Keywords: Asymmetric Message Franking, Secure Messaging, Generic Construction, Plain Model

1 Introduction

Secure messaging has become increasingly popular for online communication because of its stronger security guarantee. As the demand for secure messaging grows, it is exploited for trading illegal/harmful contents and for spreading disinformation. To mitigate such misuse, some secure messaging services have introduced contents-moderation mechanisms [Fac17, GLR17]. Contents moderation mechanism enables a recipient to report that a sender has sent a harmful message to a judge, and if the judge determines that the message violates legal or platform policies, the sender of the reported message may be penalized. To prevent dishonest recipients from falsely accusing compliant senders, a judge must be able to verify that the reported message was indeed sent by the alleged sender. Without such verification, malicious recipients could easily submit fraudulent reports, leading to the punishment of innocent users. Therefore, the recipient must provide verifiable evidence that the message truly originated from the sender in order to convince the judge.

Asymmetric Message Franking. To realize privacy-preserving third-party contents moderation, Tyagi et al. [TGL+19a] introduced asymmetric message franking (AMF). AMF is a sort of signature scheme in which three parties (sender, receiver and judge) are involved. Each of them has its own public and secret key pair, generated using the same KeyGen algorithm (thus, keys are not specific to a role). When the sender sends a message, it generates a signature for it using the signing algorithm Frank with its secret key and intended receiver's and judge's public keys. When the receiver receives a signature, it calls Verify with its secret key to verify the signature. If the receiver receives a harmful or illegal message, he/she can report the message to the judge along with the original signature so that the judge can check its validity using the Judge algorithm and its secret key. As in a standard signature scheme, AMF offers authenticity which ensures that only authorized parties can produce valid signatures.

In addition to these signing and verification algorithms, AMF provides three algorithms Forge, RForge and JForge to guarantee deniability in particular compromise scenarios. That is, their existence allows the sender to claim that the signature could have been forged by another user with one of these forging algorithms. For instance, suppose that a message sent by a sender contains information

⋆ A part of this work was conducted while Milan Gonzalez-Thauvin was affiliated with ENS de Lyon.

Table 1: Expected output of Verify and Judge algorithms of AMF when signing algorithms are honestly executed. The column "Signer's knowledge" indicates the required secret key to create signatures.

Signing algorithm	Signer's knowledge	Expected output		Related properties
		Verify	Judge	
Frank	sk_s	1	1	correctness, sender binding
Forge	—	0	0	universal deniability, unforgeability
RForge	sk_r	1	0	receiver binding, receiver compromise deniability
JForge	sk_j	1	1	judge compromise deniability

exclusively intended for a receiver, who turns out to be rather unreliable. This receiver cannot convince anybody else (except the judge in this case) of the authenticity of the message because the receiver secret key is needed to check the validity of the signature, and because even if the receiver shares its key, a fake signature could have been generated using the same key and a forging algorithm. Therefore, an external party receiving a message, a secret key and a signature (valid with respect to the message and the secret key) from a so-called leaker cannot know if it is being fooled or not. This does not contradict the authenticity property since a receiver who safely kept its secret key can be sure that a valid signature could have been produced only by the real sender.

In summary, AMF schemes satisfy two types of security properties: accountability (which is an extension of authenticity when many verification algorithms are involved) and deniability.

Accountability: AMF must satisfy three accountability notions: unforgeability, receiver binding, and sender binding. Unforgeability guarantees that it should be hard to produce a signature accepted by a receiver without any secret keys. Receiver binding guarantees that an adversarial receiver, who may generate its keys maliciously, cannot produce a signature (on behalf of a specific sender) accepted by a judge without knowing judge nor sender secret keys. Sender binding guarantees that an adversarial sender, who may generate its key maliciously, cannot produce a signature accepted by a receiver but rejected by a judge, without the receiver or judge secret key.

Deniability: AMF must satisfy three deniability notions: universal deniability, receiver compromise deniability, and judge compromise deniability. Universal deniability ensures that anyone can produce a fake signature that is indistinguishable from an honest one to an adversary knowing only the sender's secret key. Receiver compromise deniability ensures that, with the receiver secret key, one can produce a fake signature that is indistinguishable from an honest one to an adversary knowing the sender's and receiver's secret keys. Judge compromise deniability ensures that, with the judge's secret key, one can produce a fake signature that is indistinguishable from an honest one to an adversary knowing the sender's, receiver's and judge's secret keys.

These security notions define expected output of Verify and Judge when checking the validity of a signature honestly generated by one of the four signing algorithms, summarized in Table 1.

For the reasons mentioned above, accountability and deniability are very desirable properties in the context of secure messaging, but a particular care has to be taken to combine them, especially with a judge. One way to achieve this is through AMF schemes, which justifies its interest. Note that most algorithms of an AMF scheme require the public key of every party involved in the franking process, therefore a public key infrastructure is needed so that AMF works properly. Any other form of trusted setup is, a priori, not compulsory.

Previous Work on AMF. As mentioned previously, Tyagi et al. [TGL+19a] designed the first asymmetric message franking scheme. Their signatures are composed of four group elements and one signature of knowledge [CS97, CL06] derived from a Sigma protocol made non-interactive with the Fiat-Shamir heuristic. They realize desired properties by ensuring that the four group elements in the signature satisfy a precise relation (induced by the Sigma protocol) and the security of their scheme relies on knowledge-of-exponent and discrete-logarithm assumptions in the random oracle model. They also

prove that some strong notions of accountability and deniability are not compatible, and provide possible trade-offs.

Some follow-up works [LZH+23,HLZW24] appeared in the next few years. They constructed AMF schemes with different functionalities to adapt to various situations, but kept the same underlying techniques to build them. More specifically, Lai et al. [LZH+23] adapt the notion of asymmetric message franking to the group setting. Their asymmetric *group* message franking (AGMF) scheme is based on a tailor-made primitive (called HPS-based KEM supporting Sigma protocols) and their signatures is linear in the number of participants (they also prove that this drawback is inherent to their formulation of security). Finally, Huang et al. [HLZW24] design an AMF scheme where only a precise set of illegal messages can be moderated, reducing the power of the judge and providing a different balance between accountability and deniability notions. The construction is based on an extension of HPS-based KEM. The security of both schemes also relies on group-based assumptions in the random oracle model.

The previous AMF schemes are built either directly on groups or on tailor-made primitives, all of which are group-based constructions and rely on the random oracle heuristic. In other words, neither an AMF scheme in the plain model nor a generic construction of AMF based on standard primitives is known.

1.1 Our Contributions

In this work, we present a generic construction of an asymmetric message franking scheme in the plain model, i.e., which does not rely on a trusted setup or the random oracle heuristic. Our generic construction is based on a standard signature scheme, a key-private public-key encryption scheme and a ZAP proof system for NP languages. This construction has tight security and can be instantiated from standard assumptions. Especially, we obtain the first tightly-secure AMF scheme in the plain model from standard group-based assumptions and the first post-quantum AMF scheme in the plain model from sub-exponentially hard LWE assumption. To realize an AMF scheme from plain (polynomially-hard) LWE assumption, we further provide another generic construction in which the ZAP for NP is replaced with a ZAP for $\mathsf{NP} \cap \mathsf{coNP}$ languages [CGH+21].

It should be noted, however, that this generic construction is purely theoretical and is not intended to be implemented. In particular, no matter which building blocks are used, the size of the signatures would exceed by far that of previous works. A naive instantiation of our generic construction using ElGamal PKE [ElG85], the structure-preserving signature of Kiltz et al. [KPW15, Section 4.2], and the NIWI proof system for pairing groups of Groth and Sahai[3] [GS08], gives a signature size of roughly 800 group elements. In comparison, a signature of the AMF scheme of Tyagi et al. [TGL+19a], which is secure in the random oracle model, contains 9 group elements and 6 scalars.

To show that our approach can still give rise to efficient constructions in the plain model, we build a concrete AMF scheme from asymmetric pairing groups which achieves a signature size of 47 group elements. The goal of this construction is not to be as efficient as possible, but rather to show a way to come up with an efficient AMF scheme secure in the plain model using our approach. Note that we mainly focus on signature size since the natural way to use AMF is to resort to a quite static public key infrastructure, and send a substantial number of signature (one per message in the context of secure messaging).

In this work, we mainly focus on the AMF definition given in the foundational work of [TGL+19a], in which deniability is guaranteed even when every key is leaked. The consequence is that such scheme cannot ensure unframeability [HLZW24], namely a judge can produce a signature acceptable to both the receiver and the judge. Our generic construction can be adapted to an AMF variant satisfying unframeability, see Appendices B and C for more details.

1.2 Technical Overview

Existing Deniable Signatures Are Insufficient for AMF. To realize deniability, one may consider employing existing deniable signature schemes, such as ring signatures [RST01] or (multi-)designated

[3] For the sake of simplicity, we use the Groth-Sahai proof system here, even though a trusted setup generating a CRS is normally required.

verifier signatures [DHM+20, JSI96]. However, we demonstrate that existing schemes cannot be directly employed to construct the desired AMF scheme. In other words, a generic construction of AMF requires a novel approach specifically tailored to its properties.

Use Ring Signatures as AMF. Ring signatures (RS) [RST01, BKM06, BDH+19, CGH+21] are anonymous signature schemes that conceal the identity of the actual signer. In such schemes, a signer generates a signature by specifying a set of potential signers, called a ring, and it is computationally infeasible to determine which member of the ring produced the signature. This anonymity property provides signers with deniability, which naturally motivates the consideration of AMF schemes based on RS. In particular, a three-party ring signature involving the sender, receiver, and judge can be interpreted as an AMF signature. However, an AMF scheme derived from RS does not achieve receiver binding (cf. the row "RForge" in Table 1). The reason is that the anonymity of RS enables the judge to accept forged signatures created by the receiver as if they were genuine signatures from the sender. Therefore, ring signatures alone are insufficient to realize AMF.

Use Multi-Designated Verifier Signatures as AMF. Multi-designated verifier signatures (MDVS) [DHM+20, CHMR23, HYH25] are signature schemes in which a signer (i.e., the sender) specifies multiple verifiers who are authorized to verify the signature at the time of signing. MDVS provide deniability in the sense that each designated verifier can generate a simulated signature that is indistinguishable from a genuine one. Interpreting the receiver and the judge as the designated verifiers suggests that MDVS could serve as a basis for AMF. A straightforward construction is as follows: the sender generates an MDVS signing key pair, while the receiver and the judge each generate an MDVS verification key pair. The sender then produces a signature using the MDVS signing algorithm with its secret signing key and the public verification keys of the receiver and the judge. Verification is performed by the receiver and the judge using their respective secret verification keys. Moreover, both the receiver and the judge can generate simulated signatures using the MDVS simulation algorithm and their secret keys. However, this naive AMF construction based on MDVS fails to achieve the required deniability property of AMF. Specifically, MDVS allows each designated verifier to distinguish simulated signatures that were generated without its own secret key. In the AMF setting, this means that the receiver can identify signatures simulated by the judge, and vice versa. This violates judge-compromise deniability, which requires that the receiver cannot distinguish between signatures simulated by the judge and genuine signatures from the sender (cf. the row "JForge" in Table 1). Therefore, MDVS alone does not yield the desired AMF scheme. (Note that the AMF scheme derived directly from MDVS satisfies AMF with strong authenticity. A more detailed discussion of AMF from MDVS is provided in Appendix D.)

Our Approach for Generic Construction. Our idea is implementing the desired AMF functionalities summarized in Table 1 with general cryptographic primitives. Below, we give a step-by-step overview of the process. Table 2 summarizes how each element in our AMF signature is computed by each signing algorithm and could prove very useful in understanding the whole picture.

Building Blocks. Our construction uses a standard signature scheme $\mathsf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, a public key encryption (PKE) scheme $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ and a ZAP proof system $\mathsf{ZAP} = (\mathsf{Prove}, \mathsf{Verify})$ for NP languages.

Key Generation. In our construction, an AMF secret key is composed of one signing key (sik) and one decryption key (dk) of the underlying signature scheme and PKE scheme, respectively. An AMF public key is composed of some ZAP randomness (zr), the verification (vk) and encryption keys (ek) related to the signing and decryption keys mentioned above, plus another encryption key that has been generated alone (no associated decryption key), referred to as *orphan* encryption key ($\widetilde{\mathsf{ek}}$)[4].

Step 1: Anticipate the Output of Verification Algorithms. The first step of the construction is to implement the four signing algorithms (Frank and the three forging algorithms) so that Verify and Judge algorithms work as expected in Table 1. As we pointed out above, a precise relation between signing and verification algorithms is induced by the properties of the AMF scheme itself. From this observation, we deduce the following procedure for the four signing algorithms:

[4] In the actual construction, a key pair is generated by a key generation algorithm, and then the decryption key is erased.

Table 2: Overview of how the four different signing algorithms in our generic construction generate a standard signature and PKE ciphertexts in AMF signatures. Each cell indicates the value used as $\bullet$ in these algorithms listed in the first column (except that Forge sets $\sigma := 0$). The two last rows give the exhaustive assignment of b_r and b_j depending on the signing algorithm.

	Frank	Forge	RForge	JForge	
$\sigma \leftarrow \mathsf{SIG.Sign}(\bullet, m\|\|\mathsf{pk}_r)$	sik_s	$\sigma := 0$	sik_r	sik_j	
$c_\sigma \leftarrow \mathsf{PKE.Enc}(\bullet, \sigma)$	$\widetilde{\mathsf{ek}_s}$	$\widetilde{\mathsf{ek}_s}$	$\widetilde{\mathsf{ek}_j}$	$\widetilde{\mathsf{ek}_s}$	
$c_r \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_r, \bullet)$	1	0	1	1[6]	$\rightsquigarrow b_r$
$c_j \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_j, \bullet)$	1	0	0	1	$\rightsquigarrow b_j$

1. Generate a signature σ for the underlying message m using the available signing key sik as $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sik}, m)$ (or set $\sigma := 0$ for the external party using Forge). For instance, a judge performing the JForge algorithm will use its signing key sik_j to compute $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sik}_j, m)$.
2. Define a bit b_r as 1 or 0 depending on whether the receiver *should* accept or not this signature. For instance, RForge should produce a signature accepted by a receiver computing Verify in order to ensure receiver compromise deniability, so set $b_r := 1$.
3. Define another bit b_j as 1 or 0 depending on whether the judge *should* accept or not this signature. For instance, RForge should produce a signature rejected by a judge computing Judge to ensure receiver binding, so set $b_j := 0$.

The first row of Table 2 indicates which signing key is used to generate signature σ, and the last two rows give the exhaustive assignment of b_r and b_j depending on the signing algorithm.

Step 2: Encrypt Variables using Relevant Encryption Keys. Then, signing algorithms shall encrypt these three variables to make them accessible to intended parties only. For instance, only a judge should learn b_j since it is the only party who needs such information. Therefore, every algorithm shall compute c_r, c_j the encryption of b_r and b_j using the receiver encryption and judge encryption keys, respectively as $c_r \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_r, b_r)$ and $c_j \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_j, b_j)$. Further, no one should ever know who produced the signature (to ensure deniability properties). Thus, the signature σ shall be encrypted using the *orphan* encryption key of the sender as $c_\sigma \leftarrow \mathsf{PKE.Enc}(\widetilde{\mathsf{ek}}_s, \sigma)$. Note that the decryption key of an orphan encryption key may stored in some security games to extract a signature from c_σ in a forged AMF signature.[5] We remark that a dishonest sender would be the only one hurt if it decides to keep the associated decryption key since it will totally lose deniability to parties knowing this key.

Step 3: Add a ZAP Proof to Hide the Signer and Ensure Consistency. The signing algorithms shall add a ZAP proof π to ensure that the first two steps were computed correctly. In other words, the ZAP proof guarantees that the ciphertexts c_σ, c_r, c_j reflect one of the four columns of Table 2 without revealing which. The randomness used to produce the ZAP proof shall be the one included in the *judge* public key (zr_j) since it is the only party with no interest in producing non-sound poofs. Note that the witness indistinguishability of a ZAP proof holds even with tweaked randomness.

Step 4: Implement the Two Verification Algorithms Accordingly. The definitions of Verify and Judge are straightforward given steps 1, 2 and 3. Both shall first check the validity of the ZAP proof and then decrypt the bit ciphertext intended to them in order to check that is it equal to 1.

Step 5: Add Construction Tricks for Provable Security. We change here a little the construction described above in order to make our scheme provably secure.

– **Concatenate receiver's identity with the message.** In accountability proofs, a forgery produced by an adversary for the AMF scheme will be used to forge a signature for the signature scheme. However, in AMF accountability games, a forgery is valid if it has not been queried to the signing oracle *for a specific receiver* while the signature's unforgeability game simply forbids

[5] Similar technique appeared in the security proofs of ring signatures [BKM06, BHJ+19].

[6] Remark that setting b_r to be 0 in the JForge algorithm defines an AMF variant with strong unforgeability and unframeability. Roughly speaking, this modification prevents the judge from impersonating the sender. On the other hand, it weakens judge compromise deniability. See Appendices B and C for more details.

reusing a signature. Therefore, to make AMF signatures dependent on the receiver, message m is signed with its specified receiver's public key pk_r as $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sik}, m||\mathsf{pk}_r)$.

- **Use the judge orphan encryption key in** RForge (and update the ZAP proof accordingly). Roughly speaking, soundness of ZAP proofs guarantees that the only way to break sender binding is generating an AMF signature by using the RForge algorithm. Since the sender binding adversary does not know the receiver's secret key, a valid forgery breaking sender binding gives a valid forgery for the underlying signature scheme w.r.t the receiver's signing key. However, in the sender binding security game, the adversary outputs the sender key pair himself, making it impossible for the reduction to recover an encrypted σ since it does not have access to the relevant decryption key. We solve this problem by enforcing the receiver to use the judge orphan key $\widetilde{\mathsf{ek}}_j$ instead of the sender's orphan key $\widetilde{\mathsf{ek}}_s$. This modification does not reveal anything important since a judge can already distinguish a signature produced by Frank and RForge. To hide the encryption key used for c_σ, we additionally assume PKE has key-privacy (a.k.a anonymity) [BBM00].
- **Duplicate c_σ and add a trapdoor to the ZAP.** To prove that adversaries cannot notice any difference between the outputs of the Frank algorithm and a forging algorithm in deniability properties, we have to hop from game to game using indistinguishability of PKE *and* witness-indistinguishability of ZAP. However, it is not possible to use both properties at the same time. To overcome this problem, we add a PKE ciphertext in the public key of the judge such that one can generate a valid proof if that ciphertext encrypts 1 (i.e., trapdoor open) by using the associated randomness as witness. Of course, in an honestly generated key pair, the ciphertext encrypts 0 (i.e., trapdoor closed). During the security proofs, we first replace the ciphertext of 0 in the judge public key with the ciphertext of 1 (i.e., open the trapdoor), simulate a valid proof using the associated randomness, swap the plaintext of c_r and/or c_j, and finally, we close the trapdoor. This technique allows us to prove universal deniability and receiver compromise deniability. However, it does not work to prove judge compromise deniability because the judge key pair is adversarially chosen in the judge compromise deniability game. Therefore, to prove judge compromise deniability, we resort to another technique: we now duplicate c_σ (the only different ciphertext between Frank and JForge) in AMF signatures and change its content while its twin is used in the ZAP proof, and then swap witnesses. Note that the duplication technique does not work for c_r and c_j since the proof of sender binding must exploit the fact that some forgery is rejected by the judge, which in this case would imply that at least one of the ciphertext encrypts 0, but not necessarily both. This is why we implement two techniques to swap the witness of ZAP.

The final AMF public key of our AMF scheme consists of a signature verification key vk, two PKE encryption keys $\mathsf{ek}, \widetilde{\mathsf{ek}}$, a ZAP randomness zr and a PKE ciphertext td which is a trapdoor for security proofs. The final AMF signature Σ consists of four PKE ciphertexts c_σ, c'_σ encrypting a signature, c_r encrypting a bit for the receiver, c_j encrypting a bit for the judge, and a ZAP proof π ensuring the consistency of these ciphertexts. Combining with the known results on the underlying building blocks, our generic construction gives the first (tightly-secure) AMF scheme from pairing-based assumptions in the plain model and the first post-quantum secure AMF scheme from (sub-exponentially hard) Learning With Errors (LWE) assumption in the plain model.

We would like to point out that the core idea of our approach, namely, encrypting expected output of the different verification algorithms and ensuring consistency with a zero-knowledge proof, is simple but, to the best of our knowledge, unprecedented in the literature. Moreover, constructing such primitive generically in the plain model is non-trivial because, in the plain model, we cannot rely on a trusted setup, and the trusted users change depending on the security property. In fact, our AMF construction realizes all security properties by carefully observing trusted key pairs and employing different techniques for each property. This approach could be of independent interest and might be particularly useful for other types of multi-designated verifier signature schemes with specific roles or privileges.

From Generic to Efficient Construction. The generic construction mentioned above is mainly of theoretical interest and is not usable *as is* in practice due to the size of the signatures. To achieve smaller signature size, we build a concrete scheme based on pairing groups and inspired by the idea of the generic construction. We stress that this construction *is not an instantiation* of the generic one, and we explain why in the following paragraph:

Groth-Sahai proof systems [GS08] are usually considered quite efficient, but still not enough for our generic construction which is why it was necessary to resort to a more tailored proof system. We decided to use the ZAP proof system of Couteau and Hartman [CH20a] for algebraic languages, a restricted class of languages in which for instance, a group element (like a signature) cannot be part of a witness (which is the case in our generic construction). We thus had to eliminate the signature and embed the unforgeability feature directly into the soundness of the ZAP; however, algebraic languages contain little more than linear languages and disjunctions of the latter. Moreover, linear languages are insufficient to construct unforgeable signatures [DHH+21], so we designed a custom-made zero-knowledge-based Waters-like [Wat05] signature scheme (that crucially uses disjunction) inspired by a work from Dietz and Hofheinz [DH26]. In this new Waters-like signature, the group elements of the public key are replaced by ciphertexts (or commitments) with the same type of homomorphic properties. The group exponents used as trapdoor are replaced by plaintexts (and randomness) related to the ciphertexts, and a message m is signed by proving that its Waters hash (the sum of all the i-th ciphertexts of the public key if the i-th bit of m is 1) is associated with a non-zero plaintext. This verification step can be performed by checking a small number of linear equations, which will then be embedded directly into the big ZAP proof for the AMF scheme, making the encryption and extraction of the underlying signature no longer necessary. Finally, we chose ElGamal PKE for both the signing technique mentioned above and the encryption component of the AMF construction, and added a (generic) collision-resistant hash function to make signatures depend on the message and the receiver's identity.

As a result, we obtain an efficient AMF scheme secure under pairing-based assumptions in the plain model, with a signature containing 47 group elements. Again, we would like to point out that to the best of our knowledge, using this type of proof system is unprecedented in the context of constructing some form of short signature of knowledge (for algebraic languages) *in the standard model*.

Using ZAP for $\mathsf{NP} \cap \mathsf{coNP}$ for AMF from Plain LWE. Our generic construction gives an AMF scheme based on LWE assumption, but it relies on sub-exponential hardness of LWE problem because there is no ZAP argument for NP from polynomial hardness of LWE problem. To overcome this drawback, we finally provide an AMF scheme secure under the plain (polynomially hard) LWE assumption. To this end, we replace the ZAP argument for NP in our generic construction with the ZAP for $\mathsf{NP} \cap \mathsf{coNP}$ described in [CGH+21]. This ZAP system is defined for two NP languages $\mathcal{L}$ and $\widetilde{\mathcal{L}}$, such that (in our case) $\widetilde{\mathcal{L}}$ is a subset of the complement of $\mathcal{L}$. This argument system makes it hard to compute a statement x from a non-witness $\widetilde{w}$ (i.e., a witness for the $\widetilde{\mathcal{R}}$ associated with $\widetilde{\mathcal{L}}$) such that $(x, \widetilde{w}) \in \widetilde{\mathcal{R}}$. In other words, in a security game, being able to fix in advance a non-witness $\widetilde{w}$ will make any ZAP argument for a statement x sound as soon as $(x, \widetilde{w}) \in \widetilde{\mathcal{R}}$. In our case, the non-witnesses will be composed of decryption keys. Indeed, if these decryption keys are related to the encryption keys in the AMF public keys, then decrypting the ciphertexts in the AMF signature described above is enough to determine if they reflect one of the four columns of Table 2.

1.3 More Related Work

There exist many cryptographically-secure content-moderation schemes, AMF is only one of them. A famous content-moderation mechanism is Facebook Messenger's symmetric message franking (SMF) scheme [Fac17, GLR17, DGRW18]. The SMF scheme is based on symmetric-key primitives and works as follows. When a sender sends an encrypted message, the content delivery server computes a commitment of the ciphertext and the identities of both the sender and receiver with the server's secret commitment key. The commitment is sent to the receiver along with the ciphertext. Then, the receiver sends the commitment and the message to the server as a report. The server verifies the commitment with its commitment key. Since the commitment is generated by the server when the message has been sent, its validity guarantees that the message is actually sent from the committed sender to the receiver. The SMF scheme realized the described functionalities, but it has two drawbacks. First, to create reports, both sender and receiver's identities must be revealed to the server when messages are sent, reducing privacy guarantee of the system. Second, only the server can verify reported messages due to the usage of symmetric key for generating and verifying commitments. To overcome these drawbacks, Tyagi et al. [TGL+19a] have introduced AMF schemes.

Issa et al. [IAV22] introduced AMF with pre-processing and proposed a concrete protocol Hecate based on standard cryptographic primitives. While it is called AMF, its system model is very different from the AMF model by Tyagi et al. In AMF with pre-processing, the judge periodically issues authentication tokens to each sender by using its secret key, and the sender sends a message with such a token. Later, the receiver uses the attacked token to report the message, and the judge verifies a reported token with its secret key to identify the sender of the message. That is, the issuer and verifier of tokens are identical. On the other hand, in the AMF model, the judge publishes its public verification key only once, and no interaction exists between the judge and each sender. We can see AMF with pre-processing relaxes the original AMF model, and this explains why Hecate can be realized with standard primitives in the plain model.

1.4 Paper Organization

The rest of the paper is organized as follows. In Section 2, we review notation and basic cryptographic primitives used in the paper. In Section 3, we recall the definition of AMF schemes from Tyagi et al.'s original paper [TGL+19a, TGL+19b]. In Section 4, we present the proposed generic construction of AMF based on signature scheme, PKE and ZAP proof system, and provide its security proofs. In Section 5, we provide an efficient pairing-based scheme in the plain model. In Section 6, we adapt the generic construction to provide an AMF scheme secure from plain LWE.

2 Preliminaries

2.1 Notations

λ denotes the security parameter. For a distribution $\mathcal{D}$, we denote by $s \leftarrow\!\!\$\ \mathcal{D}$ the process of sampling an element s from $\mathcal{D}$. For a finite set S, we denote by $|S|$ the number of elements in S, and denote by $s \leftarrow\!\!\$\ S$ the process of uniformly sampling an element s from S. We say that an algorithm is efficient if it runs in probabilistic polynomial time. For any efficient algorithm A, we write $A(x;r)$ for the process of A on input x with randomness r, and use $y \leftarrow A(x)$ to denote the process of running A on input x with uniformly sampled randomness r, and assigning the result to y. A function $f : \mathbb{N} \to [0,1]$ is negligible if for all positive polynomials $p(\lambda)$ and all sufficiently large $\lambda \in \mathbb{N}$, we have $f(\lambda) < 1/p(\lambda)$. We use negl to denote unspecified negligible function.

2.2 Public-Key Encryption

Definition 1 (Public-Key Encryption). *A public-key encryption scheme* PKE *consists of the following three algorithms:*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{ek}, \mathsf{dk})$*: The probabilistic key generation algorithm takes a security parameter* 1^λ *as input and outputs a pair of encryption and decryption keys* $(\mathsf{ek}, \mathsf{dk}) \in \mathcal{EK} \times \mathcal{DK}$*.*
- $\mathsf{Enc}(\mathsf{ek}, \mathsf{m}) \to \mathsf{ct}$*: The probabilistic encryption algorithm takes an encryption key* ek*, and a message* m *as input and outputs a ciphertext* ct*.*
- $\mathsf{Dec}(\mathsf{dk}, \mathsf{ct}) \to \mathsf{m}/\perp$*: The deterministic decryption algorithm takes a decryption key* dk*, and a ciphertext* ct *as input and outputs a message* m *or an error symbol* $\perp$*.*

Definition 2 (Perfect Correctness). *A PKE scheme* PKE *is* perfectly correct *if for all key* $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and any message* m*, it holds that* $\Pr[\mathsf{Dec}(\mathsf{dk}, \mathsf{Enc}(\mathsf{ek}, \mathsf{m})) = \mathsf{m}] = 1$*.*

Definition 3 ($\mathsf{MC\text{-}IND\text{-}CPA}$ Security [BBM00]). *We formalize multi-challenge IND-CPA security in the game* $\mathsf{MC\text{-}IND\text{-}CPA}^{\mathcal{A},b}_{\mathsf{PKE}}$ *in Figure 1. For an adversary* $\mathcal{A}$ *and a PKE scheme* PKE*, we define the advantage of* $\mathcal{A}$ *against* PKE *as*

$$\mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) := \left|\Pr\left[\mathsf{MC\text{-}IND\text{-}CPA}^{\mathcal{A},0}_{\mathsf{PKE}} \Rightarrow 1\right] - \Pr\left[\mathsf{MC\text{-}IND\text{-}CPA}^{\mathcal{A},1}_{\mathsf{PKE}} \Rightarrow 1\right]\right|.$$

PKE *is* $\mathsf{MC\text{-}IND\text{-}CPA}$ secure *if* $\mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$ *for any efficient adversary* $\mathcal{A}$*.*

We will use a PKE scheme with key-privacy [BBDP01]. Below, we use the multi-challenge version of $\mathsf{IK\text{-}CPA}$ security, which we call $\mathsf{MC\text{-}IK\text{-}CPA}$ security.

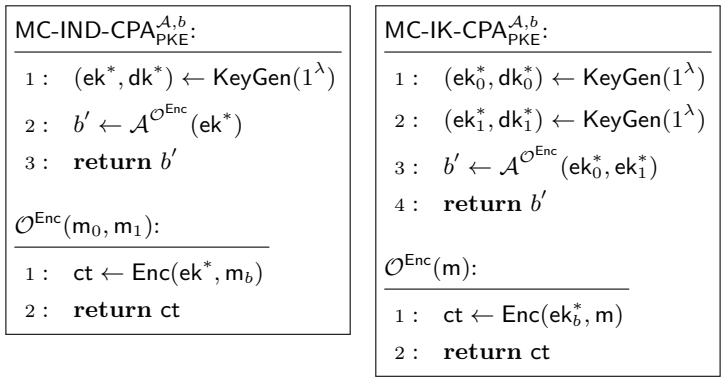


Fig. 1: Security games for PKE scheme.

Definition 4 (MC-IK-CPA Security). *We formalize multi-challenge IK-CPA security in the game* $\mathsf{MC\text{-}IK\text{-}CPA}^{\mathcal{A},b}_{\mathsf{PKE}}$ *in Figure 1. For an adversary* $\mathcal{A}$ *and a PKE scheme* PKE*, we define the advantage of* $\mathcal{A}$ *against* PKE *as*

$$\mathsf{Adv}^{\mathsf{MC\text{-}IK\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) \coloneqq \left|\Pr\left[\mathsf{MC\text{-}IK\text{-}CPA}^{\mathcal{A},0}_{\mathsf{PKE}} \Rightarrow 1\right] - \Pr\left[\mathsf{MC\text{-}IK\text{-}CPA}^{\mathcal{A},1}_{\mathsf{PKE}} \Rightarrow 1\right]\right|.$$

PKE *is* MC-IK-CPA secure *if* $\mathsf{Adv}^{\mathsf{MC\text{-}IK\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$ *for any efficient adversary* $\mathcal{A}$*.*

Remark 1. From the standard hybrid arguemnt, it can be proved that single-challenge IND-CPA (resp. IK-CPA) security implies MC-IND-CPA (resp. MC-IK-CPA) security with polynomial security loss [BBM00, Theorem 1] (resp. Appendix E).

2.3 Signature Scheme

Definition 5 (Signature Scheme). *A signature scheme* SIG *consists of the following three algorithms:*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{vk}, \mathsf{sik})$*: The probabilistic key generation algorithm takes a security parameter* 1^λ *as input and outputs a pair of verification and signing keys* $(\mathsf{vk}, \mathsf{sik})$*.*
- $\mathsf{Sign}(\mathsf{sik}, \mathsf{m}) \to \sigma$*: The probabilistic signing algorithm takes a signing key* sik *and a message* m *as input and outputs a signature* σ*.*
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{m}, \sigma) \to \{0, 1\}$*: The deterministic verification algorithm takes a verification key* vk*, a message* m*, and a signature* σ *as input and outputs* 1 *or* 0*.*

In this work we only consider signature schemes having an additional key-validity algorithm:

- $\mathsf{WellFormed}(\mathsf{vk}, \mathsf{sik}) \to \{0, 1\}$*: The deterministic well-formedness algorithm takes a pair of verification and signing keys* $(\mathsf{vk}, \mathsf{sik})$ *as input and outputs* 1 *or* 0*.*

Definition 6 (Correctness). *A signature scheme* SIG *is* correct *if it holds that* $\mathsf{WellFormed}(\mathsf{KeyGen}(1^\lambda)) = 1$ *with probability* 1*; and if for any key pair* $(\mathsf{vk}, \mathsf{sik})$ *such that* $\mathsf{WellFormed}(\mathsf{vk}, \mathsf{sik}) = 1$ *and any message* m*,*

$$\Pr[\mathsf{Verify}(\mathsf{vk}, \mathsf{m}, \mathsf{Sign}(\mathsf{sik}, \mathsf{m})) = 1] \geq 1 - \varepsilon_{\mathsf{SIG}}$$

for some correctness error $\varepsilon_{\mathsf{SIG}} \leq \mathsf{negl}(\lambda)$*, where the probability is taken over the random coins of* Sign*.*

Definition 7 (EUF-CMA Security). *We formalize unforgeability in the game* $\mathsf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{SIG}}$ *in Figure 2. For an adversary* $\mathcal{A}$ *and a signature scheme* SIG*, we define the advantage of* $\mathcal{A}$ *against* SIG *as*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A}) \coloneqq \Pr\left[\mathsf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{SIG}} \Rightarrow 1\right].$$

SIG *is* EUF-CMA secure *if* $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$ *for any efficient adversary* $\mathcal{A}$*.*

$\mathsf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{SIG}}$:

1 : $L \coloneqq \emptyset$
2 : $(\mathsf{vk}^*, \mathsf{sik}^*) \leftarrow \mathsf{KeyGen}(1^\lambda)$
3 : $(\mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Sign}}}(\mathsf{vk}^*)$
4 : **if** $\mathsf{Verify}(\mathsf{vk}^*, \mathsf{m}^*, \sigma^*) = 1 \wedge \mathsf{m}^* \notin L$ **then return** 1
5 : **else return** 0

$\mathcal{O}^{\mathsf{Sign}}(\mathsf{m})$

1 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{sik}^*, \mathsf{m})$
2 : $L \coloneqq L \cup \{\mathsf{m}\}$
3 : **return** σ

Fig. 2: Unforgeability game for signature schemes.

$\mathsf{Sound}^{\mathcal{A}}_{\mathsf{ZAP}}$:

1 : $\mathsf{zr} \leftarrow\!\!\$\ \{0,1\}^\ell$
2 : $(\mathsf{x}, \pi) \leftarrow \mathcal{A}(1^\lambda, \mathsf{zr})$
3 : **if** $\mathsf{x} \notin \mathcal{L} \wedge \mathsf{Verify}(\mathsf{zr}, \mathsf{x}, \pi) = 1$ **then**
4 : **return** 1
5 : **else**
6 : **return** 0

$\mathsf{WI}^{\mathcal{A},b}_{\mathsf{ZAP}}$:

1 : $\mathsf{zr} \leftarrow \mathcal{A}_1(1^\lambda)$
2 : $b' \leftarrow \mathcal{A}_2^{\mathcal{O}^{\mathsf{Prove}}}(\mathsf{zr})$
3 : **return** b'

$\mathcal{O}^{\mathsf{Prove}}(\mathsf{x}, \mathsf{w}_0, \mathsf{w}_1)$

1 : **if** $\mathcal{R}(\mathsf{x}, \mathsf{w}_0) \vee \mathcal{R}(\mathsf{x}, \mathsf{w}_1) = 0$ **then**
2 : **return** $\perp$
3 : **return** $\mathsf{Prove}(1^\lambda, \mathsf{zr}, \mathsf{x}, \mathsf{w}_b)$

Fig. 3: Security games for ZAP proof system: adaptive soundness (left) and witness-indistinguishability (right).

2.4 The ZAP Proof System

Let $\mathcal{X}$ be a set and $\mathcal{L} \subseteq \mathcal{X}$ be an NP language with witness space $\mathcal{W}$ and an efficiently recognizable relation $\mathcal{R}$, i.e., $\mathcal{L} = \{\mathsf{x} \in \mathcal{X} : \exists \mathsf{w} \in \mathcal{W} \text{ s.t. } \mathcal{R}(\mathsf{x}, \mathsf{w}) = 1\}$. A ZAP is a public-coin 2-round protocol for proving membership of $\mathsf{x} \in \mathcal{L}$, introduced by Dwork and Naor [DN00].

Definition 8 (ZAP). *A ZAP proof system* ZAP *for the* NP *language* $\mathcal{L}$ *consists of the following two algorithms associated with a polynomial* $\ell = \ell(\lambda)$*:*

- $\mathsf{Prove}(1^\lambda, \mathsf{zr}, \mathsf{x}, \mathsf{w}) \to \pi$*: The probabilistic prover algorithm takes a security parameter* 1^λ*, a string* $\mathsf{zr} \in \{0,1\}^\ell$*, a statement* $\mathsf{x} \in \mathcal{X}$*, and a witness* $\mathsf{w} \in \mathcal{W}$ *as input and outputs a proof* π*. We will omit the first argument* 1^λ *when it is clear from the context.*
- $\mathsf{Verify}(\mathsf{zr}, \mathsf{x}, \pi) \to \{0,1\}$*: The deterministic verifier algorithm takes a string* $\mathsf{zr} \in \{0,1\}^\ell$*, a statement* $\mathsf{x} \in \mathcal{X}$ *and a proof* π *as input and outputs a bit* $b \in \{0,1\}$*.*

In this work, we require that the Prove *algorithm always aborts if* $\mathcal{R}(\mathsf{x}, \mathsf{w}) = 0$*.*

Definition 9 (Perfect Correctness). *A ZAP proof system* ZAP *is* perfectly correct *if for all* $\lambda \in \mathbb{N}$*, all statements* $\mathsf{x} \in \mathcal{X}$ *and witnesses* $\mathsf{w} \in \mathcal{W}$ *such that* $\mathcal{R}(\mathsf{x}, \mathsf{w}) = 1$ *and* $\mathsf{zr} \in \{0,1\}^\ell$*,* $\Pr\big[\mathsf{Verify}(\mathsf{zr}, \mathsf{x}, \mathsf{Prove}(1^\lambda, \mathsf{zr}, \mathsf{x}, \mathsf{w})) = 1\big] = 1$*.*

Definition 10 (Adaptive Soundness). *We formalize adaptive soundness in the game* $\mathsf{Sound}^{\mathcal{A}}_{\mathsf{ZAP}}$ *in Figure 3. For an adversary* $\mathcal{A}$ *and a ZAP proof system* ZAP*, we define the advantage of* $\mathcal{A}$ *against* ZAP *as*

$$\mathsf{Adv}^{\mathsf{Sound}}_{\mathsf{ZAP}}(\mathcal{A}) \coloneqq \Pr\Big[\mathsf{Sound}^{\mathcal{A}}_{\mathsf{ZAP}} \Rightarrow 1\Big].$$

We say ZAP *is* adaptively sound *if* $\mathsf{Adv}^{\mathsf{Sound}}_{\mathsf{ZAP}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$ *for any efficient adversary* $\mathcal{A}$*.*

Definition 11 (Witness-Indistinguishability). *We formalize witness indistinguishability in the game* $\mathsf{WI}^{\mathcal{A},b}_{\mathsf{ZAP}}$ *in Figure 3. For an adversary* $\mathcal{A}$ *and a ZAP proof system* ZAP*, we define the advantage of* $\mathcal{A}$ *against* ZAP *as*

$$\mathsf{Adv}^{\mathsf{WI}}_{\mathsf{ZAP}}(\mathcal{A}) \coloneqq \Big|\Pr\Big[\mathsf{WI}^{\mathcal{A},0}_{\mathsf{ZAP}} \Rightarrow 1\Big] - \Pr\Big[\mathsf{WI}^{\mathcal{A},1}_{\mathsf{ZAP}} \Rightarrow 1\Big]\Big|.$$

ZAP *is* statistically witness-indistinguishable *if* $\mathsf{Adv}^{\mathsf{WI}}_{\mathsf{ZAP}}(\mathcal{A}) \leq \varepsilon_{\mathsf{WI}}$ *for any computationally unbounded adversary* $\mathcal{A}$ *and* $\varepsilon_{\mathsf{WI}}$ *negligible in* λ. ZAP *is* perfectly witness-indistinguishable *if* $\varepsilon_{\mathsf{WI}} = 0$

Statistical ZAPs for NP can be constructed from pairing-based assumptions [CKSU21], group-based assumptions without pairings [JJ21, CKSU21], indistinguishability obfuscation (with one-way functions) [BP15], and from (sub-exponentially-hard) lattice-based assumptions [LVW19, BFJ+20, GJJM20].

2.5 Collision-Resistant Hash Functions

Definition 12 (Collision-Resistant Hash Function). *A family of keyed functions* $\mathsf{H}(\mathsf{hk}, x) : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda$ *with the key generation algorithm* $\mathsf{H.KeyGen}(1^\lambda)$ *is called a* collision-resistant hash function *iff it satisfies the following properties:*

- $\mathsf{H}(\mathsf{hk}, x)$ *can be computed in polynomial time.*
- *For any efficient adversary* $\mathcal{A}$*, it holds that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{CR}}_{\mathsf{H}}(\mathcal{A}) &\coloneqq \Pr\left[\mathsf{H}(\mathsf{hk}, x) = \mathsf{H}(\mathsf{hk}, x') \wedge x \neq x' : \begin{array}{c}\mathsf{hk} \leftarrow \mathsf{H.KeyGen}(1^\lambda), \\ (x, x') \leftarrow \mathcal{A}(1^\lambda, \mathsf{hk})\end{array}\right] \\ &= \mathsf{negl}(\lambda).\end{aligned}$$

2.6 Preliminaries for Efficient Constructions

Groups and Pairings. We denote $\mathcal{PG} \coloneqq (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ by the pairing group where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are finite groups of prime order p with generators g_1, g_2, $g_T \coloneqq e(g_1, g_2)$ respectively and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map. All groups are written in implicit notation: we write $[1]_i \coloneqq g_i$ and $[x]_i \coloneqq x \cdot g_i$ for all $x \in \mathbb{Z}_p$ and $i \in \{1, 2, T\}$. If the group is clear from context, we will omit the index. We write $[x]_1 \bullet [y]_2 \coloneqq e([x]_1, [y]_2) = [xy]_T$ for pairings. The implicit notation also extends to matrices and vectors. For $\mathbf{A} \in \mathbb{Z}_p^{n \times t}$, $\mathbf{A} = (a_{ij})$, let $[\mathbf{A}]_k = ([a_{ij}]_k) \in \mathbb{G}_k^{n \times t}$ for $k \in \{1, 2, T\}$ and we also extend the pairing notation from above to $[\mathbf{A}]_1 \bullet [\mathbf{B}]_2 \coloneqq e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{AB}]_T$ for matrices $\mathbf{A} \in \mathbb{Z}_p^{n \times t}$ and $\mathbf{B} \in \mathbb{Z}_p^{t \times m}$. Furthermore, we extend the implicit notation to linear (multivariate) polynomials. Let $\mathcal{P}_l \coloneqq \left\{[a_0] + \sum_{i=1}^{l} a_i X_i\right\} \subset \mathbb{G}[\mathbf{X} = (X_1, \ldots, X_l)]$ be the set of linear multivariate polynomials over $\mathbb{G}$ in l variables. For $f \in P_l$ and $\mathbf{y} = (y_1, \ldots, y_l) \in \mathbb{Z}_p^l$, we define the evaluation of f in y as applying the group operation in the exponent, i.e.,

$$f([\mathbf{y}]) = [a_0] + \sum_{i=1}^{l} a_i [y_i].$$

For a matrix $\mathbf{A} = (a_{i,j}) \in \mathcal{P}_l^{n \times t}$, the evaluation of the matrix (or vector) over $\mathcal{P}_l$ in a vector $\mathbf{y} \in \mathbb{G}_l$ denotes the evaluation of all entries in the given vector, i.e., $\mathbf{A}(\mathbf{y}) \coloneqq (a_{i,j}(\mathbf{y})) \in \mathbb{G}^{n \times t}$.

Computational Assumptions. We recall the Decisional Diffie-Hellman (DDH) assumption and extended kernel matrix Diffie-Hellman (extKerMDH) assumption.

Definition 13 (Decisional Diffie-Hellman Assumption). *Let* $\mathbb{G}$ *be a group with prime order* p *and* g *be its generator. The Decisional Diffie-Hellman (*DDH*) assumption holds in* $\mathbb{G}$*, if for all efficient adversaries* $\mathcal{A}$*, the following probability is negligible.*

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}}(\mathcal{A}) \coloneqq \\ &|\Pr[\mathcal{A}(g, [x], [y], [xy]) = 1 | x, y \leftarrow\!\!\$\ \mathbb{Z}_p] - \Pr[\mathcal{A}(g, [x], [y], [z]) = 1 | x, y, z \leftarrow\!\!\$\ \mathbb{Z}_p]|\end{aligned}$$

Definition 14 (Extended Kernel Matrix Diffie-Hellman Assumption [CH20a]). *Let* $l, k \in \mathbb{N}$*,* $s \in \{1, 2\}$*,* $\mathcal{PG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ *and* $\mathcal{D}_k$ *be a matrix distribution (that outputs matrices*

over $\mathbb{G}^{k+1\times k}$ of full rank k in polynomial time). The $\mathcal{D}_k$-l-ExtKerMDH assumption holds in $\mathbb{G}_s$, if for all efficient adversaries $\mathcal{A}$, the following probability is negligible.

$$\mathsf{Adv}^{\mathsf{ExtKerMDH}}_{\mathbb{G}_s}(\mathcal{A}) \coloneqq \Pr\left[\begin{array}{c|c} [\mathbf{C}]_{3-s} \in \mathbb{G}_{3-s}^{l+1\times k+l+1} \wedge [\mathbf{B}]_s \in \mathbb{G}_s^{l\times k} & \mathbf{D} \leftarrow\!\$\, \mathcal{D}_k \\ \wedge [\mathbf{C}]_{3-s} \bullet [\mathbf{D}']_s = 0 & ([\mathbf{C}]_{3-s}, [\mathbf{B}]_s) \leftarrow \mathcal{A}(\mathcal{PG}, [\mathbf{D}]_s) \\ \wedge \mathsf{rank}(\mathbf{C}) \geq l+1 & [\mathbf{D}']_s \coloneqq [\frac{\mathbf{D}}{\mathbf{B}}]_s \end{array}\right]$$

The probability is taken over then randomness of $\mathcal{A}, \mathcal{D}_k$.

Distribution $\mathcal{L}_1$. Similarly to [CH20a], we define the distribution $\mathcal{L}_1$ in $\mathbb{G}_s^{2\times 1}$ (for $s \in \{1,2\}$) as follow, and use it to fix the $\mathcal{L}_1$-l-ExtKerMDH assumption in $\mathbb{G}_s$.

$$\mathcal{L}_1 = \left\{ \begin{pmatrix} g_s \\ e \end{pmatrix} \quad \text{with } e \leftarrow\!\$\, \mathbb{G}_s \right\}$$

ElGamal Public Key Encryption. We recall the ElGamal PKE and its security properties (using additive notation and component-wise multiplication). In this work we use the variant for which plaintexts are scalars and only a few them (here 0 and 1) can be decrypted by computing a discrete logarithm the naive way.

Definition 15 (ElGamal PKE [ElG85]). *Let $\mathbb{G}$ be a group with prime order p and g be its generator. The description of ElGamal PKE is as follows.*

- $\mathsf{KeyGen}(1^\lambda)$*: Sample* $\mathsf{dk} \leftarrow\!\$\, \mathbb{Z}_p$ *and computes* $\mathsf{ek} \coloneqq \mathsf{dk} \cdot g$*. Outputs* $(\mathsf{ek}, \mathsf{dk})$*.*
- $\mathsf{Enc}(\mathsf{ek}, \mathsf{m} \in \mathbb{Z}_p)$*: Sample* $r \leftarrow\!\$\, \mathbb{Z}_p$ *and output* $\mathsf{ct} \coloneqq (r \cdot g, \mathsf{m} \cdot g + r \cdot \mathsf{ek}) \in \mathbb{G}^2$*.*
- $\mathsf{Dec}(\mathsf{dk}, \mathsf{ct})$*: Let* $(\mathsf{ct}_0, \mathsf{ct}_1) \coloneqq \mathsf{ct}$*. Output the discrete logarithm of* $\mathsf{ct}_2 - (\mathsf{dk} \cdot \mathsf{ct}_1)$*.*

ElGamal PKE is perfectly correct (for $\mathsf{m} \in \{0,1\}$) and prior works have showed the following theorems.

Theorem 1 (ElGamal is MC-IND-CPA Secure [BBM00, Theorem 2]). *We can reduce the* MC-IND-CPA *security of ElGamal PKE to* DDH *with a security loss of* 2*.*

Theorem 2 (ElGamal is MC-IK-CPA Secure [MPR22a, Theorem 12]). *We can reduce the* MC-IND-CPA *security of ElGamal PKE to* DDH *with a security loss of* 2*.*

A Useful Lemma. We state the following useful lemma:

Lemma 1 (Homomorphic Properties of ElGamal Ciphertexts). *For all* $\mathsf{ek} \in \mathbb{G}$ *and* $x, y, r_x, r_y, \alpha \in \mathbb{Z}_p$*, let* $(\mathsf{ct}_0, \mathsf{ct}_1) \leftarrow \mathsf{EG.Enc}(\mathsf{ek}, x; r_x)$ *and* $(\mathsf{ct}'_0, \mathsf{ct}'_1) \leftarrow \mathsf{EG.Enc}(\mathsf{ek}, y; r_y)$*. It holds that*

(i) $(\mathsf{ct}_0 + \mathsf{ct}'_0, \mathsf{ct}_1 + \mathsf{ct}'_1) = \mathsf{EG.Enc}(\mathsf{ek}, x + y; r_x + r_y)$
(ii) $(\mathsf{ct}_0, \mathsf{ct}_1) \cdot \alpha = \mathsf{EG.Enc}(\mathsf{ek}, \alpha x; \alpha r_x)$
(iii) $x = 0 \implies \nexists r \in \mathbb{Z}_p : (\mathsf{ct}_0, \mathsf{ct}_1) \cdot \alpha = \mathsf{EG.Enc}(\mathsf{ek}, 1; r)$

Proof.

(i) can be derived from elementary group operations.
(ii) can be derived from elementary group operations.
(iii) if $x = 0$, then $(\mathsf{ct}_0, \mathsf{ct}_1) \cdot \alpha = \mathsf{EG.Enc}(\mathsf{ek}, 0; \alpha r_x)$. Because ElGamal PKE is perfectly correct, there is no $r \in \mathbb{Z}_p$ such that $\mathsf{EG.Enc}(\mathsf{ek}, 0; \alpha r_x) = \mathsf{EG.Enc}(\mathsf{ek}, 1; r)$.

□

The ZAP Proof System for Algebraic Languages. We will use the ZAP proof system for algebraic languages proposed by Couteau and Hartman [CH20a]. We recall the definition of algebraic languages and their ZAP proof system. Below are taken almost verbatim from [CH20a].

Algebraic languages. Languages that can be described by a set of algebraic equations over an Abelian group are called *algebraic languages*. The definition is as follows.

Definition 16 (Algebraic Languages [CH20a, Definition 9]). *Let $t, l, n \in \mathbb{N}, n > t$ and $\mathcal{P}_l := \left\{[a_0] + \sum_{i=1}^{l} a_i X_i\right\} \subset \mathbb{G}[\mathbf{X} = (X_1, \ldots, X_l)]$ the set of linear multivariate polynomials of degree at most 1. Let $\mathcal{D}_{par}$ be a distribution that outputs pairs $(\mathbf{M}, \mathbf{\Theta}) \in \mathcal{P}_l^{n \times t} \times \mathcal{P}_l^n$. We define the algebraic language $\mathcal{L}_{\mathbf{M},\mathbf{\Theta}} \subset \mathbb{G}^n$ as*

$$\mathcal{L}_{\mathbf{M},\mathbf{\Theta}} := \left\{\mathbf{x} \in \mathbb{G}^l | \exists \mathbf{w} \in \mathbb{Z}_p^t : \mathbf{M}(\mathbf{x}) \cdot \mathbf{w} = \mathbf{\Theta}(\mathbf{x})\right\},$$

where $\mathbf{M}(\mathbf{x})$ (resp. $\mathbf{\Theta}(\mathbf{x})$) denotes the matrix (resp. vector) received by evaluating every entry of $\mathbf{M}$ (resp. $\mathbf{\Theta}$) in the points of $\mathbf{x}$.

An Efficient ZAP Proof System for Algebraic Languages. The ZAP proof system for algebraic languages $\mathcal{L}_{\mathbf{M},\mathbf{\Theta}}$ is as follows [CH20a, Figure 3]. Let $\mathcal{PG} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ be the underlying pairing group and $\mathsf{zr} \leftarrow\!\!\$\ \mathbb{G}_2$ is a ZAP randomness.

- $\mathsf{Prove}(1^\lambda, \mathsf{zr}, \mathsf{x}, \mathsf{w})$: Parse $[\mathbf{x}]_1 := \mathsf{x} \in \mathbb{G}_1^l$ and $\mathbf{w} := \mathsf{w} \in \mathbb{Z}_p^t$. Sample $\mathbf{r} \leftarrow\!\!\$\ \mathbb{Z}_p^t$ and compute $[\mathbf{a}]_1 := [\mathbf{M}(\mathbf{x})]_1 \mathbf{r}$ and $[\mathbf{d}]_2 := \mathsf{zr} \cdot \mathbf{w} + [\mathbf{r}]_2$. Finally, output $\pi := ([\mathbf{a}]_1, [\mathbf{d}]_2)$.
- $\mathsf{Verify}(\mathsf{zr}, \mathsf{x}, \pi)$: Parse $[\mathbf{x}]_1 := \mathsf{x}$ and $([\mathbf{a}]_1, [\mathbf{d}]_2) := \pi$. Check if

$$[\mathbf{M}(\mathbf{x})]_1 \bullet [\mathbf{d}]_2 = [\mathbf{\Theta}(\mathbf{x})]_1 \bullet \mathsf{zr} + [\mathbf{a}]_1 \bullet [1]_2.$$

  If the equation holds, output 1; else output 0.

Security of the ZAP Proof System. Couteau and Hartman proved that the above ZAP proof system for algebraic languages is perfectly witness indistinguishable [CH20a, Lemma 12] and computational sound under the extended kernel matrix Diffie-Hellman $\mathcal{L}_1$-t-$\mathsf{ExtKerMDH}$ assumption (with t the size of the witness $\mathbf{w}$) [CH20a, Theorem 16].

We note that the disjunction of algebraic languages can be described as an algebraic language (see [CH20a, Section 5] and [CH20b, Appendix C] for more detail). Furthermore, it is possible to construct a ZAP proof for the disjunction of two algebraic languages if the $\mathcal{L}_1$-(t_1+t_2+1)-$\mathsf{ExtKerMDH}$ assumption holds (with t_1 and t_2 the size of the respective witnesses).

3 Recap Asymmetric Message Franking

We recall the syntax and security definitions of AMF in [TGL+19a, Section 3].

3.1 Syntax

The algorithms of AMF schemes are defined as follows.

Definition 17 (Asymmetric Message Franking). *An Asymmetric Message Franking scheme* AMF *consists of the following seven algorithms:*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*: The probabilistic key generation algorithm takes a security parameter 1^λ as input and outputs a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$. We assume that the security parameter can be inferred from $(\mathsf{pk}, \mathsf{sk})$.*
- $\mathsf{Frank}(\mathsf{sk}_s, \mathsf{pk}_r, \mathsf{pk}_j, \mathsf{m}) \to \Sigma$*: The probabilistic franking algorithm takes a sender secret key sk_s, a receiver public key pk_r, a judge public key pk_j and a message m as input and outputs a signature Σ.*
- $\mathsf{Verify}(\mathsf{pk}_s, \mathsf{sk}_r, \mathsf{pk}_j, \mathsf{m}, \Sigma) \to \{0, 1\}$*: The deterministic verification algorithm takes a receiver secret key sk_r, a sender public key pk_s, a judge public key pk_j, a message m, and a signature Σ as input and outputs 1 or 0.*

- $\mathsf{Judge}(\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{sk}_j, \mathsf{m}, \Sigma) \to \{0,1\}$*: The deterministic judge algorithm takes a judge secret key* sk_j*, a sender public key* pk_s*, a receiver public key* pk_r*, a message* m*, and a signature* Σ *as input and outputs* 1 *or* 0*.*
- $\mathsf{Forge}(\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{pk}_j, \mathsf{m}) \to \Sigma$*: The probabilistic* Forge *algorithm takes three public keys* $\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{pk}_j$ *and a message* m *as input and outputs a signature* Σ*.*
- $\mathsf{RForge}(\mathsf{pk}_s, \mathsf{sk}_r, \mathsf{pk}_j, \mathsf{m}) \to \Sigma$*: The probabilistic* RForge *algorithm takes a receiver secret key* sk_r*, a sender public key* pk_s*, a judge public key* pk_j *and a message* m *as input and outputs a signature* Σ*.*
- $\mathsf{JForge}(\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{sk}_j, \mathsf{m}) \to \Sigma$*: The probabilistic* JForge *algorithm takes a judge secret key* sk_j*, a sender public key* pk_s*, a receiver public key* pk_r *and a message* m *as input and outputs a signature* Σ*.*

We will assume that key pairs can be confirmed to be valid with a deterministic algorithm $\mathsf{WellFormed}_{\mathsf{AMF}}$ which takes a key pair $(\mathsf{pk}, \mathsf{sk})$ as input and outputs a bit b denoting whether the key pair is valid ($b = 1$) or not ($b = 0$).

AMF schemes must satisfy correctness: For any signature honestly generated by Frank, both the receiver who calls Verify and the judge who calls Judge accept the signature successfully.

Definition 18 (Correctness). *An AMF scheme* AMF *is* correct *if for any key pairs* $(\mathsf{pk}_s, \mathsf{sk}_s)$*,* $(\mathsf{pk}_r, \mathsf{sk}_r)$*, and* $(\mathsf{pk}_j, \mathsf{sk}_j)$ *honestly generated using* KeyGen*, and any message* m*, it holds that* $\mathsf{Verify}(\mathsf{pk}_s, \mathsf{sk}_r, \mathsf{pk}_j, \mathsf{m}, \Sigma) = 1$ *and* $\mathsf{Judge}(\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{sk}_j, \mathsf{m}, \Sigma) = 1$ *with overwhelming probability, where* $\Sigma \leftarrow \mathsf{Frank}(\mathsf{sk}_s, \mathsf{pk}_r, \mathsf{pk}_j, \mathsf{m})$*.*

3.2 Security Definitions

We then recall the security notions for AMF schemes. In [TGL+19b, Appendix B], Tyagi et al. discuss the trade-off between unforgeability and deniability, and define two AMF notions: AMF with strong deniability and AMF with strong authentication.[7] In this paper, we mainly focus on the construction of AMF with strong deniability [TGL+19a, Section 3].

Accountability [TGL+19a, Section 3.1]. Similar to standard signature schemes, AMF schemes must satisfy unforgeability, preventing anyone who does not know any secret keys from impersonating a sender to a receiver.

Definition 19 (Unforgeability). *We say that an AMF scheme* AMF *is* unforgeable*, if for any efficient adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{Unforge}}_{\mathsf{AMF}}(\mathcal{A}) \coloneqq \Pr\left[\mathsf{Unforge}^{\mathcal{A}}_{\mathsf{AMF}} \Rightarrow 1\right] \leq \mathsf{negl}(\lambda),$$

where the game $\mathsf{Unforge}^{\mathcal{A}}_{\mathsf{AMF}}$ *is defined on top of Figure 4.*

In addition to unforgeability, AMF scheme must satisfy other accountability properties. First, AMF schemes should prevent any sender from creating a signature that can be verified by the receiver but not the judge. This guarantees that if a receiver reports a received pair of message and signature to the judge, the judge will also acknowledge the fact that it was sent. Second, AMF schemes should prevent any receiver from reporting misbehavior of a sender by forging a signature on a message that was not sent by the sender. This prevents honest senders from being unfairly reported as malicious by a malicious receiver. These two goals are formalized as sender binding and receiver binding, respectively. Their formal definitions are as follows.

Definition 20 (Sender Binding). *We say that an AMF scheme* AMF *is* sender binding*, if for any efficient adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{s\text{-}Bind}}_{\mathsf{AMF}}(\mathcal{A}) \coloneqq \Pr\left[\mathsf{s\text{-}Bind}^{\mathcal{A}}_{\mathsf{AMF}} \Rightarrow 1\right] \leq \mathsf{negl}(\lambda),$$

where the game $\mathsf{s\text{-}Bind}^{\mathcal{A}}_{\mathsf{AMF}}$ *is defined on the right-hand side of Figure 4.*

Definition 21 (Receiver Binding). *We say that an AMF scheme* AMF *is* receiver binding*, if for any efficient adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{r\text{-}Bind}}_{\mathsf{AMF}}(\mathcal{A}) \coloneqq \Pr\left[\mathsf{r\text{-}Bind}^{\mathcal{A}}_{\mathsf{AMF}} \Rightarrow 1\right] \leq \mathsf{negl}(\lambda),$$

where the game $\mathsf{r\text{-}Bind}^{\mathcal{A}}_{\mathsf{AMF}}$ *is defined on the left-hand side of Figure 4.*

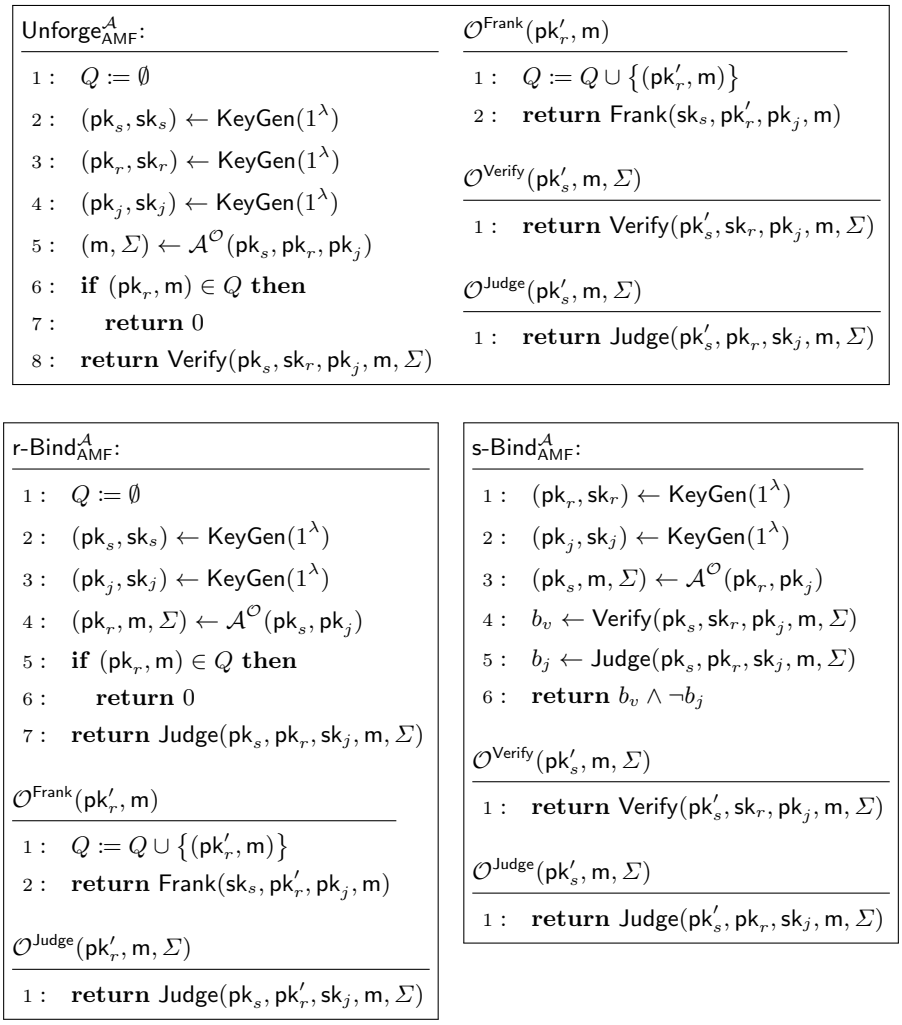


Fig. 4: Accountability games for AMF schemes: unforgeability (top), receiver binding (left) and sender binding (right). We assume that oracles have access to all public and secret keys previously defined.

Deniability [TGL+19a, Section 3.2]. To guarantee sender's privacy, AMF schemes should satisfy deniability. Tyagi et al. defined three types of deniability: universal deniability, receiver compromise deniability, and judge compromise deniability.

- Universal deniability indicates that third-parties (i.e., parties who do not know the sender's, receiver's or judge's secret key) can forge a signature, indistinguishable from honestly-generated signature against the distinguisher who only knows the sender's secret key.
- Receiver compromise deniability indicates that parties who know the receiver's secret key can forge a signature, indistinguishable from honestly-generated signature against the distinguisher who knows the sender's and the receiver's secret keys.
- Judge compromise deniability indicates that parties who know the judge's secret key can forge a signature, indistinguishable from honestly-generated signature against the distinguisher who knows the sender's, the receiver's and the judge's secret keys.

Their formal definitions are as follows.

Definition 22 (Universal Deniability). *We say that an AMF scheme* AMF *achieves* universal deniability*, if for any efficient adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{\mathsf{UnivDen}}_{\mathsf{AMF}}(\mathcal{A}) \coloneqq \left|\Pr\left[\mathsf{UnivDen}^{\mathcal{A},0}_{\mathsf{AMF}} \Rightarrow 1\right] - \Pr\left[\mathsf{UnivDen}^{\mathcal{A},1}_{\mathsf{AMF}} \Rightarrow 1\right]\right| \leq \mathsf{negl}(\lambda),$$

[7] In [TGL+19a, Section 7], Tyagi et al. used "strong deniability" to distinguish it from strong authentication.

$\mathsf{UnivDen}^{\mathcal{A},b}_{\mathsf{AMF}}$:

$(\mathsf{pk}_s, \mathsf{sk}_s) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(\mathsf{pk}_r, \mathsf{sk}_r) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Frank}}}(\mathsf{sk}_s, \mathsf{pk}_r, \mathsf{pk}_j)$
return b'

$\mathcal{O}^{\mathsf{Frank}}(\mathsf{m})$

$\Sigma_0 \leftarrow \mathsf{Frank}(\mathsf{sk}_s, \mathsf{pk}_r, \mathsf{pk}_j, \mathsf{m})$
$\Sigma_1 \leftarrow \mathsf{Forge}(\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{pk}_j, \mathsf{m})$
return Σ_b

$\mathsf{RecCompDen}^{\mathcal{A},b}_{\mathsf{AMF}}$:

$(\mathsf{pk}_s, \mathsf{sk}_s) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(\mathsf{pk}_r, \mathsf{sk}_r, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk}_s, \mathsf{pk}_j)$
$b^r_{\mathrm{wf}} \leftarrow \mathsf{WellFormed}_{\mathsf{AMF}}(\mathsf{pk}_r, \mathsf{sk}_r)$
if $b^r_{\mathrm{wf}} = 0$ **then**
  return 0
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}^{\mathsf{Frank}}}(\mathsf{sk}_s, \mathsf{sk}_r, \mathsf{pk}_j, \mathsf{st})$
return b'

$\mathcal{O}^{\mathsf{Frank}}(\mathsf{m})$

$\Sigma_0 \leftarrow \mathsf{Frank}(\mathsf{sk}_s, \mathsf{pk}_r, \mathsf{pk}_j, \mathsf{m})$
$\Sigma_1 \leftarrow \mathsf{RForge}(\mathsf{pk}_s, \mathsf{sk}_r, \mathsf{pk}_j, \mathsf{m})$
return Σ_b

$\mathsf{JdgCompDen}^{\mathcal{A},b}_{\mathsf{AMF}}$:

$(\mathsf{pk}_s, \mathsf{sk}_s) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(\mathsf{pk}_r, \mathsf{sk}_r, \mathsf{pk}_j, \mathsf{sk}_j, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk}_s)$
$b^r_{\mathrm{wf}} \leftarrow \mathsf{WellFormed}_{\mathsf{AMF}}(\mathsf{pk}_r, \mathsf{sk}_r)$
$b^j_{\mathrm{wf}} \leftarrow \mathsf{WellFormed}_{\mathsf{AMF}}(\mathsf{pk}_j, \mathsf{sk}_j)$
if $b^r_{\mathrm{wf}} \wedge b^j_{\mathrm{wf}} = 0$ **then**
  return 0
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}^{\mathsf{Frank}}}(\mathsf{sk}_s, \mathsf{sk}_r, \mathsf{sk}_j, \mathsf{st})$
return b'

$\mathcal{O}^{\mathsf{Frank}}(\mathsf{m})$

$\Sigma_0 \leftarrow \mathsf{Frank}(\mathsf{sk}_s, \mathsf{pk}_r, \mathsf{pk}_j, \mathsf{m})$
$\Sigma_1 \leftarrow \mathsf{JForge}(\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{sk}_j, \mathsf{m})$
return Σ_b

Fig. 5: Deniability security games for AMF schemes: universal deniability (top), receiver compromise deniability (left), and judge compromise deniability (right). We assume that oracles have access to all public and secret keys previously defined.

where the game $\mathsf{UnivDen}^{\mathcal{A},b}_{\mathsf{AMF}}$ *is defined on top of Figure 5.*

Definition 23 (Receiver Compromise Deniability). *We say that an AMF scheme* AMF *achieves* receiver compromise deniability*, if for any efficient adversary* $\mathcal{A}$*, it holds that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{RecCompDen}}_{\mathsf{AMF}}(\mathcal{A}) &:= \left|\Pr\left[\mathsf{RecCompDen}^{\mathcal{A},0}_{\mathsf{AMF}} \Rightarrow 1\right] - \Pr\left[\mathsf{RecCompDen}^{\mathcal{A},1}_{\mathsf{AMF}} \Rightarrow 1\right]\right| \\ &\leq \mathsf{negl}(\lambda),\end{aligned}$$

where the game $\mathsf{RecCompDen}^{\mathcal{A},b}_{\mathsf{AMF}}$ *is defined on the left-hand side of Figure 5.*

Definition 24 (Judge Compromise Deniability). *We say that an AMF scheme* AMF *achieves* judge compromise deniability*, if for any efficient adversary* $\mathcal{A}$*, it holds that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{JdgCompDen}}_{\mathsf{AMF}}(\mathcal{A}) &:= \left|\Pr\left[\mathsf{JdgCompDen}^{\mathcal{A},0}_{\mathsf{AMF}} \Rightarrow 1\right] - \Pr\left[\mathsf{JdgCompDen}^{\mathcal{A},1}_{\mathsf{AMF}} \Rightarrow 1\right]\right| \\ &\leq \mathsf{negl}(\lambda),\end{aligned}$$

where the game $\mathsf{JdgCompDen}^{\mathcal{A},b}_{\mathsf{AMF}}$ *is defined on the right-hand side of Figure 5.*

The judge compromise deniability is said to be strong because forged signatures by JForge are indistinguishable from real signatures even if all secret keys of three parties are leaked. The security requirements for AMF schemes with strong deniability [TGL+19a] are summarized as follows.

Definition 25 (AMF Schemes with Strong Deniability). *An AMF scheme* AMF *is a* secure AMF scheme with strong deniability *if* AMF *satisfies correctness, unforgeability*[8]*, sender binding,*

[8] Tyagi et al. have proved that sender binding and receiver binding imply unforgeability [TGL+19b, Theorem 4]. Therefore, it is not necessary to directly prove unforgeability.

receiver binding, universal deniability, receiver compromise deniability, and judge compromise deniability.

In the literature, other security notions than strong deniability are proposed. Tyagi et al. considered an AMF scheme with strong authentication [TGL+19b] and Huang et al. [HLZW24] considered another accountability notion called unframeability. We summarize these notions in Appendix B.

Remark 2 (Semi-Honest Key Generation). Like in [TGL+19a], both receiver and judge compromise deniability are considered in a strong semi-honest setting in the sense that the adversary is allowed to choose any AMF key pair it wants, as long as it is well-formed. This notion is stronger than in other related works [LZH+23, HLZW24] where the adversary is given (or can corrupt) honestly generated keys. Still, no deniability guarantee exists if the adversarial keys are ill-formed. The way to ensure well-formedness is not discussed in the paper and will depend on concrete instantiations. We refer to [FL25] for some interesting results related to deniability against malicious verifiers in the slightly different context of *publicly verifiable* designated verifier signatures.

4 Generic Construction of AMF in the Plain Model

4.1 Proposed Construction $\mathsf{AMF_{GC}}$

Let $\mathsf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme equipped with a key-validity algorithm $\mathsf{WellFormed_{SIG}}$ and $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme as defined in Section 2. From now on, we use Sign, Enc and Dec to denote $\mathsf{SIG.Sign}$, $\mathsf{PKE.Enc}$ and $\mathsf{PKE.Dec}$ respectively since they do not induce any ambiguity. Let $\mathsf{ZAP} = (\mathsf{Prove}, \mathsf{Verify})$ be a ZAP proof system for the NP language $\mathcal{L}$ defined by the relation $\mathcal{R}$ described in $(\star)$:

$$
\begin{aligned}
\mathcal{R} := \{\ & (\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{pk}_j, c_\sigma, c'_\sigma, c_r, c_j, m_r), (\sigma, \sigma', r_\sigma, r'_\sigma, r_r, r_j, r_{\mathsf{td}}) \quad \text{such that} && (\star)\\
& [\mathsf{td}_j = \mathsf{Enc}(\mathsf{ek}_j, 1; r_{\mathsf{td}})] && (0)\\
\vee\ & \left[\mathsf{SIG.Verify}(\mathsf{vk}_s, m_r, \sigma) \wedge c_\sigma = \mathsf{Enc}(\widetilde{\mathsf{ek}_s}, \sigma; r_\sigma) \wedge c_r = \mathsf{Enc}(\mathsf{ek}_r, 1; r_r) \wedge c_j = \mathsf{Enc}(\mathsf{ek}_j, 1; r_j)\right] && (1)\\
\vee\ & [c_r = \mathsf{Enc}(\mathsf{ek}_r, 0; r_r) \wedge c_j = \mathsf{Enc}(\mathsf{ek}_j, 0; r_j)] && (2)\\
\vee\ & \left[\mathsf{SIG.Verify}(\mathsf{vk}_r, m_r, \sigma') \wedge c'_\sigma = \mathsf{Enc}(\widetilde{\mathsf{ek}_j}, \sigma'; r'_\sigma) \wedge c_r = \mathsf{Enc}(\mathsf{ek}_r, 1; r_r) \wedge c_j = \mathsf{Enc}(\mathsf{ek}_j, 0; r_j)\right] && (3)\\
\vee\ & \left[\mathsf{SIG.Verify}(\mathsf{vk}_j, m_r, \sigma') \wedge c'_\sigma = \mathsf{Enc}(\widetilde{\mathsf{ek}_s}, \sigma'; r'_\sigma) \wedge c_r = \mathsf{Enc}(\mathsf{ek}_r, 1; r_r) \wedge c_j = \mathsf{Enc}(\mathsf{ek}_j, 1; r_j)\right]\} && (4)
\end{aligned}
$$

with $\mathsf{pk}_s = (\mathsf{vk}_s, \mathsf{ek}_s, \widetilde{\mathsf{ek}_s}, \mathsf{zr}_s, \mathsf{td}_s)$, $\mathsf{pk}_r = (\mathsf{vk}_r, \mathsf{ek}_r, \widetilde{\mathsf{ek}_r}, \mathsf{zr}_r, \mathsf{td}_r)$, $\mathsf{pk}_j = (\mathsf{vk}_j, \mathsf{ek}_j, \widetilde{\mathsf{ek}_j}, \mathsf{zr}_j, \mathsf{td}_j)$ and $m_r = m||\mathsf{pk}_r$.
The detailed construction of our AMF scheme $\mathsf{AMF_{GC}} = (\mathsf{KeyGen}, \mathsf{Frank}, \mathsf{Verify}, \mathsf{Judge}, \mathsf{Forge}, \mathsf{RForge}, \mathsf{JForge})$ is given in Figure 6. The well-formedness of keys is verified with the algorithm $\mathsf{WellFormed_{AMF_{GC}}}(\mathsf{pk}, \mathsf{sk}) := \mathsf{WellFormed_{SIG}}(\mathsf{vk}, \mathsf{sik})$.

Remark 3. A statement for the relation $\mathcal{R}$ is composed of three public keys (the identities of the parties involved in the franking), four ciphertexts and one message. Note that this notation includes zr in the statements of our ZAP proof system but does not lead to an ill-defined language since zr is a dumb variable (we decided to keep this notation for the sake of simplicity). A witness for $\mathcal{R}$ is composed of two signatures and five random values. Apart from Equation (0) which is used as trapdoor to produce valid proofs when td_j encrypts 1, every line corresponds to the output of one of the four signing algorithms: Frank (1), Forge (2), RForge (3) and JForge (4). Indeed, Conjunction (1), which corresponds to the output of an honest Frank algorithm, can be translated as follows:

> *There exists σ a valid signature of m_r for the sender verification key vk_s such that c_σ is an encryption of σ using the sender's orphan encryption key $\widetilde{\mathsf{ek}_s}$ and randomness r_σ; AND c_r (resp. c_j) is an encryption of 1 using the receiver (resp. judge) encryption key and randomness r_r (resp. r_j).*

The other conjunctions can be translated similarly.

$\mathsf{KeyGen}(1^\lambda)$:

$(\mathsf{vk}, \mathsf{sik}) \leftarrow \mathsf{SIG.KeyGen}(1^\lambda)$
$(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$
$(\widetilde{\mathsf{ek}}, _) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$
$\mathsf{zr} \leftarrow\!\!\$\ \{0,1\}^\ell$
$\mathsf{td} \leftarrow \mathsf{Enc}(\mathsf{ek}, 0)$ // Trapdoor is closed
$\mathsf{pk} \leftarrow (\mathsf{vk}, \mathsf{ek}, \widetilde{\mathsf{ek}}, \mathsf{zr}, \mathsf{td})$; $\mathsf{sk} \leftarrow (\mathsf{sik}, \mathsf{dk}, \mathsf{pk})$
return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Frank}(\mathsf{sk}_s, \mathsf{pk}_r, \mathsf{pk}_j, m)$:

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sik}_s, m||\mathsf{pk}_r)$; $\sigma' \leftarrow 0$
$c_\sigma \leftarrow \mathsf{Enc}(\widetilde{\mathsf{ek}_s}, \sigma; r_\sigma)$; $c'_\sigma \leftarrow \mathsf{Enc}(\widetilde{\mathsf{ek}_s}, \sigma'; r'_\sigma)$
$c_r \leftarrow \mathsf{Enc}(\mathsf{ek}_r, 1; r_r)$
$c_j \leftarrow \mathsf{Enc}(\mathsf{ek}_j, 1; r_j)$
$\pi \leftarrow \mathsf{ZAP.Prove}(\mathsf{zr}_j, (\mathsf{pk}_s, \mathsf{pk}_r, \cdots, m||\mathsf{pk}_r), (\sigma, \bot, r_\sigma, \bot, r_r, r_j, \bot))$
return $\Sigma \coloneqq (\pi, c_\sigma, c'_\sigma, c_r, c_j)$

$\mathsf{Verify}(\mathsf{pk}_s, \mathsf{sk}_r, \mathsf{pk}_j, m, \Sigma)$:

$b_\pi \leftarrow \mathsf{ZAP.Verify}(\mathsf{zr}_j, (\mathsf{pk}_s, \mathsf{pk}_r, \cdots, m||\mathsf{pk}_r), \pi)$
$b_r \leftarrow \mathsf{Dec}(\mathsf{dk}_r, c_r)$
return $b_\pi \wedge b_r$

$\mathsf{Judge}(\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{sk}_j, m, \Sigma)$:

$b_\pi \leftarrow \mathsf{ZAP.Verify}(\mathsf{zr}_j, (\mathsf{pk}_s, \mathsf{pk}_r, \cdots, m||\mathsf{pk}_r), \pi)$
$b_j \leftarrow \mathsf{Dec}(\mathsf{dk}_j, c_j)$
return $b_\pi \wedge b_j$

$\mathsf{Forge}(\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{pk}_j, m)$:

$\sigma \leftarrow 0$; $\sigma' \leftarrow 0$
$c_\sigma \leftarrow \mathsf{Enc}(\widetilde{\mathsf{ek}_s}, \sigma; r_\sigma)$; $c'_\sigma \leftarrow \mathsf{Enc}(\widetilde{\mathsf{ek}_s}, \sigma'; r'_\sigma)$
$c_r \leftarrow \mathsf{Enc}(\mathsf{ek}_r, 0; r_r)$
$c_j \leftarrow \mathsf{Enc}(\mathsf{ek}_j, 0; r_j)$
$\pi \leftarrow \mathsf{ZAP.Prove}(\mathsf{zr}_j, (\mathsf{pk}_s, \mathsf{pk}_r, \cdots, m||\mathsf{pk}_r), (\bot, \bot, \bot, \bot, r_r, r_j, \bot))$
return $\Sigma \coloneqq (\pi, c_\sigma, c'_\sigma, c_r, c_j)$

$\mathsf{RForge}(\mathsf{pk}_s, \mathsf{sk}_r, \mathsf{pk}_j, m)$:

$\sigma \leftarrow 0$; $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sik}_r, m||\mathsf{pk}_r)$
$c_\sigma \leftarrow \mathsf{Enc}(\widetilde{\mathsf{ek}_j}, \sigma; r_\sigma)$; $c'_\sigma \leftarrow \mathsf{Enc}(\widetilde{\mathsf{ek}_j}, \sigma'; r'_\sigma)$
$c_r \leftarrow \mathsf{Enc}(\mathsf{ek}_r, 1; r_r)$
$c_j \leftarrow \mathsf{Enc}(\mathsf{ek}_j, 0; r_j)$
$\pi \leftarrow \mathsf{ZAP.Prove}(\mathsf{zr}_j, (\mathsf{pk}_s, \mathsf{pk}_r, \cdots, m||\mathsf{pk}_r), (\bot, \sigma', \bot, r'_\sigma, r_r, r_j, \bot))$
return $\Sigma \coloneqq (\pi, c_\sigma, c'_\sigma, c_r, c_j)$

$\mathsf{JForge}(\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{sk}_j, m)$:

$\sigma \leftarrow 0$; $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sik}_j, m||\mathsf{pk}_r)$
$c_\sigma \leftarrow \mathsf{Enc}(\widetilde{\mathsf{ek}_s}, \sigma; r_\sigma)$; $c'_\sigma \leftarrow \mathsf{Enc}(\widetilde{\mathsf{ek}_s}, \sigma'; r'_\sigma)$
$c_r \leftarrow \mathsf{Enc}(\mathsf{ek}_r, 1; r_r)$
$c_j \leftarrow \mathsf{Enc}(\mathsf{ek}_j, 1; r_j)$
$\pi \leftarrow \mathsf{ZAP.Prove}(\mathsf{zr}_j, (\mathsf{pk}_s, \mathsf{pk}_r, \cdots, m||\mathsf{pk}_r), (\bot, \sigma', \bot, r'_\sigma, r_r, r_j, \bot))$
return $\Sigma \coloneqq (\pi, c_\sigma, c'_\sigma, c_r, c_j)$

Fig. 6: Algorithms of our generic construction $\mathsf{AMF}_{\mathsf{GC}}$: The encryption randomness r_σ, r'_σ, r_r and r_j are assumed to be generated honestly when needed. The symbol $\bot$ represents the universal void witness.

4.2 Security Proofs for $\mathsf{AMF}_{\mathsf{GC}}$

We now prove that our generic construction $\mathsf{AMF}_{\mathsf{GC}}$ described in Section 4.1 is a secure AMF scheme with strong deniability (cf. Definition 25).

Theorem 3 (Correctness). *If* SIG, PKE *and* ZAP *are correct then the AMF scheme* $\mathsf{AMF}_{\mathsf{GC}}$ *is correct.*

Proof. The correctness of $\mathsf{AMF}_{\mathsf{GC}}$ is a direct consequence of the correctness of SIG, PKE and ZAP. Indeed, a signature produced by an honest sender is composed of a proof π and four ciphertexts including c_r and c_j. The witness used to produce π is valid with respect to the statement, and the two aforementioned ciphertexts are genuine encryption of 1. Therefore, both Verify and Judge will accept such AMF signature with overwhelming probability. □

Accountability Notions. To prove receiver biding and sender binding, we mainly resort to the EUF-CMA security of SIG and the adaptive soundness of ZAP. In both cases, we suppose that an adversary $\mathcal{A}$ has some advantage in breaking one of these security notions for $\mathsf{AMF}_{\mathsf{GC}}$ and then use it to produce a forgery against the EUF-CMA security of SIG.

Theorem 4 (Receiver Binding). *If* SIG *is* EUF-CMA *secure, if* PKE *is perfectly correct and if* ZAP *is adaptively sound then the AMF scheme* $\mathsf{AMF}_{\mathsf{GC}}$ *is receiver binding. Furthermore, if* $\mathcal{A}$ *is*

an adversary against the receiver binding security of $\mathsf{AMF_{GC}}$*, then there exist adversaries* $\mathcal{B}$ *and* $\mathcal{B}'$ *against the* EUF-CMA *security of* SIG *and the adaptive soundness of* ZAP *such that*

$$\mathsf{Adv}^{\mathsf{r\text{-}Bind}}_{\mathsf{AMF_{GC}}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{Sound}}_{\mathsf{ZAP}}(\mathcal{B}').$$

Note that the running time of $\mathcal{B}$ *and* $\mathcal{B}'$ *is about that of* $\mathcal{A}$*.*

Proof. Let $\mathcal{A}$ be an adversary against the receiver binding security of $\mathsf{AMF_{GC}}$. The goal of this proof is to construct an adversary $\mathcal{B}$ against the EUF-CMA security of SIG which can transform a forgery for $\mathsf{AMF_{GC}}$ into a forgery for SIG. However, recall that a signature produced by Frank is somehow indistinguishable from a signature produced by JForge (judge compromise deniability). Therefore, we have to construct two adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ which produce valid forgeries for the sender verification key or judge verification key respectively, and then run one of them hoping to choose the right one.

Suppose without loss of generality that $\mathcal{A}$ never outputs a forbidden forgery, i.e., such that (pk^*_r, m^*) has already been queried. Let game $\mathsf{G_0}$ be the original $\mathsf{r\text{-}Bind}^{\mathcal{A}}_{\mathsf{AMF_{GC}}}$ game.

Game $\mathsf{G_1}$. Let game $\mathsf{G_1}$ be the $\mathsf{EUF\text{-}CMA}^{\mathcal{B}}_{\mathsf{SIG}}$ game (with $\mathcal{B}$ the adversary described below against the EUF-CMA security of SIG and using $\mathcal{A}$ as a black box), but with an easy winning condition: $\mathcal{B}$ wins game $\mathsf{G_1}$ if $\mathsf{EUF\text{-}CMA}^{\mathcal{B}}_{\mathsf{SIG}}$ does not abort and if $\mathcal{B}$ produces a forgery attempt (independently of the game output). The following paragraphs describe the EUF-CMA adversary $\mathcal{B}_1$ (and its variant $\mathcal{B}_2$) and we define $\mathcal{B}$ as the EUF-CMA adversary which runs $\mathcal{B}_1$ or $\mathcal{B}_2$ with the same probability 1/2.

Setup. When receiving a verification key vk from the EUF-CMA challenger, $\mathcal{B}_1$ generates an incomplete *sender* key pair $\left(\mathsf{pk}_s, \overline{\mathsf{sk}_s}\right)$ for $\mathsf{AMF_{GC}}$ by running twice the PKE.KeyGen algorithm and storing both encryption *and* decryption keys $(\mathsf{ek}_s, \mathsf{dk}_s)$ and also $(\widetilde{\mathsf{ek}_s}, \widetilde{\mathsf{dk}_s})$. $\mathcal{B}_1$ then sets $\overline{\mathsf{sk}_s} \coloneqq (_, \mathsf{dk}_s, \mathsf{pk}_s)$. It also sets the associated public key as $\mathsf{pk}_s \coloneqq (\mathsf{vk}, \mathsf{ek}_s, \widetilde{\mathsf{ek}_s}, \mathsf{zr}_s, \mathsf{td}_s)$ with honestly generated zr_s and td_s. Finally, it generates an authentic AMF judge key pair $(\mathsf{pk}_j, \mathsf{sk}_j)$ using $\mathsf{AMF_{GC}}$.KeyGen and sends the two public keys $(\mathsf{pk}_s, \mathsf{pk}_j)$ to $\mathcal{A}$.

Frank Queries. When receiving a Frank query from $\mathcal{A}$ for message m and receiver pk_r, $\mathcal{B}_1$ starts by computing the receiver-dependent message $m_r \coloneqq m||\mathsf{pk}_r$. It then sends a Sign query to the EUF-CMA challenger for message m_r and stores the answer in the variables σ and σ'. Finally, it computes the AMF signature Σ for m the same way as in the authentic Frank algorithm since it already knows σ and σ'. Remark that the missing signing key sik_s is not necessary to produce the ZAP proof.

Judge Queries. $\mathcal{B}_1$ can answer Judge queries on its own since it has an authentic judge secret key.

Forgery. When receiving a forgery $(\mathsf{pk}^*_r, m^*, \Sigma^*)$ for $\mathsf{AMF_{GC}}$ with $\Sigma^* = (\pi^*, c^*_\sigma, {c'_\sigma}^*, \cdots)$, $\mathcal{B}_1$ first checks its validity using the Judge algorithm and aborts if the latter rejects Σ. Then, $\mathcal{B}_1$ sets σ^* as the decryption of c^*_σ using $\widetilde{\mathsf{dk}_s}$ and sends (σ^*, m^*_r) to the EUF-CMA challenger.

$\mathcal{B}_2$ ***Variant.*** Let $\mathcal{B}_2$ be the adversary which behaves as $\mathcal{B}_1$ except that it swaps the roles of the sender and judge key pairs in the setup phase: the verification key received from the EUF-CMA challenger is inserted in the judge key pair (which thus becomes incomplete). However, $\widetilde{\mathsf{dk}_s}$ is still generated for the *sender* key pair. In the query phase, $\mathcal{B}_2$ can answer all queries himself, without resorting to the Sign oracle because the missing judge signing key is not necessary to perform Frank or Judge. Finally, $\mathcal{B}_2$ sets ${\sigma'}^*$ as the decryption of ${c'_\sigma}^*$ (instead of c^*_σ) still using $\widetilde{\mathsf{dk}_s}$ before sending $({\sigma'}^*, m^*_r)$ to the EUF-CMA challenger.

First, remark that all public keys and answers to $\mathcal{A}$'s queries sent by $\mathcal{B}_1$ or $\mathcal{B}_2$ are genuine (sometimes with the help of a Sign oracle) and therefore indistinguishable from one another or from an interaction with the real r-Bind game. Then, notice that if a forgery Σ^* produced by $\mathcal{A}$ results in winning game $\mathsf{G_0}$ then this forgery was accepted by Judge and therefore would not make $\mathcal{B}_1$ nor $\mathcal{B}_2$ abort. Since SIG.Verify never aborts, $\mathsf{EUF\text{-}CMA}^{\mathcal{B}}_{\mathsf{SIG}}$ would not abort in the same context. From this, we deduce that

$$\Pr[\mathsf{G_1} \Rightarrow 1] \geq \Pr[\mathsf{G_0} \Rightarrow 1]. \tag{5}$$

Game $\mathsf{G_2}$. Let game $\mathsf{G_2}$ be similar to game $\mathsf{G_1}$ but with the following change in the winning condition: $\mathcal{B}$ is successful if $\mathsf{EUF\text{-}CMA}^{\mathcal{B}}_{\mathsf{SIG}}$ does not abort, if $\mathcal{B}$ produces a forgery attempt $\Sigma^* = (\pi^*, c^*_\sigma, \cdots, {c_j}^*)$, public key pk^*_r and message m^* *and if the statement* $x^* = (\mathsf{pk}_s, \mathsf{pk}^*_r, \mathsf{pk}_j, c^*_\sigma, \cdots, m^*||\mathsf{pk}^*_r)$ *is in the language* $\mathcal{L}$ *defined by* $\mathcal{R}$ *in* $(\star)$. A reduction to the adaptive soundness of ZAP yields

$$0 \leq \Pr[\mathsf{G_1} \Rightarrow 1] - \Pr[\mathsf{G_2} \Rightarrow 1] \leq \mathsf{Adv}^{\mathsf{Sound}}_{\mathsf{ZAP}}(\mathcal{B}') \tag{6}$$

for a suitable ZAP soundness adversary $\mathcal{B}'$. This adversary would insert its zr input into the judge public key (pk_j follows the same distribution as in the original game) and would, if $\mathsf{EUF\text{-}CMA}^{\mathcal{B}}_{\mathsf{SIG}}$ terminates, output the ZAP statement and proof (x^*, π^*) associated with $\mathcal{A}$'s forgery.

Game G_3. Let game G_3 be the *original* $\mathsf{EUF\text{-}CMA}^{\mathcal{B}}_{\mathsf{SIG}}$ game. If $\mathcal{B}$ wins G_2, then its forgery Σ^* has been accepted by Judge (thus c^*_j decrypts to 1) and $x^* = (\mathsf{pk}_s, \mathsf{pk}^*_r, \mathsf{pk}_j, c^*_\sigma, \cdots, m^*||\mathsf{pk}^*_r) \in \mathcal{L}$ with the same notations as before. Because td_j encrypts 0 and because of perfect correctness of PKE, Equation (0) is false and therefore one of the other four conjunctions of $(\star)$ is true for some witness w^*. Conjunctions (2) and (3) are false because of the decryption of c^*_j and perfect correctness of PKE, so (1) or (4) must be true. If (1) is true then $\mathsf{SIG.Verify}(\mathsf{vk}_s, m^*_r, \sigma^*) = 1$. Since the tuple (pk^*_r, m^*) has never been queried to $\mathcal{B}$ (assumption at the beginning of the proof), $m^*_r = m^*||\mathsf{pk}^*_r$ has never been queried to the EUF-CMA challenger, so $\mathcal{B}_1$ is successful in the EUF-CMA forgery game. Similarly, if (4) is true then $\mathsf{SIG.Verify}(\mathsf{vk}_j, m^*_r, \sigma'^*) = 1$ so $\mathcal{B}_2$ is successful in the EUF-CMA forgery game. Therefore, the probability that $\mathcal{B}$ runs a successful adversary is at least $1/2$:

$$\Pr[\mathsf{G}_3 \Rightarrow 1] \geq \frac{1}{2} \cdot \Pr[\mathsf{G}_2 \Rightarrow 1]. \tag{7}$$

Combining Equations (5) to (7) yields the result stated in Theorem 4. □

Theorem 5 (Sender Binding). *If* SIG *is* EUF-CMA *secure, if* PKE *is perfectly correct and if* ZAP *is adaptively sound then the AMF scheme* $\mathsf{AMF}_{\mathsf{GC}}$ *is sender binding. Furthermore, if* $\mathcal{A}$ *is an adversary against the sender binding security of* $\mathsf{AMF}_{\mathsf{GC}}$*, then there exist adversaries* $\mathcal{C}$ *and* $\mathcal{C}'$ *against the* EUF-CMA *security of* SIG *and the adaptive soundness of* ZAP *such that*

$$\mathsf{Adv}^{\mathsf{s\text{-}Bind}}_{\mathsf{AMF}_{\mathsf{GC}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{C}) + \mathsf{Adv}^{\mathsf{Sound}}_{\mathsf{ZAP}}(\mathcal{C}').$$

Note that the running time of $\mathcal{C}$ *and* $\mathcal{C}'$ *is about that of* $\mathcal{A}$*.*

Proof. The proof of Theorem 5 is quite similar to the proof of Theorem 4 so we defer it to Appendix A.1. □

Deniability notions. To prove universal deniability, receiver compromise deniability and judge compromise deniability, we use a game-based approach. Starting from the interaction between an adversary $\mathcal{A}$ and a challenger for one of the three deniability games, we will slightly change the behavior of this challenger step by step in order to reach one of the three deniability algorithms.

Theorem 6 (Universal Deniability). *If* SIG *is correct, if* PKE *is* MC-IND-CPA *secure and if* ZAP *is statistically witness indistinguishable then the AMF scheme* $\mathsf{AMF}_{\mathsf{GC}}$ *achieves universal deniability. Furthermore, if* $\mathcal{A}$ *is an adversary against the universal deniability security of* $\mathsf{AMF}_{\mathsf{GC}}$*, then there exists an adversary* $\mathcal{D}$ *against the* MC-IND-CPA *security of* PKE *such that*

$$\mathsf{Adv}^{\mathsf{UnivDen}}_{\mathsf{AMF}_{\mathsf{GC}}}(\mathcal{A}) \leq 5 \cdot \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{D}) + 2Q \cdot \varepsilon_{\mathsf{WI}} + Q \cdot \varepsilon_{\mathsf{SIG}}$$

with Q *the number of* Frank *queries of* $\mathcal{A}$ *and* $\varepsilon_{\mathsf{WI}}$ *(resp.* $\varepsilon_{\mathsf{SIG}}$*) related to the statistical witness-indistinguishability of* ZAP *(resp. correctness error of* SIG*). Note that the running time of* $\mathcal{D}$ *is about that of* $\mathcal{A}$*.*

Proof. In the universal deniability game, an adversary $\mathcal{A}$ is asked to distinguish between an oracle that outputs signatures generated by Frank or Forge. Figure 7 describes the behavior of this oracle between game G_1 and game G_6 (the first and last game hops are not represented since the updates occur in the key generation phase). We justify and evaluate the hops between different games in the following paragraphs.

Game G_0. This is the genuine $\mathsf{UnivDen}^{\mathcal{A},0}_{\mathsf{AMF}_{\mathsf{GC}}}$ security game: $\mathcal{O}^{\mathsf{Frank}}$ uses the real Frank algorithm to answer queries.

Game G_1. In game G_1, the judge key pair is generated with the trapdoor open: instead of encrypting 0 in td_j, the challenger encrypts 1 using randomness. A reduction to the MC-IND-CPA security of PKE yields

$$|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_0 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{D}_1)$$

for a suitable MC-IND-CPA adversary $\mathcal{D}_1$.

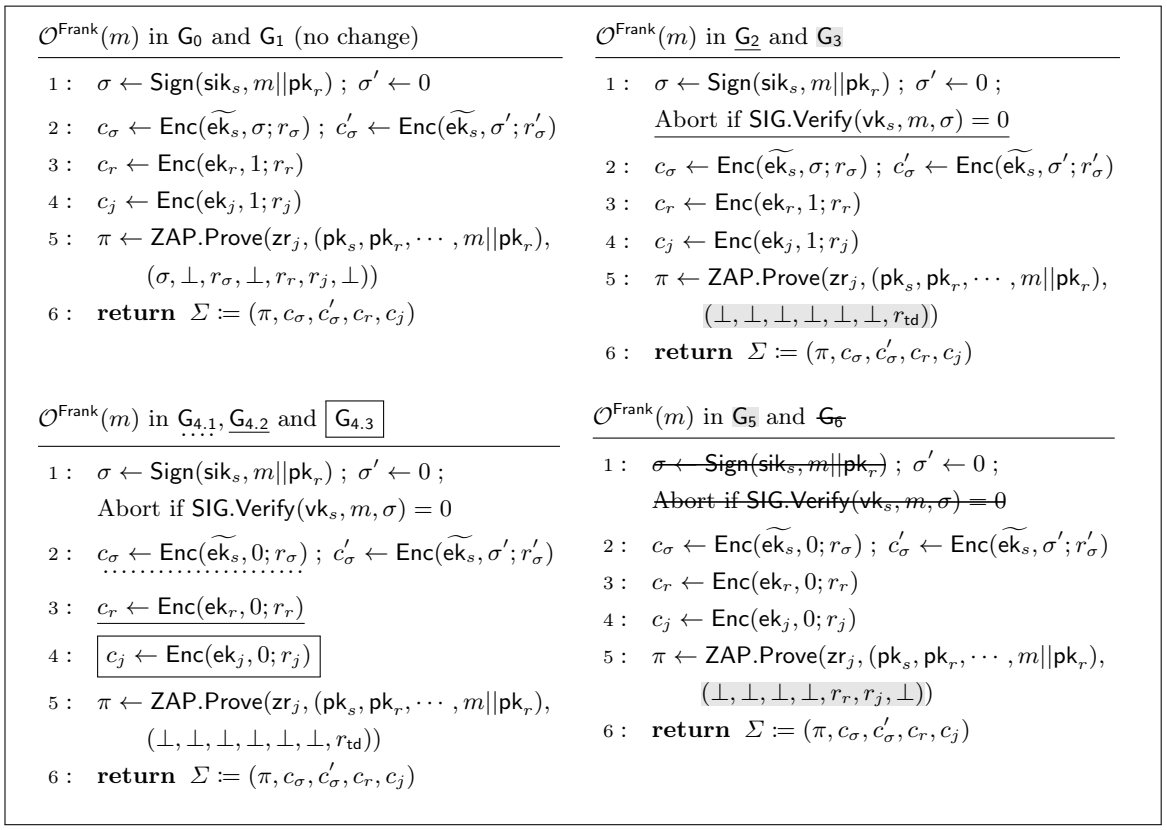


Fig. 7: Changes in the behavior of $\mathcal{O}^{\mathsf{Frank}}$ during the universal deniability proof. The first and last game hops are not represented since the updates occur in the key generation phase.

Game G_2. In game G_2, $\mathcal{O}^{\mathsf{Frank}}$ aborts as soon as it produces an invalid signature for verification key vk_s. This change is purely conceptual since in this case, the oracle would have aborted when trying to generate the ZAP proof π.

$$\Pr[\mathsf{G}_2 \Rightarrow 1] = \Pr[\mathsf{G}_1 \Rightarrow 1]$$

Game G_3. In game G_3, $\mathcal{O}^{\mathsf{Frank}}$ switches the witness of the ZAP proof to use the randomness r_{td} (generated in Game G_1) associated with the trapdoor ciphertext td_j. Both witnesses are correct for the statement with respect to $\mathcal{R}$ so a reduction to the statistical witness-indistinguishability of ZAP yields:

$$|\Pr[\mathsf{G}_3 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1]| \leq Q \cdot \varepsilon_{\mathsf{WI}}$$

with Q the number of Frank queries of $\mathcal{A}$ and $\varepsilon_{\mathsf{WI}}$ related to the statistical witness-indistinguishability of ZAP.

Games $\mathsf{G}_{4.1}$, $\mathsf{G}_{4.2}$, $\mathsf{G}_{4.3}$. In game $\mathsf{G}_{4.1}$, $\mathcal{O}^{\mathsf{Frank}}$ changes the value of the plaintext inside c_σ while still producing a signature with sik_s and aborting in case it is invalid. A reduction to the MC-IND-CPA security of PKE yields

$$|\Pr[\mathsf{G}_{4.1} \Rightarrow 1] - \Pr[\mathsf{G}_3 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{D}_{4.1})$$

for a suitable MC-IND-CPA adversary $\mathcal{D}_{4.1}$. The hops between games $\mathsf{G}_{4.1}$ and $\mathsf{G}_{4.2}$, and between $\mathsf{G}_{4.2}$ and $\mathsf{G}_{4.3}$ are similar thus give similar reductions with suitable adversaries $\mathcal{D}_{4.2}$ and $\mathcal{D}_{4.3}$ respectively.

$$|\Pr[\mathsf{G}_{4.2} \Rightarrow 1] - \Pr[\mathsf{G}_{4.1} \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{D}_{4.2})$$

$$|\Pr[\mathsf{G}_{4.3} \Rightarrow 1] - \Pr[\mathsf{G}_{4.2} \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{D}_{4.3})$$

Game G_5. In game G_5, $\mathcal{O}^{\mathsf{Frank}}$ switches again the witness of the ZAP proof to use the same witness as the Forge algorithm. A reduction to the statistical witness-indistinguishability of ZAP yields:

$$|\Pr[\mathsf{G}_3 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1]| \leq Q \cdot \varepsilon_{\mathsf{WI}}$$

with Q the number of Frank queries of $\mathcal{A}$ and $\varepsilon_{\mathsf{WI}}$ related to the statistical witness-indistinguishability of ZAP.

Game G_6. In game G_6, $\mathcal{O}^{\mathsf{Frank}}$ stops producing σ and checking its validity. The output of the game changes only when an invalid signature was produced in G_5, which happens with negligible probability since SIG is correct:

$$|\Pr[\mathsf{G}_6 \Rightarrow 1] - \Pr[\mathsf{G}_5 \Rightarrow 1]| \leq Q \cdot \varepsilon_{\mathsf{SIG}}$$

with Q the number of Frank queries and $\varepsilon_{\mathsf{SIG}}$ the correctness error of SIG.

Game G_7. In game G_7, the judge key pair is generated with td_j encrypting 0 again (trapdoor closed). G_7 perfectly describes $\mathsf{UnivDen}^{\mathcal{A},1}_{\mathsf{AMF}_{\mathsf{GC}}}$. A reduction to the $\mathsf{MC\text{-}IND\text{-}CPA}$ security of PKE yields

$$|\Pr[\mathsf{G}_7 \Rightarrow 1] - \Pr[\mathsf{G}_6 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{D}_7)$$

for a suitable $\mathsf{MC\text{-}IND\text{-}CPA}$ adversary $\mathcal{D}_7$.

Collecting all differences of probabilities yields the result stated in Theorem 6. □

Theorem 7 (Receiver Compromise Deniability). *If* SIG *is correct, if* PKE *is* $\mathsf{MC\text{-}IND\text{-}CPA}$ *and* $\mathsf{MC\text{-}IK\text{-}CPA}$ *secure, and if* ZAP *is statistically witness indistinguishable then the AMF scheme* $\mathsf{AMF}_{\mathsf{GC}}$ *achieves receiver compromise deniability. Furthermore, if* $\mathcal{A}$ *is an adversary against the receiver compromise deniability security of* $\mathsf{AMF}_{\mathsf{GC}}$*, then there exist adversaries* $\mathcal{E}$ *and* $\mathcal{E}'$ *against the* $\mathsf{MC\text{-}IND\text{-}CPA}$ *and* $\mathsf{MC\text{-}IK\text{-}CPA}$ *security of* PKE *such that*

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{RecCompDen}}_{\mathsf{AMF}_{\mathsf{GC}}}(\mathcal{A})\\ &\quad\leq 5 \cdot \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{E}) + \mathsf{Adv}^{\mathsf{MC\text{-}IK\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{E}') + 2Q \cdot \varepsilon_{\mathsf{WI}} + 2Q \cdot \varepsilon_{\mathsf{SIG}}\end{aligned}$$

with Q *the number of* Frank *queries of* $\mathcal{A}$ *and* $\varepsilon_{\mathsf{WI}}$ *(resp.* $\varepsilon_{\mathsf{SIG}}$*) related to the statistical witness-indistinguishability of* ZAP *(resp. correctness error of* SIG*). Note that the running time of* $\mathcal{E}$ *and* $\mathcal{E}'$ *is about that of* $\mathcal{A}$*.*

Proof. The proof of Theorem 7 is very similar to the proof of Theorem 6 so we defer it to Appendix A.2. We note that $\mathsf{MC\text{-}IK\text{-}CPA}$ security of PKE hides which of $\widetilde{\mathsf{ek}_s}$ or $\widetilde{\mathsf{ek}_j}$ is used to generate c_σ. □

Theorem 8 (Judge Compromise Deniability). *If* SIG *is correct, if* PKE *is* $\mathsf{MC\text{-}IND\text{-}CPA}$ *secure and if* ZAP *is statistically witness indistinguishable then the AMF scheme* $\mathsf{AMF}_{\mathsf{GC}}$ *achieves judge compromise deniability. Furthermore, if* $\mathcal{A}$ *is an adversary against the judge compromise deniability security of* $\mathsf{AMF}_{\mathsf{GC}}$*, there exists an adversary* $\mathcal{F}$ *against the* $\mathsf{MC\text{-}IND\text{-}CPA}$ *security of* PKE *such that*

$$\mathsf{Adv}^{\mathsf{JdgCompDen}}_{\mathsf{AMF}_{\mathsf{GC}}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{F}) + Q \cdot \varepsilon_{\mathsf{WI}} + 2Q \cdot \varepsilon_{\mathsf{SIG}}$$

with Q *the number of* Frank *queries of* $\mathcal{A}$ *and* $\varepsilon_{\mathsf{WI}}$ *(resp.* $\varepsilon_{\mathsf{SIG}}$*) related to the statistical witness-indistinguishability of* ZAP *(resp. correctness error of* SIG*). Note that the running time of* $\mathcal{F}$ *is about that of* $\mathcal{A}$*.*

Proof. The proof of Theorem 8 is very similar to the proof of Theorem 6 so we defer it to Appendix A.3. □

From Theorems 3 to 8, we conclude the following corollary.

Corollary 1. *If* PKE *is perfectly correct,* $\mathsf{MC\text{-}IND\text{-}CPA}$ *secure and* $\mathsf{MC\text{-}IK\text{-}CPA}$ *secure, if* SIG *is correct and* $\mathsf{EUF\text{-}CMA}$ *secure, and if* ZAP *is correct, witness indistinguishable and adaptively sound, then the AMF scheme* $\mathsf{AMF}_{\mathsf{GC}}$ *is a secure AMF scheme with strong deniability.*

Remark 4 (Construction of AMF with Unframeability). By tweaking our generic construction $\mathsf{AMF}_{\mathsf{GC}}$, an AMF scheme with unframeability can be obtained. The details are provided in Appendix C.

4.3 Instantiations of Generic Construction

Our generic construction $\mathsf{AMF}_{\mathsf{GC}}$ can be instantiated from concrete assumptions in the plain model. Especially, we obtain

- the first tightly secure AMF scheme from standard group-based assumptions (with pairings) by using ElGamal PKE scheme based on DDH assumption [ElG85], a standard signature scheme based on DLIN assumption of Hofheinz and Jager [HJ12], and a statistical ZAP for NP based on DDH and Kernel DH assumptions of Couteau, Katsumata, Sadeghi, and Ursu [CKSU21], and
- the first post-quantum secure AMF scheme based on LWE assumption by using the perfectly correct version of Regev PKE scheme described in [CGH+21, Section 2.3], a standard signature scheme based on LWE assumption [CHKP10, DM14], and a statistical ZAP for NP based on sub-exponentially-hard LWE assumption [LVW19, BFJ+20, GJJM20].

Remark 5. The post-quantum scheme requires sub-exponential hardness of LWE problem. To remove this stronger assumption, in Section 6, we show how to replace ZAP for NP with ZAP for $\mathsf{NP} \cap \mathsf{coNP}$ which can be constructed from polynomially-hard LWE assumption [CGH+21].

5 Efficient Construction of AMF from Pairings

We give here an efficient construction of an asymmetric message franking scheme $\mathsf{AMF}_{\mathsf{EC}}$ with the message space $\mathcal{M} := \{0,1\}^*$ using:

- The ZAP proof system of Couteau and Hartman [CH20a] for algebraic languages which works over pairing groups $\mathcal{PG} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ with $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order p prime and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a bilinear map,
- The ZAP version of Waters-like signature scheme inspired from [DH26],
- The scalar variant[9] of ElGamal PKE $(\mathsf{EG.KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ over $\mathbb{G}_1$, and
- A collision resistant hash function $\mathsf{H} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^k$ with $k = k(\lambda) \in \mathbb{N}$. We think that it is irrelevant to pick a hash function at this point, since it has a rather external role and could be instantiated later.

In this section we use the additive notation for group operations, and component-wise multiplication over tuples of group elements. Furthermore, the summation over $h[i] = 1$ (for some finite bit-string h) denotes the sum over all i such that the i-th bit of h is equal to 1.

Recall that the main challenge of this section is to circumvent the impossibility to design unforgeable signatures for which the verification algorithm solely consists of checking linear equations [DHH+21], while using an efficient proof system which can handle little more than disjunction of the latter. Note that contrary to the generic construction, there is no concrete signature produced, the proof (here ZAP proof) *is* the signature.

5.1 Proposed Construction $\mathsf{AMF}_{\mathsf{EC}}$

The detailed construction of $\mathsf{AMF}_{\mathsf{EC}} = (\mathsf{KeyGen}, \mathsf{Frank}, \mathsf{Verify}, \mathsf{Judge}, \mathsf{Forge}, \mathsf{RForge}, \mathsf{JForge})$ is given in Figure 8. The signing part of our construction is inspired by Waters signature [Wat05] and more specifically a witness indistinguishability version of it [DH26]. The well-formedness of keys is verified by checking if $C^{(0)} = \mathsf{Enc}(\mathsf{ek}, 1; t^{(0)})$ and $C^{(i)} = \mathsf{Enc}(\mathsf{ek}, 0; t^{(i)})$ for all $i \in [1, k]$ hold. The language $\mathcal{L}'$ for the ZAP proof system is defined by the following relation:

[9] Plaintexts are scalars and only a few of them can be decrypted (here $\{0,1\}$).

$$\mathcal{R}' \coloneqq \{ \underbrace{\mathsf{ek}_s, \mathsf{ek}_r, \mathsf{ek}_j, \mathsf{td}_j, (C_s^{(i)}, C_r^{(i)}, C_j^{(i)})_{0\le i\le k}}_{\text{can be derived from } (\mathsf{pk}_s,\mathsf{pk}_r,\mathsf{pk}_j)}, c_r, c_j, h), (\alpha, t, r_r, r_j, r_{\mathsf{td}}) \text{ such that} \quad (*)$$

$$[\mathsf{td}_j = \mathsf{Enc}(\mathsf{ek}_j, 1; r_{\mathsf{td}})] \quad (0')$$

$$\vee \left[C_s^{(h)} \cdot \alpha = \mathsf{Enc}(\mathsf{ek}_s, 1; t) \wedge c_r = \mathsf{Enc}(\mathsf{ek}_r, 1; r_r) \wedge c_j = \mathsf{Enc}(\mathsf{ek}_j, 1; r_j)\right] \quad (1')$$

$$\vee \quad [c_r = \mathsf{Enc}(\mathsf{ek}_r, 0; r_r) \wedge c_j = \mathsf{Enc}(\mathsf{ek}_j, 0; r_j)] \quad (2')$$

$$\vee \left[C_r^{(h)} \cdot \alpha = \mathsf{Enc}(\mathsf{ek}_r, 1; t) \wedge c_r = \mathsf{Enc}(\mathsf{ek}_r, 1; r_r) \wedge c_j = \mathsf{Enc}(\mathsf{ek}_j, 0; r_j)\right] \quad (3')$$

$$\vee \left[C_j^{(h)} \cdot \alpha = \mathsf{Enc}(\mathsf{ek}_j, 1; t) \wedge c_r = \mathsf{Enc}(\mathsf{ek}_r, 1; r_r) \wedge c_j = \mathsf{Enc}(\mathsf{ek}_j, 1; r_j)\right]\} \quad (4')$$

with $C_s^{(h)} \coloneqq C_s^{(0)} + \sum_{h[i]=1} C_s^{(i)}$ and $C_r^{(h)} \coloneqq C_r^{(0)} + \sum_{h[i]=1} C_r^{(i)}$ and $C_r^{(h)} \coloneqq C_j^{(0)} + \sum_{h[i]=1} C_j^{(i)}$.

$\mathcal{L}'$ is an algebraic language in the sense of Definition 16 (Section 2.6) since it is a disjunction of the five algebraic languages

$$\begin{aligned}
\mathcal{L}'_0 &\coloneqq \{\mathbf{x} = (\mathsf{ek}_s, \mathsf{ek}_r, \cdots, c_j, h) \mid \exists \mathbf{w} = (r_{\mathsf{td}}) \ : \mathbf{M}_0(\mathbf{x}) \cdot \mathbf{w} = \mathbf{\Theta}_0(\mathbf{x})\} \\
\mathcal{L}'_1 &\coloneqq \{\mathbf{x} = (\mathsf{ek}_s, \mathsf{ek}_r, \cdots, c_j, h) \mid \exists \mathbf{w} = (\alpha, t, r_r, r_j) \ : \mathbf{M}_1(\mathbf{x}) \cdot \mathbf{w} = \mathbf{\Theta}_1(\mathbf{x})\} \\
\mathcal{L}'_2 &\coloneqq \{\mathbf{x} = (\mathsf{ek}_s, \mathsf{ek}_r, \cdots, c_j, h) \mid \exists \mathbf{w} = (r_r, r_j) \ : \mathbf{M}_2(\mathbf{x}) \cdot \mathbf{w} = \mathbf{\Theta}_2(\mathbf{x})\} \\
\mathcal{L}'_3 &\coloneqq \{\mathbf{x} = (\mathsf{ek}_s, \mathsf{ek}_r, \cdots, c_j, h) \mid \exists \mathbf{w} = (\alpha, t, r_r, r_j) \ : \mathbf{M}_3(\mathbf{x}) \cdot \mathbf{w} = \mathbf{\Theta}_3(\mathbf{x})\} \\
\mathcal{L}'_4 &\coloneqq \{\mathbf{x} = (\mathsf{ek}_s, \mathsf{ek}_r, \cdots, c_j, h) \mid \exists \mathbf{w} = (\alpha, t, r_r, r_j) \ : \mathbf{M}_4(\mathbf{x}) \cdot \mathbf{w} = \mathbf{\Theta}_4(\mathbf{x})\} \text{ with}
\end{aligned}$$

$$\mathbf{M}_0(\mathbf{x}) = \begin{pmatrix} g_1 \\ \mathsf{ek}_j \end{pmatrix} \quad \mathbf{\Theta}_0(\mathbf{x}) = \begin{pmatrix} \mathsf{td}[0] \\ \mathsf{td}[1] - g_1 \end{pmatrix} \quad \mathbf{M}_1(\mathbf{x}) = \begin{pmatrix} C_s^{(h)}[0] & -g_1 & \cdot & \cdot \\ C_s^{(h)}[1] & -\mathsf{ek}_s & \cdot & \cdot \\ \cdot & \cdot & g_1 & \cdot \\ \cdot & \cdot & \mathsf{ek}_r & \cdot \\ \cdot & \cdot & \cdot & g_1 \\ \cdot & \cdot & \cdot & \mathsf{ek}_j \end{pmatrix} \quad \mathbf{\Theta}_1(\mathbf{x}) = \begin{pmatrix} 0 \\ g_1 \\ c_r[0] \\ c_r[1] - g_1 \\ c_j[0] \\ c_j[1] - g_1 \end{pmatrix}$$

$$\mathbf{M}_2(\mathbf{x}) = \begin{pmatrix} g_1 & \cdot \\ \mathsf{ek}_r & \cdot \\ \cdot & g_1 \\ \cdot & \mathsf{ek}_j \end{pmatrix} \quad \mathbf{\Theta}_2(\mathbf{x}) = \begin{pmatrix} c_r[0] \\ c_r[1] \\ c_j[0] \\ c_j[1] \end{pmatrix} \quad \mathbf{\Theta}_3(\mathbf{x}) = \begin{pmatrix} 0 \\ g_1 \\ c_r[0] \\ c_r[1] - g_1 \\ c_j[0] \\ c_j[1] \end{pmatrix} \quad \mathbf{\Theta}_4(\mathbf{x}) = \begin{pmatrix} 0 \\ g_1 \\ c_r[0] \\ c_r[1] - g_1 \\ c_j[0] \\ c_j[1] - g_1 \end{pmatrix} \ ;$$

$\mathbf{M}_3(\mathbf{x})$ (resp. $\mathbf{M}_4(\mathbf{x})$) is identical to $\mathbf{M}_1(\mathbf{x})$ with $C_s^{(h)}$ substituted by $C_r^{(h)}$ (resp. $C_j^{(h)}$) and ek_s substituted by ek_r (resp. ek_j).

Efficiency of $\mathsf{AMF}_{\mathsf{EC}}$. The public key of $\mathsf{AMF}_{\mathsf{EC}}$ has $2k+5$ elements in $\mathbb{G}_1$, one in $\mathbb{G}_2$, plus the hash key. A signature contains 2 ElGamal ciphertexts (4 elements in $\mathbb{G}_1$) plus the ZAP proof. A ZAP proof for $\mathcal{L}'_0$ (resp. $\mathcal{L}'_1$, $\mathcal{L}'_2$, $\mathcal{L}'_3$ and $\mathcal{L}'_4$) is composed of 2 elements in $\mathbb{G}_1$ plus 1 in $\mathbb{G}_2$ (resp. 6+4, 4+2, 6+4 and 6+4). Finally, to factor the disjunction in, 4 more elements of $\mathbb{G}_2$ are needed, which yields a signature size of 47 group elements (28 in $\mathbb{G}_1$ and 19 in $\mathbb{G}_2$). This is a significant improvement compared to the naive instantiation of the generic construction using the NIWI proof of Groth and Sahai [GS08] (roughly 800 group elements, see Section 1.1). However, signatures are still much bigger than the ones of [TGL+19a] (9 group elements and 4 scalars), but we would like to stress the fact that using a random oracle makes it possible to prove a much simpler relation since all the unforgeability-related equations are replaced by the ability to extract secret keys with rewinding. The construction $\mathsf{AMF}_{\mathsf{EC}}$ could still be improved, but the purpose of this section is not to provide the shortest signatures possible, but rather to give an intelligible example of a somehow practical AMF scheme provably secure in the plain model using the approach in Section 4.

5.2 Security Proofs for $\mathsf{AMF}_{\mathsf{EC}}$

We now prove that our efficient construction $\mathsf{AMF}_{\mathsf{EC}}$ described in Section 5.1 is a secure AMF scheme with strong deniability. We point out that this construction is not tightly secure because the security

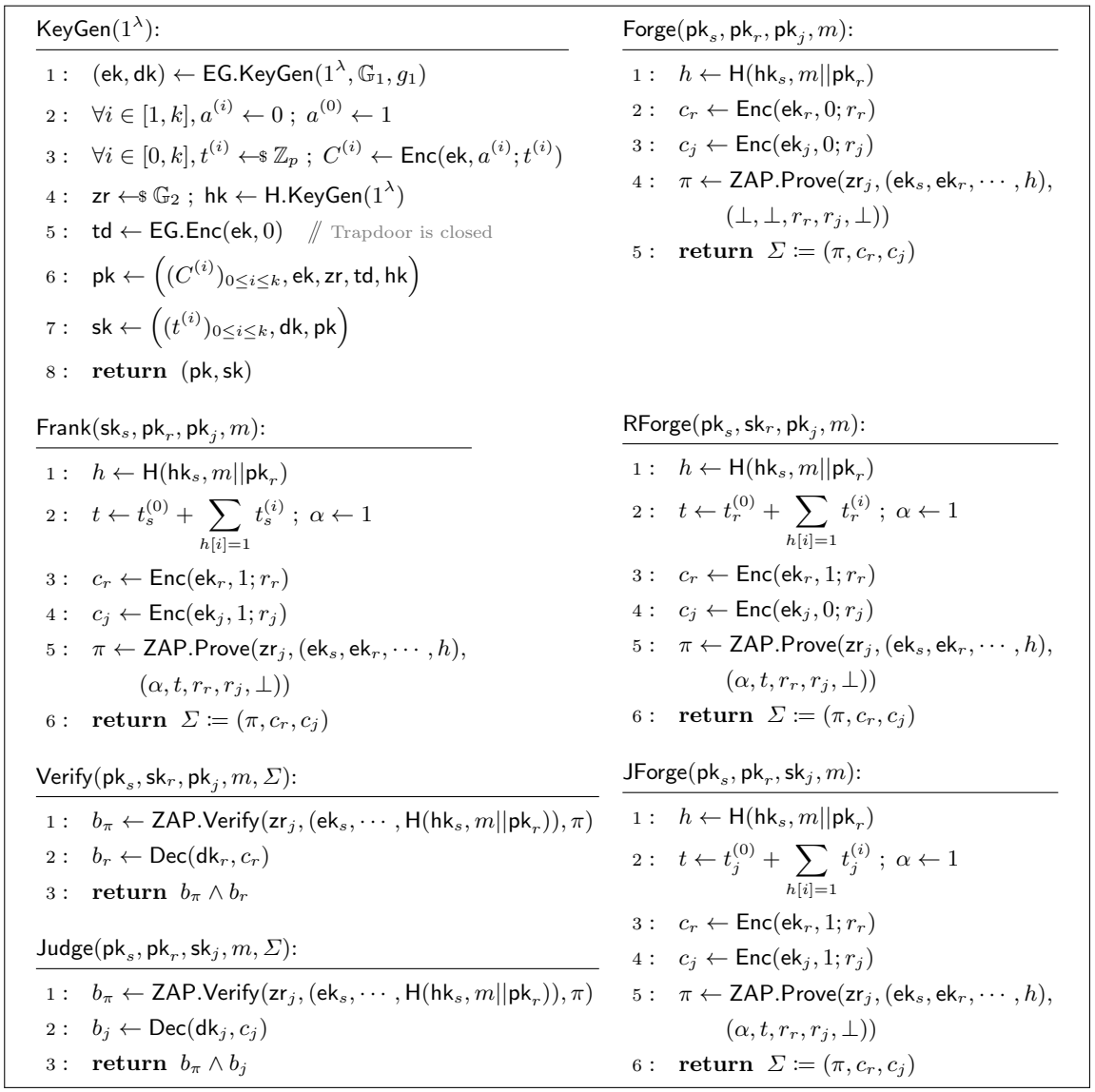


Fig. 8: Algorithms of our efficient construction $\mathsf{AMF}_{\mathsf{EC}}$: The encryption randomness r_r and r_j are assumed to be generated honestly when needed. The symbol $\bot$ represents the universal void witness.

loss for the receiver binding property depends on the number of Frank queries that an adversary is allowed to make.

Theorem 9 (Correctness). *The AMF scheme* $\mathsf{AMF}_{\mathsf{EC}}$ *is (unconditionally) correct.*

Proof. The correctness of $\mathsf{AMF}_{\mathsf{EC}}$ is a consequence of the correctness of ElGamal PKE, of the homomorphic properties of ElGamal ciphertexts and of the correctness of ZAP. Indeed, the witness used to produce the ZAP proof in the Frank algorithm is a valid one (because Conjunction $(1')$ is true) and all four ciphertexts are genuine encryptions of 1. □

Theorem 10 (Receiver Binding). *If the* DDH *assumption holds in* $\mathbb{G}_1$*, if the* $\mathcal{L}_1$-19-ExtKerMDH *assumption holds in* $\mathbb{G}_2$ *and if* H *is collision resistant then the AMF scheme* $\mathsf{AMF}_{\mathsf{EC}}$ *is receiver binding. More precisely, if* $\mathcal{A}$ *is an adversary against the receiver binding security of* $\mathsf{AMF}_{\mathsf{EC}}$ *making at most* Q Frank *queries (with* kQ^2 *much smaller than* p*), then there exist adversaries* $\mathcal{G}, \mathcal{G}'$ *and* $\mathcal{G}''$ *against the* DDH *assumption in* $\mathbb{G}_1$*, the* $\mathcal{L}_1$-19-ExtKerMDH *assumption in* $\mathbb{G}_2$ *and the collision resistance of* H *such that*

$$\mathsf{Adv}^{\mathsf{r\text{-}Bind}}_{\mathsf{AMF}_{\mathsf{EC}}}(\mathcal{A}) \le Q\sqrt{k} \cdot \left(4 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\mathcal{G}) + \mathsf{Adv}^{\mathsf{ExtKerMDH}}_{\mathbb{G}_2}(\mathcal{G}')\right) + \mathsf{Adv}^{\mathsf{CR}}_{\mathsf{H}}(\mathcal{G}'')$$

Note that the running time of $\mathcal{G}$, $\mathcal{G}'$ and $\mathcal{G}''$ is about that of $\mathcal{A}$.

Proof. The goal of the proof is to start from the original r-Bind game and to slightly drift toward a game in which the adversary produces a valid proof for a statement not in $\mathcal{L}'$. In this proof, $\mathcal{A}$ is an adversary against the r-Bind security of AMF making at most Q Frank queries, and we call *challenger* the PPT algorithm that interact with it.

Game $\mathsf{G_0}$. Let game $\mathsf{G_0}$ be the r-Bind game. The sender and judge key pairs are generated honestly, in particular the two public keys only contain encryptions of 0 and 1, and td_j encrypts 0. Frank and Judge queries are answered using the genuine algorithms.

$$\Pr[\mathsf{G_0} \Rightarrow 1] = \Pr\left[\mathsf{r\text{-}Bind}^{\mathcal{A}}_{\mathsf{AMF_{EC}}} \Rightarrow 1\right] \tag{8}$$

Game $\mathsf{G_1}$. Game $\mathsf{G_1}$ is similar to Game $\mathsf{G_0}$ but the challenger now aborts in the end if the hash of the forgery $\mathsf{H}(\mathsf{hk}_s, m^*||\mathsf{pk}_r^*)$ is equal to the hash $\mathsf{H}(\mathsf{hk}_s, m||\mathsf{pk}_r)$ of any Frank query (m, pk_r) made by the adversary during the game. Because $\mathcal{A}$ can only win game $\mathsf{G_0}$ if it outputs a forgery such that the tuple (m^*, pk_r^*) has never been queried before, a reduction to the collision resistance of H yields:

$$|\Pr[\mathsf{G_1} \Rightarrow 1] - \Pr[\mathsf{G_0} \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{CR}}_{\mathsf{H}}(\mathcal{G}'') \tag{9}$$

for a suitable adversary $\mathcal{G}''$.

Game $\mathsf{G_{2.1}}$. In Game $\mathsf{G_{2.1}}$, some fresh $b^{(i)} \in \mathbb{Z}_p$ for $0 \leq i \leq k$ are sampled as the a_i in the proof of Theorem 6 of [HK12]: let J be a positive integer in the order of Q^2 and sample uniformly and independently $b_{i,v}$ in $\{-1, 0, 1\}$ for $1 \leq i \leq k$ and $1 \leq v \leq J$. Then for all $1 \leq i \leq k$ set $b^{(i)} \leftarrow \sum_{v=1}^{J} b_{i,v}$ and finally $b^{(0)} \leftarrow (-1)$. At the very end of $\mathsf{G_{2.1}}$ the challenger aborts:

- if during that game, $\mathcal{A}$ sent any Frank query for a message m and public key pk_r such that $b^{(h)} := b^{(0)} + \sum_{h[i]=1} b^{(i)} = 0$ with $h = \mathsf{H}(\mathsf{hk}_s, m||\mathsf{pk}_r)$; or
- if in the end of the r-Bind game, $\mathcal{A}$ gave a forgery of the form (m^*, pk_r^*) such that $b^{(h^*)} := b^{(0)} + \sum_{h^*[i]=1} b^{(i)} \neq 0$, with $h^* = \mathsf{H}(\mathsf{hk}_s, m^*||\mathsf{pk}_r^*)$.

In other words, the challenger ensures that the all the $b^{(h)}$ are invertible in $\mathbb{Z}_p$ and that $b^{(h^*)}$ is not. We defer to Appendix A.4 the proof of the following lemma, which is a consequence of [HK12, Theorem 6].

Lemma 2. *If $(Q^2 \cdot k)$ is much smaller than p, the probability that the challenger aborts at the end of game $\mathsf{G_{2.1}}$ (provided that it produced a valid forgery for a message and a public key with a hash never seen before) is smaller than $(1 - \frac{1}{Q\sqrt{k}})$.*

As a consequence,

$$\Pr[\mathsf{G_1} \Rightarrow 1] \leq Q\sqrt{k} \cdot \Pr[\mathsf{G_{2.1}} \Rightarrow 1]. \tag{10}$$

Game $\mathsf{G_{2.2}}$. In $\mathsf{G_{2.2}}$, the challenger aborts as soon as it receives a Frank query such that $b^{(h)} = 0$ (it does not wait until the end of the game). Aborting earlier does not change the success probability (recall that the $b^{(i)}$ are still perfectly hidden from $\mathcal{A}$) so

$$\Pr[\mathsf{G_{2.2}} \Rightarrow 1] = \Pr[\mathsf{G_{2.1}} \Rightarrow 1]. \tag{11}$$

Game $\mathsf{G_3}$. In Game $\mathsf{G_3}$, the plaintexts in the $C_j^{(i)}$ ciphertexts (judge public key) are swapped from $a_j^{(i)}$ to $b_j^{(i)}$ during key generation. The Frank queries are still answered with a genuine sender secret key, and the Judge algorithm do not make use of the ciphertexts $C_j^{(i)}$ but only the judge decryption key dk_j. A reduction to the MC-IND-CPA security of ElGamal PKE gives

$$|\Pr[\mathsf{G_3} \Rightarrow 1] - \Pr[\mathsf{G_{2.2}} \Rightarrow 1]| \leq 2 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\mathcal{G}_3) \tag{12}$$

for a suitable DDH adversary $\mathcal{G}_3$.

Game $\mathsf{G_4}$. In Game $\mathsf{G_4}$, the challenger swaps the witness of all the answers to Frank queries (m, pk_r) from $w = (\alpha, t, r_r, r_j, \perp)$ to $w' = (\alpha', t', r_r, r_j, \perp)$ with $h = \mathsf{H}(\mathsf{hk}_s, m||\mathsf{pk}_r)$, α' the *inverse* of $b^{(h)} = b^{(0)} + \sum_{h[i]=1} b^{(i)}$ in $\mathbb{Z}_p$ and $t' = \alpha' \cdot (t_j^{(0)} + \sum_{h[i]=1} t_j^{(i)})$. Such inverse always exists because since game $\mathsf{G_{2.2}}$, the challenger aborts when it receives a message such that $b^{(h)} = 0$. Due to the homomorphic properties of ElGamal ciphertexts[10], $\alpha' \cdot (C_j^{(0)} + \sum_{h[i]=1} C_j^{(i)}) = \mathsf{Enc}(\mathsf{ek}_j, 1; t')$ so w' is a valid witness. The perfectly witness-indistinguishability of ZAP yields

$$\Pr[\mathsf{G_4} \Rightarrow 1] = \Pr[\mathsf{G_3} \Rightarrow 1]. \tag{13}$$

Game $\mathsf{G_5}$. In Game $\mathsf{G_5}$, the plaintext inside $C_s^{(0)}$ (first ciphertext of the sender public key) is swapped from 1 to 0 during key generation. The Frank queries are still answered with a ZAP proof using the witness of game $\mathsf{G_4}$. A reduction to the $\mathsf{MC\text{-}IND\text{-}CPA}$ security of ElGamal PKE yields

$$|\Pr[\mathsf{G_5} \Rightarrow 1] - \Pr[\mathsf{G_4} \Rightarrow 1]| \leq 2 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\mathcal{G}_5) \tag{14}$$

for a suitable DDH adversary $\mathcal{G}_5$.

Conclusion. In Game $\mathsf{G_5}$, the adversary sends a valid forgery (pk_r^*, m^*); however it satisfies

$$\left(\mathsf{ek}_s, \mathsf{ek}_r^*, \mathsf{ek}_j, \mathsf{td}_j, (C_s^{(i)}, {C_r^{(i)}}^*, C_j^{(i)})_{0 \leq i \leq k}, c_r^*, c_j^*, h^*\right) \notin \mathcal{L}'$$

since:

- td_j is a genuine encryption of 0 so Equation $(0')$ must be false since ElGamal PKE is perfectly correct and since the pair $(\mathsf{ek}_j, \mathsf{dk}_j)$ has been generated honestly.
- $\mathsf{Dec}(\mathsf{dk}_j, {c_j}^*) = 1$ so there is no ${r_j}^* \in \mathbb{Z}_p$ such that ${c_j}^* = \mathsf{Enc}(\mathsf{ek}_j, 0; {r_j}^*)$ for the same reason as above. Therefore Conjunctions $(2')$ and $(3')$ evaluate to false.
- $b^{(h^*)} = 0$ (abort condition in Game $\mathsf{G_2}$) so $C_j^{(h^*)} := (C_j^{(0)} + \sum_{h^*[i]=1} C_j^{(i)})$ encrypts 0 and thus there is no $(\alpha^*, t^*) \in \mathbb{Z}_p^2$ such that $C_j^{(h^*)} \cdot \alpha^* = \mathsf{Enc}(\mathsf{ek}_j, 1; t^*)$[11].
- $\forall i \in [0, k], a_s^{(i)} = 0$ (Game $\mathsf{G_5}$) so for the same reason as above, there is no $(\alpha^*, t^*) \in \mathbb{Z}_p^2$ such that $(C_s^{(0)} + \sum_{h^*[i]=1} C_s^{(i)}) \cdot \alpha^* = \mathsf{Enc}(\mathsf{ek}_j, 1; t^*)$.

A reduction to the computational soundness of ZAP yields

$$\Pr[\mathsf{G_4} \Rightarrow 1] \leq \mathsf{Adv}^{\mathsf{ExtKerMDH}}_{\mathbb{G}_2}(\mathcal{G}') \tag{15}$$

for a suitable $\mathcal{L}_1$-19-$\mathsf{ExtKerMDH}$ adversary $\mathcal{G}'$[12].

Combining Equations (8) to (15) yields the result stated in Theorem 10. □

We state here the theorems related to sender binding and three deniability properties of $\mathsf{AMF_{EC}}$, and defer the proofs to Appendix A.

Theorem 11 (Sender Binding). *If* DDH *assumption holds in* $\mathbb{G}_1$ *and if the* $\mathcal{L}_1$-19-$\mathsf{ExtKerMDH}$ *assumption holds in* $\mathbb{G}_2$ *then the AMF scheme* $\mathsf{AMF_{EC}}$ *is sender binding. Furtheremore, if* $\mathcal{A}$ *is an adversary against the sender binding security of* $\mathsf{AMF_{EC}}$ *then there exist adversaries* $\mathcal{H}$ *and* $\mathcal{H}'$ *against the* DDH *assumption in* $\mathbb{G}_1$ *and the* $\mathcal{L}_1$-19-$\mathsf{ExtKerMDH}$ *assumption in* $\mathbb{G}_2$ *such that*

$$\mathsf{Adv}^{\mathsf{r\text{-}Bind}}_{\mathsf{AMF_{EC}}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\mathcal{H}) + \mathsf{Adv}^{\mathsf{ExtKerMDH}}_{\mathbb{G}_2}(\mathcal{H}')$$

Note that the running time of $\mathcal{H}$*, and* $\mathcal{H}'$ *is about that of* $\mathcal{A}$*.*

Theorem 12 (Deniability Properties). *If the* DDH *assumption holds in* $\mathbb{G}_1$*, then the AMF scheme* $\mathsf{AMF_{EC}}$ *achieves universal deniability, and receiver compromise deniability. More precisely, if* $\mathcal{A}$ *is an*

[10, 11] See Lemma 1: *Homomorphic Properties of ElGamal Ciphertexts*, Section 2.6

[12] The constant 19 comes from the size of the witnesses used to define $\mathcal{L}_1'$, $\mathcal{L}_2'$, $\mathcal{L}_3'$ and $\mathcal{L}_4'$, plus the disjunction overhead. See Section 2.6.

adversary against the universal deniability (resp. receiver compromise deniability) security of $\mathsf{AMF}_{\mathsf{EC}}$*, then there exist adversaries* $\mathcal{I}$ *(resp.* $\mathcal{I}'$*) against the* DDH *assumption in* $\mathbb{G}_1$ *such that*

$$\mathsf{Adv}^{\mathsf{UnivDen}}_{\mathsf{AMF}_{\mathsf{EC}}}(\mathcal{A}) \leq 8 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\mathcal{I}) \quad \left(\textit{resp. } \mathsf{Adv}^{\mathsf{RecCompDen}}_{\mathsf{AMF}_{\mathsf{EC}}}(\mathcal{A}) \leq 6 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\mathcal{I}')\right)$$

Note that the running time of $\mathcal{I}$ *and* $\mathcal{I}'$ *is about that of* $\mathcal{A}$*.*

Moreover, $\mathsf{AMF}_{\mathsf{EC}}$ *achieves perfect judge compromise deniability (i.e., we have* $\mathsf{Adv}^{\mathsf{JdgCompDen}}_{\mathsf{AMF}_{\mathsf{EC}}}(\mathcal{A}) = 0$*).*

From these results, we conclude that $\mathsf{AMF}_{\mathsf{EC}}$ is a secure AMF scheme with strong deniability.

Corollary 2. *If the* DDH *assumption holds in* $\mathbb{G}_1$ *and the* $\mathcal{L}_1$*-19-*ExtKerMDH *assumption holds in* $\mathbb{G}_2$ *relative to the pairing groups* $\mathcal{PG}$*, and* H *is a collision-resistant hash function, then the AMF scheme* $\mathsf{AMF}_{\mathsf{EC}}$ *is a secure AMF scheme with strong deniability.*

6 Asymmetric Message Franking from Plain LWE

In this section we adapt the generic construction $\mathsf{AMF}_{\mathsf{GC}}$ to make it work with the ZAP argument system designed by Chatterjee et al. [CGH+21]. This ZAP was initially used as a building block to construct compact ring signatures from plain LWE. Their *compact relaxed ZAP for* $\mathsf{NP} \cap \mathsf{coNP}$ is defined for two languages, the latter being an extension of the complement of the former, and only guarantees some form of selective soundness. After a brief summary of this ZAP argument system (Section 6.1), we describe the languages used to instantiate the ZAP for our generic construction (Section 6.2) and finally prove that the new construction is secure even with the selective soundness provided by ZAP (Section 6.3). A concrete instantiation is given in Section 6.4.

6.1 Recap ZAP for NP ∩ coNP

We first recap the (relaxed) ZAP for $\mathsf{NP} \cap \mathsf{coNP}$, defined by Chatterjee et al. [CGH+21]. Let $\mathcal{X}$ be a set and $\mathcal{L} \subseteq \mathcal{X}$ be an NP language with witness space $\mathcal{W}$ and an efficiently recognizable relation $\mathcal{R}$, i.e., $\mathcal{L} = \{\mathsf{x} \in \mathcal{X} : \exists \mathsf{w} \in \mathcal{W} \text{ s.t. } \mathcal{R}(\mathsf{x}, \mathsf{w}) = 1\}$. In addition, let $\widetilde{\mathcal{L}} \subseteq \mathcal{X}$ be another NP language with witness space $\widetilde{\mathcal{W}}$ such that $\widetilde{\mathcal{L}}$ is a subset of the complement[13] of $\mathcal{L}$, i.e. $\widetilde{\mathcal{L}} \subset \bar{\mathcal{L}} = \mathcal{X} \setminus \mathcal{L}$. Finally, let $\widetilde{\mathcal{R}}$ be the NP relation corresponding to $\widetilde{\mathcal{L}}$. A ZAP for $\mathsf{NP} \cap \mathsf{coNP}$ is a public-coin 2-round protocol that consists of three algorithms $\mathsf{ZAP} = (\mathsf{ZAP.V}, \mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$. Note that ZAP.V outputs a uniformly random bit-string whose length only depends on λ and on the languages $\mathcal{L}$ and $\widetilde{\mathcal{L}}$. In our case, $\mathcal{L}$ and $\widetilde{\mathcal{L}}$ will depend on λ, so we can define a polynomial $\ell = \ell(\lambda)$ which describes such length.

ZAP for $\mathsf{NP} \cap \mathsf{coNP}$ satisfies correctness, witness-indistinguishability and selective non-witness and adaptive statement soundness. The former two notions are defined identically as ZAP for NP. Below, we define the latter new notion.

Definition 26 (Selective Non-Witness and Adaptive Statement Soundness). *We formalize selective non-witness and adaptive statement soundness (*selective soundness *for short) in the game* $\mathsf{SelSound}^{\mathcal{A},\widetilde{w}}_{\mathsf{ZAP}}$ *in Figure 9. For an adversary* $\mathcal{A}$ *and a ZAP system* ZAP *for* NP *languages* $(\mathcal{L}, \widetilde{\mathcal{L}})$ *with* $\widetilde{\mathcal{L}} \subset \bar{\mathcal{L}}$*, we define the advantage of* $\mathcal{A}$ *against* ZAP *as*

$$\mathsf{Adv}^{\mathsf{SelSound}}_{\mathsf{ZAP}}(\mathcal{A}) \coloneqq \max_{\widetilde{w} \in \widetilde{\mathcal{W}}} \Pr\left[\mathsf{SelSound}^{\mathcal{A},\widetilde{w}}_{\mathsf{ZAP}} \Rightarrow 1\right]$$

We say ZAP *is* selectively sound *if* $\mathsf{Adv}^{\mathsf{SelSound}}_{\mathsf{ZAP}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$ *for any efficient adversary* $\mathcal{A}$*.*

[13] In [CGH+21], $\widetilde{\mathcal{L}} \subset \bar{\mathcal{L}} \times \{0,1\}^*$ is defined as a *super*-complement of $\mathcal{L}$. In this work, we do not need the second part of such *super*-complement thus we only require $\widetilde{\mathcal{L}} \subset \bar{\mathcal{L}}$.

$\mathsf{SelSound}^{\mathcal{A},\widetilde{w}}_{\mathsf{ZAP}}$:

1 : $\mathsf{zr} \leftarrow\!\$\ \{0,1\}^{\ell}$
2 : $(\mathsf{x}, \pi) \leftarrow \mathcal{A}(1^{\lambda}, \widetilde{w}, \mathsf{zr})$
3 : **if** $(\mathsf{x}, \widetilde{w}) \in \widetilde{\mathcal{R}} \wedge \mathsf{Verify}(\mathsf{zr}, \mathsf{x}, \pi) = 1$ **then return** 1
4 : **else return** 0

Fig. 9: Selective Non-Witness and Adaptive Statement Soundness for ZAP.

6.2 Proposed Construction $\mathbf{AMF'_{GC}}$

Description of $\widetilde{\mathcal{L}}$. To see that ZAP for $\mathsf{NP} \cap \mathsf{coNP}$ is sufficient for our AMF generic construction, we describe $\widetilde{\mathcal{L}}$ and prove that it is a subset of $\bar{\mathcal{L}}$. Let $\widetilde{\mathcal{L}}$ be the NP language defined by $\widetilde{\mathcal{R}}$ in Equation $(\widetilde{\star})$ with the non-witness space $\widetilde{\mathcal{W}} = (\mathcal{DK} \cup \{\bot\})^4$, where $\mathcal{DK}$ is the decryption key space of PKE. Like in [CGH+21], witnesses related to $\widetilde{\mathcal{R}}$ will be referred to as *non-witness* and noted $\widetilde{w}$ to avoid any confusion.

$$\widetilde{\mathcal{R}} := \{\ (\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{pk}_j, c_\sigma, c'_\sigma, c_r, c_j, \mathsf{td}_j, m_r), (\widetilde{\mathsf{dk}_s}, \mathsf{dk}_r, \mathsf{dk}_j, \widetilde{\mathsf{dk}_j}) \quad \text{such that} \tag{$\widetilde{\star}$}$$
$$\text{let } \widetilde{\mathsf{wf}_s} := \mathsf{WellFormed}_{\mathsf{PKE}}(\widetilde{\mathsf{ek}_s}, \widetilde{\mathsf{dk}_s}) \text{ and } \mathsf{wf}_r := \mathsf{WellFormed}_{\mathsf{PKE}}(\mathsf{ek}_r, \mathsf{dk}_r)$$
$$\text{and } \mathsf{wf}_j := \mathsf{WellFormed}_{\mathsf{PKE}}(\mathsf{ek}_j, \mathsf{dk}_j) \text{ and } \widetilde{\mathsf{wf}_j} := \mathsf{WellFormed}_{\mathsf{PKE}}(\widetilde{\mathsf{ek}_j}, \widetilde{\mathsf{dk}_j}) \text{ in}$$
$$\Big[(\widetilde{\mathsf{wf}_s} \vee \widetilde{\mathsf{dk}_s} = \bot) \wedge (\mathsf{wf}_r \vee \mathsf{dk}_r = \bot) \wedge (\mathsf{wf}_j \vee \mathsf{dk}_j = \bot) \wedge (\widetilde{\mathsf{wf}_j} \vee \widetilde{\mathsf{dk}_j} = \bot)\Big] \tag{$\widetilde{0'}$}$$
$$\wedge\ [\mathsf{wf}_j \wedge \mathsf{Dec}(\mathsf{dk}_j, \mathsf{td}_j) \neq 1] \tag{$\widetilde{0}$}$$
$$\wedge\ \Big[\Big(\widetilde{\mathsf{wf}_s} \wedge \mathsf{SIG.Verify}(\mathsf{vk}_s, m_r, \mathsf{Dec}(\widetilde{\mathsf{dk}_s}, c_\sigma)) \neq 1\Big) \vee (\mathsf{wf}_r \wedge \mathsf{Dec}(\mathsf{dk}_r, c_r) \neq 1) \vee (\mathsf{wf}_j \wedge \mathsf{Dec}(\mathsf{dk}_j, c_j) \neq 1)\Big] \tag{$\widetilde{1}$}$$
$$\wedge\ [(\mathsf{wf}_r \wedge \mathsf{Dec}(\mathsf{dk}_r, c_r) \neq 0) \vee (\mathsf{wf}_j \wedge \mathsf{Dec}(\mathsf{dk}_j, c_j) \neq 0)] \tag{$\widetilde{2}$}$$
$$\wedge\ \Big[\Big(\widetilde{\mathsf{wf}_j} \wedge \mathsf{SIG.Verify}(\mathsf{vk}_r, m_r, \mathsf{Dec}(\widetilde{\mathsf{dk}_j}, c'_\sigma)) \neq 1\Big) \vee (\mathsf{wf}_r \wedge \mathsf{Dec}(\mathsf{dk}_r, c_r) \neq 1) \vee (\mathsf{wf}_j \wedge \mathsf{Dec}(\mathsf{dk}_j, c_j) \neq 0)\Big] \tag{$\widetilde{3}$}$$
$$\wedge\ \Big[\Big(\widetilde{\mathsf{wf}_s} \wedge \mathsf{SIG.Verify}(\mathsf{vk}_j, m_r, \mathsf{Dec}(\widetilde{\mathsf{dk}_s}, c'_\sigma)) \neq 1\Big) \vee (\mathsf{wf}_r \wedge \mathsf{Dec}(\mathsf{dk}_r, c_r) \neq 1) \vee (\mathsf{wf}_j \wedge \mathsf{Dec}(\mathsf{dk}_j, c_j) \neq 1)\Big]\} \tag{$\widetilde{4}$}$$

with $\mathsf{pk}_s = (\mathsf{vk}_s, \mathsf{ek}_s, \widetilde{\mathsf{ek}_s}, \mathsf{zr}_s)$, $\mathsf{pk}_r = (\mathsf{vk}_r, \mathsf{ek}_r, \widetilde{\mathsf{ek}_r}, \mathsf{zr}_r)$ and $\mathsf{pk}_j = (\mathsf{vk}_j, \mathsf{ek}_j, \widetilde{\mathsf{ek}_j}, \mathsf{zr}_j)$.

Lemma 3. *If* PKE *is perfectly correct then the language* $\widetilde{\mathcal{L}}$ *defined by* $\widetilde{\mathcal{R}}$ *in* $(\widetilde{\star})$ *is a subset of the complement of* $\mathcal{L}$ *defined by* $\mathcal{R}$ *in* $(\star)$*, i.e.,* $\widetilde{\mathcal{L}} \subset \bar{\mathcal{L}}$.

Proof. Lemma 3 is a direct consequence of the fact that for all i in $\{0,1,2,3,4\}$, it holds that $(\widetilde{i}) \implies \neg(i)$. Let $x = (\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{pk}_j, \cdots, m_r)$ be in $\widetilde{\mathcal{L}}$ and let $\widetilde{w} = (\widetilde{\mathsf{dk}_s}, \mathsf{dk}_r, \mathsf{dk}_j, \widetilde{\mathsf{dk}_j}) \in \widetilde{\mathcal{W}}$ be a non-witness for x, i.e., such that $(x, \widetilde{w}) \in \widetilde{\mathcal{R}}$. Finally, let $w = (\sigma, \sigma', \cdots, r_{\mathsf{td}})$ be a *potential* witness for $\mathcal{R}$. Using the De Morgan's laws and because PKE is perfectly correct, the following five implications hold:

$$\forall i \in \{0,1,2,\cdots,4\}, (\widetilde{i}) \implies \neg(i)$$

where $(\widetilde{i})$ are the equations of $(\widetilde{\star})$ and (i) are the equations of $(\star)$. For instance when $i = 1$, we have

$$\begin{aligned}
\widetilde{\mathsf{wf}_s} \wedge \mathsf{SIG.Verify}(\mathsf{vk}_s, m_r, \mathsf{Dec}(\widetilde{\mathsf{dk}_s}, c_\sigma)) \neq 1 &\implies \mathsf{SIG.Verify}(\mathsf{vk}_s, m_r, \sigma) \neq 1 \\
&\quad \vee c_\sigma \neq \mathsf{Enc}(\widetilde{\mathsf{ek}_s}, \sigma; r_\sigma) \\
\mathsf{wf}_r \wedge \mathsf{Dec}(\mathsf{dk}_r, c_r) \neq 1 &\implies c_r \neq \mathsf{Enc}(\mathsf{ek}_r, 1; r_r) \\
\mathsf{wf}_j \wedge \mathsf{Dec}(\mathsf{dk}_j, c_j) \neq 1 &\implies c_j \neq \mathsf{Enc}(\mathsf{ek}_j, 1; r_j),
\end{aligned}$$

and thus $(\widetilde{1}) \implies \neg(1)$. Using again De Morgan's laws,

$$\bigwedge_{i=0}^{4} (\widetilde{i}) \implies \neg \bigvee_{i=0}^{4} (i) \quad \text{and, a fortiori,} \quad (\widetilde{0'}) \wedge \bigwedge_{i=0}^{4} (\widetilde{i}) \implies \neg \bigvee_{i=0}^{4} (i)$$

thus $x \notin \mathcal{L}$. Therefore, $\widetilde{\mathcal{L}} \subset \bar{\mathcal{L}}$, which concludes the proof. □

Additional Properties for the PKE Scheme. To construct a generic AMF from ZAP for $\mathsf{NP} \cap \mathsf{coNP}$, we require a PKE scheme with some additional properties introduced in [CGH+21]. First, we assume in this Section that there exists an algorithm $\mathsf{WellFormed}_{\mathsf{PKE}}$ that can check if a key pair is well-formed. Therefore, correctness for PKE now requires that for any key pair $(\mathsf{ek}, \mathsf{dk})$ and any message m, $\mathsf{WellFormed}_{\mathsf{PKE}}(\mathsf{ek}, \mathsf{dk}) \implies \mathsf{Dec}(\mathsf{dk}, \mathsf{Enc}(\mathsf{ek}, m)) = m$. Also, the key generation algorithm of PKE must sample a decryption key dk from some distribution $\mathsf{Dist}_{\mathcal{DK}}$, and then compute an associated encryption key with some $\mathsf{GenWithKey}(\mathsf{dk})$ procedure. Finally, we require PKE to have *injective key generation*, meaning that for any decryption key dk,

$$\Pr_{\mathsf{ek} \leftarrow \mathsf{GenWithKey}(\mathsf{dk})}\left[\exists \mathsf{dk}' \neq \mathsf{dk} \text{ such that } \mathsf{WellFormed}_{\mathsf{PKE}}(\mathsf{ek}, \mathsf{dk}')\right] \leq \gamma_{\mathsf{KI}}$$

with γ_{KI} negligible in λ. It is proved that Regev's LWE-based PKE scheme satisfies these additional requirements [CGH+21, Theorem 2].

$\mathsf{AMF}'_{\mathsf{GC}}$*: Generic AMF scheme with ZAP for* $\mathsf{NP} \cap \mathsf{coNP}$. The construction of the AMF scheme $\mathsf{AMF}'_{\mathsf{GC}}$ is the same as the AMF scheme $\mathsf{AMF}_{\mathsf{GC}}$ in Section 4 but using the ZAP proof system for $(\mathcal{L}, \widetilde{\mathcal{L}})$ defined above. Since the syntax of the two ZAP proof systems are identical, Figure 6 also describes $\mathsf{AMF}'_{\mathsf{GC}}$. $\mathsf{AMF}'_{\mathsf{GC}}$ checks the well-formedness of its keys with the algorithm $\mathsf{WellFormed}_{\mathsf{AMF}'_{\mathsf{GC}}}(\mathsf{pk}, \mathsf{sk}) := \mathsf{WellFormed}_{\mathsf{SIG}}(\mathsf{vk}, \mathsf{sik}) \wedge \mathsf{WellFormed}_{\mathsf{PKE}}(\mathsf{ek}, \mathsf{dk})$.

6.3 Security Proofs for $\mathsf{AMF}'_{\mathsf{GC}}$

The ZAP proof system for $\mathsf{NP} \cap \mathsf{coNP}$ (described in [CGH+21]) is correct, statistically witness indistinguishable and *selective non-witness and adaptive statement sound*. Therefore, to prove that $\mathsf{AMF}'_{\mathsf{GC}}$ described in Section 6.2 is secure, only accountability theorems have to be adapted.

Theorem 13 (Receiver Binding). *If* SIG *is* EUF-CMA *secure, if* PKE *is perfectly correct and has injective key generation, and if* ZAP *is selective non-witness and adaptive statement sound then the AMF scheme* $\mathsf{AMF}'_{\mathsf{GC}}$ *is receiver binding. Furthermore, if* $\mathcal{A}$ *is an adversary against the receiver binding security of* $\mathsf{AMF}'_{\mathsf{GC}}$*, then there exist adversaries* $\mathcal{J}$ *and* $\mathcal{J}'$ *against the* EUF-CMA *security of* SIG *and the selective soundness of* ZAP *such that*

$$\mathsf{Adv}^{\mathsf{r\text{-}Bind}}_{\mathsf{AMF}'_{\mathsf{GC}}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{J}) + \mathsf{Adv}^{\mathsf{SelSound}}_{\mathsf{ZAP}}(\mathcal{J}') + 2 \cdot \gamma_{\mathsf{KI}}$$

with $\gamma_{\mathsf{KI}} = \mathsf{negl}(\lambda)$ *related to the injective key generation of* PKE*. Note that the running time of* $\mathcal{J}$ *and* $\mathcal{J}'$ *is about that of* $\mathcal{A}$*.*

Proof. The proof of Theorem 13 is similar to the proof of Theorem 4: let $\mathcal{A}$ be an adversary against the receiver binding security of $\mathsf{AMF}'_{\mathsf{GC}}$ and suppose without loss of generality that $\mathcal{A}$ never outputs a forbidden forgery. Let $\mathcal{J}$, $\mathcal{J}_1$ and $\mathcal{J}_2$ be adversaries against the EUF-CMA security of SIG identical to $\mathcal{B}$, $\mathcal{B}_1$ and $\mathcal{B}_2$ respectively.

Game G_1. Let game G_1 be the $\mathsf{EUF\text{-}CMA}^{\mathcal{J}}_{\mathsf{SIG}}$ game with the same winning condition as in the proof of Theorem 4. We have:

$$\Pr[\mathsf{G}_1 \Rightarrow 1] \geq \Pr[\mathsf{G}_0 \Rightarrow 1]. \tag{16}$$

Game G_2. We change the winning condition in the end: $\mathcal{J}$ is successful in Game G_2 if $\mathsf{EUF\text{-}CMA}^{\mathcal{J}}_{\mathsf{SIG}}$ does not abort, and if $(x^*, \widetilde{w}^*) \notin \widetilde{\mathcal{R}}$ for all non-witnesses of the form $\widetilde{w}^* = (\widetilde{\mathsf{dk}^*_s}, \bot, \mathsf{dk}^*_j, \bot)$, with $x^* = (\mathsf{pk}_s, \mathsf{pk}^*_r, \mathsf{pk}_j, c^*_\sigma, \cdots, m^*_r)$ and $\widetilde{\mathsf{dk}^*_s}, \mathsf{dk}^*_j \in \mathcal{DK}$. Let $\mathcal{J}'$ (described below) be the adversary against the selective soundness of ZAP for some arbitrary fixed non-witness of the form $\widetilde{w} = (\widetilde{\mathsf{dk}_s}, \bot, \mathsf{dk}_j, \bot)$. $\mathcal{J}'$ takes as input $\widetilde{w}$ and some ZAP randomness zr, then generates $\widetilde{\mathsf{ek}_s} \leftarrow \mathsf{PKE.GenWithKey}(\widetilde{\mathsf{dk}_s})$ and $\mathsf{ek}_j \leftarrow \mathsf{PKE.GenWithKey}(\mathsf{dk}_j)$, and finally runs $\mathcal{J}$ (identical to $\mathcal{B}$) but using $\widetilde{\mathsf{ek}_s}, \widetilde{\mathsf{dk}_s}, \mathsf{ek}_j, \mathsf{dk}_j$ and $\mathsf{zr}_j := \mathsf{zr}$ instead of generating new ones in the setup phase. Like $\mathcal{B}'$, $\mathcal{J}'$ outputs the ZAP statement and proof (x^*, π^*) associated with the forgery of $\mathcal{A}$ if $\mathsf{EUF\text{-}CMA}^{\mathcal{J}}_{\mathsf{SIG}}$ terminates. A reduction to the selective soundness of ZAP yields that if $\widetilde{w} = (\widetilde{\mathsf{dk}_s}, \bot, \mathsf{dk}_j, \bot)$ with $\widetilde{\mathsf{dk}_s}, \mathsf{dk}_j \leftarrow\!\!\$\ \mathsf{Dist}_{\mathcal{DK}}$, then

$$\begin{aligned}\Pr\Big[\mathsf{EUF\text{-}CMA}^{\mathcal{J}}_{\mathsf{SIG}} \text{ terminates} \wedge \exists\, \widetilde{\mathsf{dk}^*_s}, \mathsf{dk}^*_j \in \mathcal{DK} : (x^*, (\widetilde{\mathsf{dk}^*_s}, \bot, \mathsf{dk}^*_j, \bot)) \in \widetilde{\mathcal{R}}\Big] \\ \leq \Pr\Big[\mathsf{SelSound}^{\mathcal{J}', \widetilde{w}}_{\mathsf{ZAP}} \Rightarrow 1\Big] + 2 \cdot \gamma_{\mathsf{KI}}\end{aligned}$$

provided $\mathcal{J}$ generated $\widetilde{\mathsf{dk}_s}$ and dk_j in the setup phase. Indeed, suppose that $\mathcal{A}$ manages to output a valid forgery with a ZAP statement and proof (x^*, π^*) such that there exists $\widetilde{w}^* = (\widetilde{\mathsf{dk}_s^*}, \perp, \mathsf{dk}_j^*, \perp)$, with $\widetilde{\mathsf{dk}_s^*}, \mathsf{dk}_j^* \in \mathcal{DK}$ verifying $(x^*, \widetilde{w}^*) \in \widetilde{\mathcal{R}}$. Because of $(\widetilde{0'})$ in $\widetilde{\mathcal{R}}$, $(\widetilde{\mathsf{ek}_s}, \widetilde{\mathsf{dk}_s^*})$ and $(\mathsf{ek}_j, \mathsf{dk}_j^*)$ are well-formed. By injectivity of key generation of PKE, except with negligible probability $(2 \cdot \gamma_{\mathsf{KI}})$ over the choice of $\widetilde{\mathsf{dk}_s}$ and dk_j, we have $\widetilde{\mathsf{dk}_s^*} = \widetilde{\mathsf{dk}_s}$ and $\mathsf{dk}_j^* = \mathsf{dk}_j$, thus $\widetilde{w}^* = \widetilde{w}$. So $\mathcal{J}'$ is successful in the SelSound game for ZAP with fixed non-witness $\widetilde{w}$. Therefore

$$0 \leq \Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1] \leq \mathsf{Adv}^{\mathsf{SelSound}}_{\mathsf{ZAP}}(\mathcal{J}') + 2 \cdot \gamma_{\mathsf{KI}}. \tag{17}$$

Game G_3. Let game G_3 be the *original* $\mathsf{EUF\text{-}CMA}^{\mathcal{J}}_{\mathsf{SIG}}$ game. If $\mathcal{J}$ wins G_2, then $\mathsf{Dec}(\mathsf{dk}_j, c_j^*) = \mathsf{Dec}(\mathsf{dk}_j, c_j'^*) = 1$ and for the specific $\widetilde{w} = (\widetilde{\mathsf{dk}_s}, \perp, \mathsf{dk}_j, \perp)$, with $\widetilde{\mathsf{dk}_s}$ and dk_j generated by $\mathcal{J}$ in the setup phase, it holds that $(x^*, \widetilde{w}) \notin \widetilde{\mathcal{R}}$. Therefore, one of the five equations of $(\widetilde{\star})$ is false (using the previous notations, i.e., pk_r^* instead of pk_r etc.). Equation $(\widetilde{0})$ is true because td_j is a genuine encryption of 0. Disjunctions $(\widetilde{2})$ and $(\widetilde{3})$ are true because the clause $\mathsf{wf}_j \wedge \mathsf{Dec}(\mathsf{dk}_j, c_j{}^*) \neq 0$ is true. Therefore, $(\widetilde{1})$ or $(\widetilde{4})$ must be false. If $(\widetilde{1})$ is false then $\mathsf{SIG.Verify}(\mathsf{vk}_s, m_r^*, \mathsf{Dec}(\widetilde{\mathsf{dk}_s}, c_\sigma^*)) = 1$ because $(\widetilde{\mathsf{ek}_s}, \widetilde{\mathsf{dk}_s})$ is well-formed. Since the tuple (pk_r^*, m^*) has never been queried to $\mathcal{J}$ by hypothesis, $m_r^* = m^* || \mathsf{pk}_r^*$ has never been queried to the EUF-CMA challenger, so $\mathcal{J}_1$ is successful in the EUF-CMA forgery game. Similarly, if $(\widetilde{4})$ is false, $\mathcal{J}_2$ is successful in the EUF-CMA forgery game. Therefore, the probability that $\mathcal{J}$ runs a successful adversary is at least $1/2$:

$$\Pr[\mathsf{G}_3 \Rightarrow 1] \geq \frac{1}{2} \cdot \Pr[\mathsf{G}_2 \Rightarrow 1]. \tag{18}$$

Combining Equations (16) to (18) yields the result stated in Theorem 13. □

Theorem 14 (Sender Binding). *If* SIG *is* EUF-CMA *secure, if* PKE *is perfectly correct and have injective key generation, and if* ZAP *is selective non-witness and adaptive statement sound then the AMF scheme* $\mathsf{AMF}'_{\mathsf{GC}}$ *is sender binding. Furthermore, if* $\mathcal{A}$ *is an adversary against the sender binding security of* $\mathsf{AMF}'_{\mathsf{GC}}$*, then there exist adversaries* $\mathcal{K}$ *and* $\mathcal{K}'$ *against the* EUF-CMA *security of* SIG *and the selective soundness of* ZAP *such that*

$$\mathsf{Adv}^{\mathsf{s\text{-}Bind}}_{\mathsf{AMF}'_{\mathsf{GC}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{K}) + \mathsf{Adv}^{\mathsf{SelSound}}_{\mathsf{ZAP}}(\mathcal{K}') + 3 \cdot \gamma_{\mathsf{KI}}$$

with $\gamma_{\mathsf{KI}} = \mathsf{negl}(\lambda)$ *related to the injective key generation of* PKE*. Note that the running time of* $\mathcal{K}$ *and* $\mathcal{K}'$ *is about that of* $\mathcal{A}$*.*

Proof. The proof of Theorem 14 is similar to the proof of Theorem 5: let $\mathcal{A}$ be an adversary against the sender binding security of AMF'. Let $\mathcal{K}$, $\mathcal{K}_1$ and $\mathcal{K}_2$ be adversaries against the EUF-CMA security of SIG that behave exactly like $\mathcal{C}$, $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively.

Game G_1. Let game G_1 be the $\mathsf{EUF\text{-}CMA}^{\mathcal{K}}_{\mathsf{SIG}}$ game with the same winning condition as in the proof of Theorem 5. We have:

$$\Pr[\mathsf{G}_1 \Rightarrow 1] \geq \Pr[\mathsf{G}_0 \Rightarrow 1]. \tag{19}$$

Game G_2. We change the winning condition: $\mathcal{K}$ is successful in Game G_2 if $\mathsf{EUF\text{-}CMA}^{\mathcal{K}}_{\mathsf{SIG}}$ does not abort and if $(x^*, \widetilde{w}^*) \notin \widetilde{\mathcal{R}}$ for all non-witnesses of the form $\widetilde{w}^* = (\perp, \mathsf{dk}_r^*, \mathsf{dk}_j^*, \widetilde{\mathsf{dk}_j^*})$, with $x^* = (\mathsf{pk}_s, \mathsf{pk}_r^*, \mathsf{pk}_j, c_\sigma^*, \cdots, m_r^*)$ and $\mathsf{dk}_r^*, \mathsf{dk}_j^*, \widetilde{\mathsf{dk}_j^*} \in \mathcal{DK}$. Let $\mathcal{K}'$ (described below) be the adversary against the selective soundness of ZAP for some arbitrary fixed non-witness of the form $\widetilde{w} = (\perp, \mathsf{dk}_r, \mathsf{dk}_j, \widetilde{\mathsf{dk}_j})$. $\mathcal{K}'$ takes as input $\widetilde{w}$ and some ZAP randomness zr, then generates $\mathsf{ek}_r, \mathsf{ek}_j$ and $\widetilde{\mathsf{ek}_j}$ using $\mathsf{dk}_r, \mathsf{dk}_j$ and $\widetilde{\mathsf{dk}_j}$ respectively, and finally runs $\mathcal{K}$ using $\mathsf{ek}_r, \mathsf{dk}_r, \mathsf{ek}_j, \mathsf{dk}_j, \widetilde{\mathsf{ek}_j}, \widetilde{\mathsf{dk}_j}$ and $\mathsf{zr}_j := \mathsf{zr}$ instead of generating new ones in the setup phase. Like $\mathcal{C}'$, $\mathcal{K}'$ outputs the ZAP statement and proof (x^*, π^*) associated with the forgery of $\mathcal{A}$ if $\mathsf{EUF\text{-}CMA}^{\mathcal{K}}_{\mathsf{SIG}}$ terminates. A reduction to the selective soundness of ZAP yields that if $\widetilde{w} = (\perp, \mathsf{dk}_r, \mathsf{dk}_j, \widetilde{\mathsf{dk}_j})$ with $\mathsf{dk}_r, \mathsf{dk}_j, \widetilde{\mathsf{dk}_j} \leftarrow\!\!\$\ \mathsf{Dist}_{\mathcal{DK}}$, then

$$\begin{aligned}\Pr\Big[\mathsf{EUF\text{-}CMA}^{\mathcal{K}}_{\mathsf{SIG}} \text{ terminates} \wedge \exists\, \mathsf{dk}_r^*, \mathsf{dk}_j^*, \widetilde{\mathsf{dk}_j^*} \in \mathcal{DK} : (x^*, (\perp, \mathsf{dk}_r^*, \mathsf{dk}_j^*, \widetilde{\mathsf{dk}_j^*})) \in \widetilde{\mathcal{R}}\Big] \\ \leq \Pr\Big[\mathsf{SelSound}^{\mathcal{K}', \widetilde{w}}_{\mathsf{ZAP}} \Rightarrow 1\Big] + 3 \cdot \gamma_{\mathsf{KI}}\end{aligned}$$

provided $\mathcal{K}$ generated $\mathsf{dk}_r, \mathsf{dk}_j$ and $\widetilde{\mathsf{dk}_j}$ in the setup phase. Indeed, suppose that $\mathcal{A}$ manages to output a valid forgery with a ZAP statement and proof (x^*, π^*) such that there exists $\widetilde{w}^* = (\bot, \mathsf{dk}_r^*, \mathsf{dk}_j^*, \widetilde{\mathsf{dk}_j^*})$, with $\mathsf{dk}_r^*, \mathsf{dk}_j^*, \widetilde{\mathsf{dk}_j^*} \in \mathcal{DK}$ verifying $(x^*, \widetilde{w}^*) \in \widetilde{\mathcal{R}}$. Because of $(\widetilde{0'})$ in $\widetilde{\mathcal{R}}$, $(\mathsf{ek}_j, \mathsf{dk}_j^*)$, $(\mathsf{ek}_r, \mathsf{dk}_r^*)$ and $(\widetilde{\mathsf{ek}_j}, \widetilde{\mathsf{dk}_j^*})$ are well-formed. By injectivity of key generation of PKE, except with negligible probability $(3 \cdot \gamma_{\mathsf{KI}})$ over the choice of dk_r, dk_j and $\widetilde{\mathsf{dk}_j}$, we have $\mathsf{dk}_r^* = \mathsf{dk}_r$, $\mathsf{dk}_j^* = \mathsf{dk}_j$ and $\widetilde{\mathsf{dk}_j^*} = \widetilde{\mathsf{dk}_j}$ thus $\widetilde{w}^* = \widetilde{w}$ so $\mathcal{K}'$ is successful in the $\mathsf{SelSound}$ game for ZAP with fixed non witness $\widetilde{w}$. We can finally deduce that

$$0 \leq \Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1] \leq \mathsf{Adv}_{\mathsf{ZAP}}^{\mathsf{SelSound}}(\mathcal{K}') + 3 \cdot \gamma_{\mathsf{KI}}. \tag{20}$$

Game G_3. Let game G_3 be the *original* $\mathsf{EUF\text{-}CMA}_{\mathsf{SIG}}^{\mathcal{K}}$ game. If $\mathcal{K}$ wins G_2, then $\mathsf{Dec}(\mathsf{dk}_r, c_r^*) = 1$ while $\mathsf{Dec}(\mathsf{dk}_j, c_j^*) = 0$ and for the specific $\widetilde{w} = (\bot, \mathsf{dk}_r, \mathsf{dk}_j, \widetilde{\mathsf{dk}_j})$, with $\widetilde{\mathsf{dk}_s}$ and dk_j generated by $\mathcal{G}$ in the setup phase, it holds that $(x^*, \widetilde{w}) \notin \widetilde{\mathcal{R}}$. Therefore, one of the five equations of $(\widetilde{\star})$ is false. Equation $(\widetilde{0})$ is true because td_j is a genuine encryption of 0. Disjunctions $(\widetilde{2})$ is true because the clause $\mathsf{wf}_r \wedge \mathsf{Dec}(\mathsf{dk}_r, {c_r}^*) \neq 0$ is true. Disjunctions $(\widetilde{1})$ and $(\widetilde{4})$ are true because the clause $\mathsf{wf}_j \wedge \mathsf{Dec}(\mathsf{dk}_j, c_j^*) \neq 1$ is true. Therefore, Disjunction $(\widetilde{3})$ must be false, thus $\mathsf{SIG.Verify}(\mathsf{vk}_r, m_r^*, \mathsf{Dec}(\widetilde{\mathsf{dk}_j}, {c'_\sigma}^*)) = 1$ because $(\widetilde{\mathsf{ek}_j}, \widetilde{\mathsf{dk}_j})$ is well-formed. $\mathcal{K}$ never makes any signing query to the challenger so it is successful in the $\mathsf{EUF\text{-}CMA}$ forgery game.

$$\Pr[\mathsf{G}_3 \Rightarrow 1] \geq \Pr[\mathsf{G}_2 \Rightarrow 1]. \tag{21}$$

Combining Equations (19) to (21) yields the result stated in Theorem 14. □

From Theorems 3, 6 to 8, 13 and 14, we conclude the following corollary.

Corollary 3. *If* PKE *is perfectly correct,* $\mathsf{MC\text{-}IND\text{-}CPA}$ *secure,* $\mathsf{MC\text{-}IK\text{-}CPA}$ *secure and has injective key generation, if* SIG *is correct and* $\mathsf{EUF\text{-}CMA}$ *secure, and if* ZAP *for* $(\mathcal{L}, \widetilde{\mathcal{L}})$ *is correct, witness indistinguishable and selective non-witness and adaptive statement sound then the AMF scheme* $\mathsf{AMF}'_{\mathsf{GC}}$ *is a secure AMF scheme with strong deniability.*

6.4 Instantiation of $\mathsf{AMF}'_{\mathsf{GC}}$

$\mathsf{AMF}'_{\mathsf{GC}}$ gives the first post-quantum secure AMF scheme in the plain model from *polynomially-hard* LWE assumption by using the following concrete schemes:

- The perfectly correct version of Regev PKE scheme described in [CGH+21, Section 2.3]. We note that $\mathsf{WellFormed}_{\mathsf{PKE}}$ for Regev PKE scheme is identical to the Valid algorithm given in [CGH+21, Section 2.3].
- A standard signature scheme based on LWE assumption, e.g., [CHKP10, DM14].
- The statistical ZAP for $\mathsf{NP} \cap \mathsf{coNP}$ based on polynomially-hard LWE assumption [CGH+21].

Acknowledgments

We would like to thank the anonymous reviewers of Asiacrypt 2025, Eurocrypt 2026, PrivCrypt 2026, and Crypto 2026 for their helpful comments and suggestions. We thank Shota Yamada, Yusuke Sakai, Takahiro Matsuda, Keisuke Hara and Jacob Schuldt for helpful discussions on early stage of this work as well as Toi Tomita and Cecilia Boschini for their comments about properties of lattice-based PKE schemes. Also, we thank Dennis Hofheinz and Marian Dietz for their help in designing the efficient construction of AMF presented in Section 5. Finally, we would like to thank Damien Stehlé for providing us with the opportunity to initiate this work. Keitaro Hashimoto was partially supported by JST CREST JPMJCR22M1, Japan.

References

BBDP01. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT 2001*, *LNCS* 2248, pages 566–582. Springer, Berlin, Heidelberg, December 2001.

BBM00. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT 2000*, *LNCS* 1807, pages 259–274. Springer, Berlin, Heidelberg, May 2000.

BDH+19. M. Backes, N. Döttling, L. Hanzlik, K. Kluczniak, and J. Schneider. Ring signatures: Logarithmic-size, no setup - from standard assumptions. In *EUROCRYPT 2019, Part III*, *LNCS* 11478, pages 281–311. Springer, Cham, May 2019.

BFJ+20. S. Badrinarayanan, R. Fernando, A. Jain, D. Khurana, and A. Sahai. Statistical ZAP arguments. In *EUROCRYPT 2020, Part III*, *LNCS* 12107, pages 642–667. Springer, Cham, May 2020.

BHJ+19. B. Barak, S. B. Hopkins, A. Jain, P. Kothari, and A. Sahai. Sum-of-squares meets program obfuscation, revisited. In *EUROCRYPT 2019, Part I*, *LNCS* 11476, pages 226–250. Springer, Cham, May 2019.

BKM06. A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *TCC 2006*, *LNCS* 3876, pages 60–79. Springer, Berlin, Heidelberg, March 2006.

BP15. N. Bitansky and O. Paneth. ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In *TCC 2015, Part II*, *LNCS* 9015, pages 401–427. Springer, Berlin, Heidelberg, March 2015.

CGH+21. R. Chatterjee, S. Garg, M. Hajiabadi, D. Khurana, X. Liang, G. Malavolta, O. Pandey, and S. Shiehian. Compact ring signatures from learning with errors. In *CRYPTO 2021, Part I*, *LNCS* 12825, pages 282–312, Virtual Event, August 2021. Springer, Cham.

CH20a. G. Couteau and D. Hartmann. Shorter non-interactive zero-knowledge arguments and ZAPs for algebraic languages. In *CRYPTO 2020, Part III*, *LNCS* 12172, pages 768–798. Springer, Cham, August 2020.

CH20b. G. Couteau and D. Hartmann. Shorter non-interactive zero-knowledge arguments and ZAPs for algebraic languages. Cryptology ePrint Archive, Report 2020/286, 2020.

CHKP10. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT 2010*, *LNCS* 6110, pages 523–552. Springer, Berlin, Heidelberg, May / June 2010.

CHMR23. S. Chakraborty, D. Hofheinz, U. Maurer, and G. Rito. Deniable authentication when signing keys leak. In *EUROCRYPT 2023, Part III*, *LNCS* 14006, pages 69–100. Springer, Cham, April 2023.

CKSU21. G. Couteau, S. Katsumata, E. Sadeghi, and B. Ursu. Statistical ZAPs from group-based assumptions. In *TCC 2021, Part I*, *LNCS* 13042, pages 466–498. Springer, Cham, November 2021.

CL06. M. Chase and A. Lysyanskaya. On signatures of knowledge. In *CRYPTO 2006*, *LNCS* 4117, pages 78–96. Springer, Berlin, Heidelberg, August 2006.

CS97. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO'97*, *LNCS* 1294, pages 410–424. Springer, Berlin, Heidelberg, August 1997.

DGRW18. Y. Dodis, P. Grubbs, T. Ristenpart, and J. Woodage. Fast message franking: From invisible salamanders to encryptment. In *CRYPTO 2018, Part I*, *LNCS* 10991, pages 155–186. Springer, Cham, August 2018.

DH26. M. Dietz and D. Hofheinz. Signatures in the non-observable random oracle model. Cryptology ePrint Archive, Paper 2026/9999, 2026.

DHH+21. N. Döttling, D. Hartmann, D. Hofheinz, E. Kiltz, S. Schäge, and B. Ursu. On the impossibility of purely algebraic signatures. In *TCC 2021, Part III*, *LNCS* 13044, pages 317–349. Springer, Cham, November 2021.

DHM+20. I. Damgård, H. Haagh, R. Mercer, A. Nitulescu, C. Orlandi, and S. Yakoubov. Stronger security and constructions of multi-designated verifier signatures. In *TCC 2020, Part II*, *LNCS* 12551, pages 229–260. Springer, Cham, November 2020.

DM14. L. Ducas and D. Micciancio. Improved short lattice signatures in the standard model. In *CRYPTO 2014, Part I*, *LNCS* 8616, pages 335–352. Springer, Berlin, Heidelberg, August 2014.

DN00. C. Dwork and M. Naor. Zaps and their applications. In *41st FOCS*, pages 283–293. IEEE Computer Society Press, November 2000.

ElG85. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

Fac17. Facebook. Messenger secret conversations technical whitepaper. Facebook messenger app, 2017.

FL25. R. Fiedler and R. Langrehr. On deniable authentication against malicious verifiers. In *CRYPTO 2025, Part VIII*, *LNCS* 16007, pages 3–38. Springer, Cham, August 2025.

GJJM20. V. Goyal, A. Jain, Z. Jin, and G. Malavolta. Statistical zaps and new oblivious transfer protocols. In *EUROCRYPT 2020, Part III*, *LNCS* 12107, pages 668–699. Springer, Cham, May 2020.

GLR17. P. Grubbs, J. Lu, and T. Ristenpart. Message franking via committing authenticated encryption. In *CRYPTO 2017, Part III*, *LNCS* 10403, pages 66–97. Springer, Cham, August 2017.

GS08. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT 2008*, *LNCS* 4965, pages 415–432. Springer, Berlin, Heidelberg, April 2008.

HJ12. D. Hofheinz and T. Jager. Tightly secure signatures and public-key encryption. In *CRYPTO 2012*, *LNCS* 7417, pages 590–607. Springer, Berlin, Heidelberg, August 2012.

HK12. D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. *Journal of Cryptology*, 25(3):484–527, July 2012.

HLZW24. Z. Huang, J. Lai, G. Zeng, and J. Weng. Mild asymmetric message franking: Illegal-messages-only and retrospective content moderation. In *ASIACRYPT 2024, Part II*, *LNCS* 15485, pages 266–295. Springer, Singapore, December 2024.

HYH25. K. Hashimoto, K. Yamashita, and K. Hara. Foundations of Multi-Designated Verifier Signature: Comprehensive Formalization and New Constructions in Subset Simulation . In *2025 IEEE 38th Computer Security Foundations Symposium (CSF)*, pages 174–189, Los Alamitos, CA, USA, June 2025. IEEE Computer Society.

IAV22. R. Issa, N. Alhaddad, and M. Varia. Hecate: Abuse reporting in secure messengers with sealed sender. In *USENIX Security 2022*, pages 2335–2352. USENIX Association, August 2022.

JJ21. A. Jain and Z. Jin. Non-interactive zero knowledge from sub-exponential DDH. In *EUROCRYPT 2021, Part I*, *LNCS* 12696, pages 3–32. Springer, Cham, October 2021.

JSI96. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT'96*, *LNCS* 1070, pages 143–154. Springer, Berlin, Heidelberg, May 1996.

KPW15. E. Kiltz, J. Pan, and H. Wee. Structure-preserving signatures from standard assumptions, revisited. In *CRYPTO 2015, Part II*, *LNCS* 9216, pages 275–295. Springer, Berlin, Heidelberg, August 2015.

LVW19. A. Lombardi, V. Vaikuntanathan, and D. Wichs. 2-message publicly verifiable WI from (subexponential) LWE. Cryptology ePrint Archive, Report 2019/808, 2019.

LZH+23. J. Lai, G. Zeng, Z. Huang, S. M. Yiu, X. Mu, and J. Weng. Asymmetric group message franking: Definitions and constructions. In *EUROCRYPT 2023, Part V*, *LNCS* 14008, pages 67–97. Springer, Cham, April 2023.

MF21. A. Mittelbach and M. Fischlin. *The Theory of Hash Functions and Random Oracles: An Approach to Modern Cryptography*. Information Security and Cryptography. Springer Cham, 2021.

MPR22a. U. Maurer, C. Portmann, and G. Rito. Multi-designated receiver signed public key encryption. Cryptology ePrint Archive, Report 2022/256, 2022.

MPR22b. U. Maurer, C. Portmann, and G. Rito. Multi-designated receiver signed public key encryption. In *EUROCRYPT 2022, Part II*, *LNCS* 13276, pages 644–673. Springer, Cham, May / June 2022.

RST01. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT 2001*, *LNCS* 2248, pages 552–565. Springer, Berlin, Heidelberg, December 2001.

TGL+19a. N. Tyagi, P. Grubbs, J. Len, I. Miers, and T. Ristenpart. Asymmetric message franking: Content moderation for metadata-private end-to-end encryption. In *CRYPTO 2019, Part III*, *LNCS* 11694, pages 222–250. Springer, Cham, August 2019.

TGL+19b. N. Tyagi, P. Grubbs, J. Len, I. Miers, and T. Ristenpart. Asymmetric message franking: Content moderation for metadata-private end-to-end encryption. Cryptology ePrint Archive, Report 2019/565, 2019.

Wat05. B. R. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT 2005*, *LNCS* 3494, pages 114–127. Springer, Berlin, Heidelberg, May 2005.

A Missing Proofs for $\mathsf{AMF_{GC}}$ and $\mathsf{AMF_{EC}}$

In this Section, we recall and provide the proofs of Theorems 5, 7, 8, 11 and 12 and Lemma 2.

A.1 Sender Binding for $\mathsf{AMF_{GC}}$

Theorem 5 (Sender Binding). *If* SIG *is* EUF-CMA *secure, if* PKE *is perfectly correct and if* ZAP *is adaptively sound then the AMF scheme* $\mathsf{AMF_{GC}}$ *is sender binding. Furthermore, if* $\mathcal{A}$ *is an adversary against the sender binding security of* $\mathsf{AMF_{GC}}$*, then there exist adversaries* $\mathcal{C}$ *and* $\mathcal{C}'$ *against the* EUF-CMA *security of* SIG *and the adaptive soundness of* ZAP *such that*

$$\mathsf{Adv}^{\mathsf{s\text{-}Bind}}_{\mathsf{AMF_{GC}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{C}) + \mathsf{Adv}^{\mathsf{Sound}}_{\mathsf{ZAP}}(\mathcal{C}').$$

Note that the running time of $\mathcal{C}$ *and* $\mathcal{C}'$ *is about that of* $\mathcal{A}$*.*

Proof. Let $\mathcal{A}$ be an adversary against the receiver binding security of AMF. The proof of Theorem 5 is similar to the proof of Theorem 4, indeed the goal is to construct an adversary $\mathcal{C}$ against the EUF-CMA security of SIG which turns a forgery of $\mathcal{A}$ into a forgery for SIG, provided that the ZAP proof in the forgery was sound. Let game $\mathsf{G_0}$ be the original $\mathsf{s\text{-}Bind}^{\mathcal{A}}_{\mathsf{AMF_{GC}}}$ game.

Game $\mathsf{G_1}$. Let game $\mathsf{G_1}$ be the $\mathsf{EUF\text{-}CMA}^{\mathcal{C}}_{\mathsf{SIG}}$ game, (with $\mathcal{C}$ the adversary described below against the EUF-CMA security of SIG and using $\mathcal{A}$ as a black box) but with the same winning condition as in the proof of Theorem 4: $\mathcal{C}$ wins game $\mathsf{G_1}$ if $\mathsf{EUF\text{-}CMA}^{\mathcal{C}}_{\mathsf{SIG}}$ does not abort and if $\mathcal{C}$ produces a forgery attempt (independently of the game output). The following paragraphs describe the EUF-CMA adversary $\mathcal{C}$.

Setup. When receiving a verification key vk from the EUF-CMA challenger, $\mathcal{C}$ generates an incomplete *receiver* key pair $(\mathsf{pk}_r, \overline{\mathsf{sk}_r})$ the same way as $\mathcal{B}_1$. It also generates a full judge key pair $(\mathsf{pk}_j, \mathsf{sk}_j)$, and keeps $\widetilde{\mathsf{dk}_j}$. Finally, $\mathcal{C}$ sends the two public keys $(\mathsf{pk}_r, \mathsf{pk}_j)$ to $\mathcal{A}$.
Verify and Judge Queries. $\mathcal{C}$ can answer both Verify and Judge queries on its own since it knows the receiver and judge decryption keys.
Forgery. When receiving a forgery $(\mathsf{pk}^*_s, m^*, \Sigma^*)$ with $\Sigma^* = (\pi^*, c^*_\sigma, {c'_\sigma}^*, \cdots)$, $\mathcal{C}$ first checks that the Verify algorithm accepts it but not the Judge algorithm (it aborts if this is not the case). Then, $\mathcal{C}$ sets ${\sigma'}^*$ as the decryption of ${c'_\sigma}^*$ using $\widetilde{\mathsf{dk}_j}$ and sends $({\sigma'}^*, m^*_r)$ to the EUF-CMA game. First, remark that all public keys and answers to $\mathcal{A}$'s queries sent by $\mathcal{C}$ are genuine and therefore indistinguishable from an interaction with the real s-Bind game. Then, notice that if a forgery Σ^* produced by $\mathcal{A}$ results in winning game $\mathsf{G_0}$ then this forgery was accepted by Verify but not by Judge and therefore would not make $\mathcal{C}$ abort. Since SIG.Verify never aborts, $\mathsf{EUF\text{-}CMA}^{\mathcal{C}}_{\mathsf{SIG}}$ would not abort in the same context. From this, we deduce that

$$\Pr[\mathsf{G_1} \Rightarrow 1] \geq \Pr[\mathsf{G_0} \Rightarrow 1]. \tag{22}$$

Game $\mathsf{G_2}$. Let game $\mathsf{G_2}$ be similar to game $\mathsf{G_1}$ but with the following change in the winning condition: $\mathcal{C}$ is successful if $\mathsf{EUF\text{-}CMA}^{\mathcal{C}}_{\mathsf{SIG}}$ does not abort, if $\mathcal{C}$ produces a forgery attempt $\Sigma^* = (\pi^*, c^*_\sigma, \cdots, {c_j}^*)$, public key pk^*_s and message m^* *and if the statement* $x^* = (\mathsf{pk}^*_s, \mathsf{pk}_r, \mathsf{pk}_j, c^*_\sigma, \cdots, m^* || \mathsf{pk}_r)$ *is in* $\mathcal{L}$ with the same notations as before (recall that in the s-Bind game, $\mathcal{A}$ receives pk_r and outputs some pk^*_s). A reduction to the adaptive soundness of ZAP yields

$$0 \leq \Pr[\mathsf{G_1} \Rightarrow 1] - \Pr[\mathsf{G_2} \Rightarrow 1] \leq \mathsf{Adv}^{\mathsf{Sound}}_{\mathsf{ZAP}}(\mathcal{C}'). \tag{23}$$

for a suitable ZAP soundness adversary $\mathcal{C}'$.

Game $\mathsf{G_3}$. Let game $\mathsf{G_3}$ be the *original* $\mathsf{EUF\text{-}CMA}^{\mathcal{C}}_{\mathsf{SIG}}$ game. If $\mathcal{C}$ wins $\mathsf{G_2}$, then its forgery Σ^* has been accepted by Verify but not by Judge (thus c^*_r decrypts to 1 while c^*_j decrypts to 0) and $x^* = (\mathsf{pk}^*_s, \mathsf{pk}_r, \mathsf{pk}_j, c^*_\sigma, \cdots, m^* || \mathsf{pk}_r) \in \mathcal{L}$. Because td_j encrypts 0 and because of the perfect correctness of PKE, Conjunction (3) must be true. Finally, for this conjunction, the first clause requires that $\mathsf{SIG.Verify}(\mathsf{vk}_r, m^*_r, {\sigma'}^*) = 1$, making $\mathcal{C}$ successful in the EUF-CMA forgery game because vk_r was the verification key initially sent by the challenger and because no signing queries were made to him. Therefore, the probability of success is at least:

$$\Pr[\mathsf{G_3} \Rightarrow 1] \geq \Pr[\mathsf{G_2} \Rightarrow 1] \tag{24}$$

Combining Equations (22) to (24) yields the result stated in Theorem 5. □

A.2 Receiver Compromise Deniability for $\mathsf{AMF_{GC}}$

Theorem 7 (Receiver Compromise Deniability). *If* SIG *is correct, if* PKE *is* MC-IND-CPA *and* MC-IK-CPA *secure, and if* ZAP *is statistically witness indistinguishable then the AMF scheme* $\mathsf{AMF_{GC}}$ *achieves receiver compromise deniability. Furthermore, if* $\mathcal{A}$ *is an adversary against the receiver compromise deniability security of* $\mathsf{AMF_{GC}}$*, then there exist adversaries* $\mathcal{E}$ *and* $\mathcal{E}'$ *against the* MC-IND-CPA *and* MC-IK-CPA *security of* PKE *such that*

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{RecCompDen}}_{\mathsf{AMF_{GC}}}(\mathcal{A})\\ &\quad\leq 5 \cdot \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{E}) + \mathsf{Adv}^{\mathsf{MC\text{-}IK\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{E}') + 2Q \cdot \varepsilon_{\mathsf{WI}} + 2Q \cdot \varepsilon_{\mathsf{SIG}}\end{aligned}$$

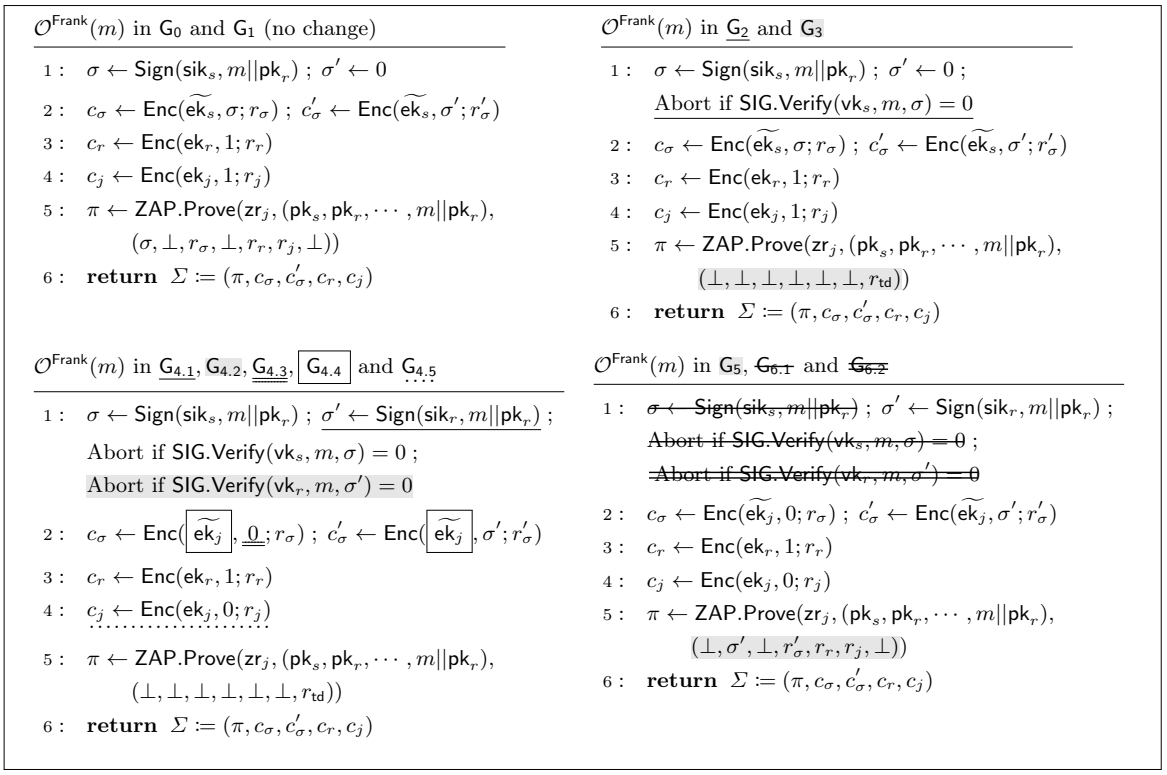


Fig. 10: Changes in the behavior of $\mathcal{O}^{\mathsf{Frank}}$ during the receiver compromise deniability proof. The first and last game hops are not represented since the updates occur in the key generation phase.

with Q the number of Frank *queries of $\mathcal{A}$ and $\varepsilon_{\mathsf{WI}}$ (resp. $\varepsilon_{\mathsf{SIG}}$) related to the statistical witness-indistinguishability of* ZAP *(resp. correctness error of* SIG*). Note that the running time of $\mathcal{E}$ and $\mathcal{E}'$ is about that of $\mathcal{A}$.*

Proof. The proof of Theorem 7 is very similar to the proof of Theorem 6. Figure 10 describes the behavior of the Frank oracle between game G_1 and game $\mathsf{G}_{6.2}$ (again, the first and last game hops are not represented since the updates occur in the key generation phase). We only justify the first hop, last hop and the hop from game $\mathsf{G}_{4.3}$ to $\mathsf{G}_{4.4}$ because the justification to all the other hops can be deduced from arguments that appear in the proofs of Theorems 6 and 8.

Game G_0. This is the genuine $\mathsf{RecCompDen}^{\mathcal{A},0}_{\mathsf{AMF}_{\mathsf{GC}}}$ security game: $\mathcal{O}^{\mathsf{Frank}}$ uses the real Frank algorithm to answer queries.

Game G_1. In game G_1, the judge key pair is generated with the trapdoor open: instead of encrypting 0 in td_j, the challenger encrypts 1 using randomness r_{td}. A reduction to the MC-IND-CPA security of PKE yields

$$|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_0 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{E}_1)$$

for a suitable MC-IND-CPA adversary $\mathcal{E}_1$.

Game $\mathsf{G}_{4.4}$. In game $\mathsf{G}_{4.4}$, $\mathcal{O}^{\mathsf{Frank}}$ swaps the encryption key used to produce both c_σ and c'_σ from the sender to the judge orphan encryption key. A reduction to the MC-IK-CPA security of PKE yields

$$|\Pr[\mathsf{G}_{4.4} \Rightarrow 1] - \Pr[\mathsf{G}_{4.3} \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{MC\text{-}IK\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{E}'_{4.4})$$

for a suitable MC-IK-CPA adversary $\mathcal{E}'_{4.4}$.

Game G_7. In game G_7, the judge key pair is generated with td_j encrypting 0 again (trapdoor closed). G_7 perfectly describes $\mathsf{RecCompDen}^{\mathcal{A},1}_{\mathsf{AMF}_{\mathsf{GC}}}$. A reduction to the MC-IND-CPA security of PKE yields

$$|\Pr[\mathsf{G}_7 \Rightarrow 1] - \Pr[\mathsf{G}_{6.2} \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{E}_7)$$

for a suitable MC-IND-CPA adversary $\mathcal{E}_7$. □

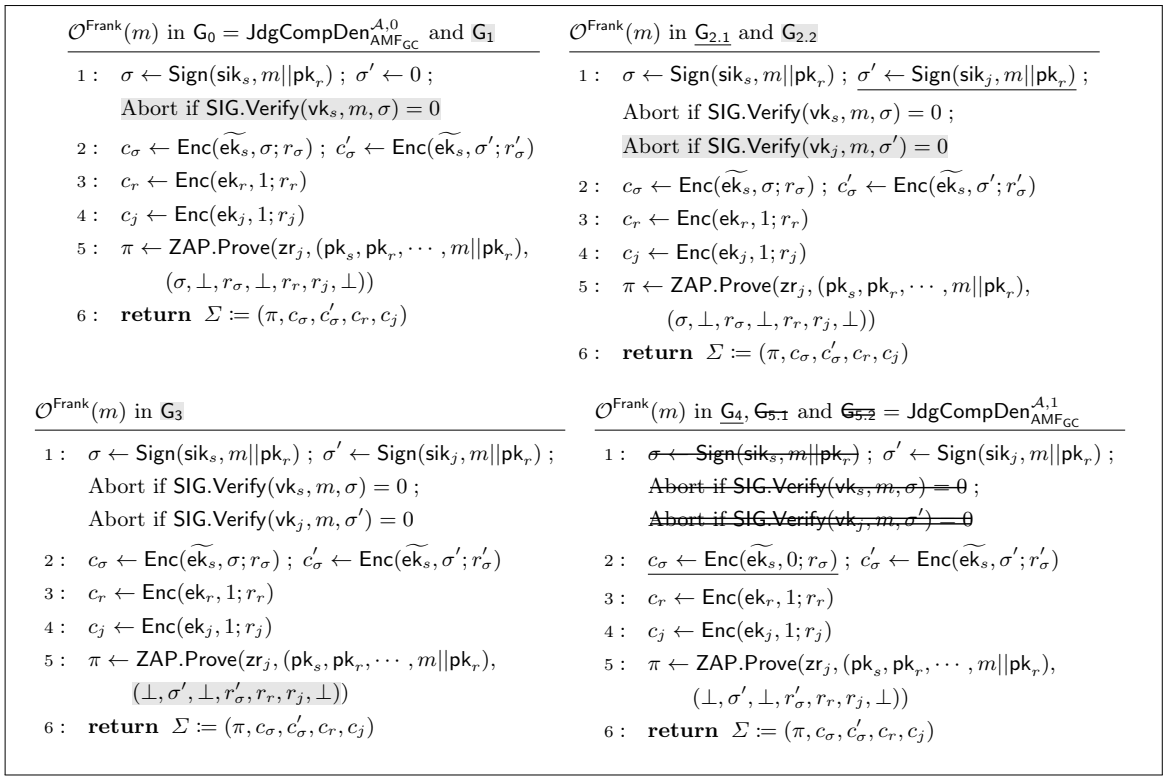


Fig. 11: Games for judge compromise deniability.

A.3 Judge Compromise Deniability for $\mathsf{AMF_{GC}}$

Theorem 8 (Judge Compromise Deniability). *If* SIG *is correct, if* PKE *is* MC-IND-CPA *secure and if* ZAP *is statistically witness indistinguishable then the AMF scheme* $\mathsf{AMF_{GC}}$ *achieves judge compromise deniability. Furthermore, if* $\mathcal{A}$ *is an adversary against the judge compromise deniability security of* $\mathsf{AMF_{GC}}$*, there exists an adversary* $\mathcal{F}$ *against the* MC-IND-CPA *security of* PKE *such that*

$$\mathsf{Adv}^{\mathsf{JdgCompDen}}_{\mathsf{AMF_{GC}}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{F}) + Q \cdot \varepsilon_{\mathsf{WI}} + 2Q \cdot \varepsilon_{\mathsf{SIG}}$$

with Q *the number of* Frank *queries of* $\mathcal{A}$ *and* $\varepsilon_{\mathsf{WI}}$ *(resp.* $\varepsilon_{\mathsf{SIG}}$*) related to the statistical witness-indistinguishability of* ZAP *(resp. correctness error of* SIG*). Note that the running time of* $\mathcal{F}$ *is about that of* $\mathcal{A}$*.*

Proof. The proof of Theorem 8 is a bit different from the proof of Theorem 6 because the judge public key is not generated by the challenger, therefore the trapdoor td_j can not be used to swap witnesses. However, remark that both c_r and c_j encrypt 1 in the output of both Frank and Judge algorithms, thus only the encryption of signatures have to be updated, using the c'_σ to encrypt the judge signature *before* swapping witnesses. This is why our scheme duplicates the ciphertext for signatures c_σ. Figure 11 describes the behavior of the Frank oracle between game $\mathsf{G}_0 = \mathsf{JdgCompDen}^{\mathcal{A},0}_{\mathsf{AMF_{GC}}}$ and game $\mathsf{G}_{5.2} = \mathsf{JdgCompDen}^{\mathcal{A},1}_{\mathsf{AMF_{GC}}}$ (the rest of the experiment remains unchanged, including key generation).

Game G_0. This is the genuine $\mathsf{JdgCompDen}^{\mathcal{A},0}_{\mathsf{AMF_{GC}}}$ security game: $\mathcal{O}^{\mathsf{Frank}}$ uses the real Frank algorithm to answer queries.

Game G_1. In game G_1, $\mathcal{O}^{\mathsf{Frank}}$ aborts as soon as it produces an invalid signature for verification key vk_s. This change is purely conceptual since in this case, the oracle would have aborted when trying to generate the ZAP proof π.

$$\Pr[\mathsf{G}_1 \Rightarrow 1] = \Pr[\mathsf{G}_0 \Rightarrow 1]$$

Games $\mathsf{G}_{2.1}$, $\mathsf{G}_{2.2}$. In game $\mathsf{G}_{2.1}$, $\mathcal{O}^{\mathsf{Frank}}$ produces a signature for $m||\mathsf{pk}_r$ using the judge signing key, and updates c'_σ. A reduction to the MC-IND-CPA security of PKE yields

$$|\Pr[\mathsf{G}_{2.1} \Rightarrow 1] - \Pr[\mathsf{G}_1 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{F}_{2.1})$$

for a suitable MC-IND-CPA adversary $\mathcal{F}_{2.1}$. In game $\mathsf{G}_{2.2}$, $\mathcal{O}^{\mathsf{Frank}}$ now aborts if the aforementioned signature turns out to be invalid for the verification key vk_s, which happens with negligible probability since SIG is correct:

$$|\Pr[\mathsf{G}_{2.2} \Rightarrow 1] - \Pr[\mathsf{G}_{2.1} \Rightarrow 1]| \leq Q \cdot \varepsilon_{\mathsf{SIG}}$$

with Q the number of Frank queries and $\varepsilon_{\mathsf{SIG}}$ the correctness error of SIG.

Game G_3. In game G_3, $\mathcal{O}^{\mathsf{Frank}}$ changes the witness of the ZAP proof to use prime variables. Both witnesses are correct for the statement with respect to $\mathcal{R}$ so a reduction to the statistical witness-indistinguishability of ZAP yields:

$$|\Pr[\mathsf{G}_3 \Rightarrow 1] - \Pr[\mathsf{G}_{2.2} \Rightarrow 1]| \leq Q \cdot \varepsilon_{\mathsf{WI}}$$

with Q the number of Frank queries of $\mathcal{A}$ and $\varepsilon_{\mathsf{WI}}$ related to the statistical witness-indistinguishability of ZAP.

Game G_4. In game G_4, $\mathcal{O}^{\mathsf{Frank}}$ changes the value of the plaintext inside c_σ while still producing a signature using sik_s and aborting in case it is invalid. A reduction to the MC-IND-CPA security of PKE yields

$$|\Pr[\mathsf{G}_4 \Rightarrow 1] - \Pr[\mathsf{G}_3 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{MC\text{-}IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{F}_4)$$

for a suitable MC-IND-CPA adversary $\mathcal{F}_4$.

Games $\mathsf{G}_{5.1}$, $\mathsf{G}_{5.2}$. In game $\mathsf{G}_{5.1}$, $\mathcal{O}^{\mathsf{Frank}}$ stops producing σ and checking its validity. The output of the game changes only when an invalid signature was produced in G_4, which happens with negligible probability since SIG is correct:

$$|\Pr[\mathsf{G}_{5.1} \Rightarrow 1] - \Pr[\mathsf{G}_4 \Rightarrow 1]| \leq Q \cdot \varepsilon_{\mathsf{SIG}}$$

with Q the number of Frank queries and $\varepsilon_{\mathsf{SIG}}$ the correctness error of SIG. In game $\mathsf{G}_{5.2}$, $\mathcal{O}^{\mathsf{Frank}}$ stops checking the validity of σ'. This change is purely conceptual since if σ' is invalid, then the oracle would abort when trying to generate the ZAP proof π.

$$\Pr[\mathsf{G}_{5.2} \Rightarrow 1] = \Pr[\mathsf{G}_{5.1} \Rightarrow 1]$$

Collecting all differences of probabilities yields the result stated in Theorem 8. □

A.4 Probability That the Challenger Aborts in Game $\mathsf{G}_{2.1}$ (Receiver Binding of $\mathsf{AMF}_{\mathsf{EC}}$)

Lemma 2. *If $(Q^2 \cdot k)$ is much smaller than p, the probability that the challenger aborts at the end of game $\mathsf{G}_{2.1}$ (provided that it produced a valid forgery for a message and a public key with a hash never seen before) is smaller than $(1 - \frac{1}{Q\sqrt{k}})$.*

Proof. In the proof of [HK12, Theorem 6], Hofheinz and Kiltz show that for a fixed $q = q(\lambda)$ and group $\mathbb{G}$ with known order, some hash function H^{MG} is a $(1, q)$-programmable hash function[14]. To do so, they show (in particular) that the two following PPT algorithms PHF.TrapGen and PHF.TrapEval have *well-distributed logarithms*.

- $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$: Choose uniformly and independently $a_{i,j} \in \{-1, 0, 1\}$ for $1 \leq i \leq l$ and $1 \leq J$, as well as random group exponents $b_0, \cdots, b_k$. Then set [15] $a_i \leftarrow \sum_{j=1}^{J} a_{i,j}$, $h_0 \leftarrow g^{-1} h^{b_0}$ and $h_i \leftarrow g_{a_i} h^{b_i}$ for all i. Finally return $K' \coloneqq (h_0, \cdots, h_k)$ and $t \coloneqq (b_0, a_1, b_1, \cdots, a_k, b_k)$.
- $\mathsf{PHF.TrapEval}(t, X)$: Parse $X = (x_1, \cdot, x_k) \in \{0, 1\}^k$ and return $a \coloneqq -1 + \sum_{i=1}^{k} a_i x_i$ and $b \coloneqq b_0 + \sum_{i=1}^{k} b_i x_i$.

Namely, for all generators $g, h \in \mathbb{G}$ and all possible K' in the range of the first component of $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$, for all $X, Z_1, \cdots, Z_q \in \{0, 1\}^k$ such that $X \neq Z_i$ for any i, and for the corresponding $(a_X, b_X) \leftarrow \mathsf{PHF.TrapEval}(t, X)$ and $(a_{Z_i}, b_{Z_i}) \leftarrow \mathsf{PHF.TrapEval}(t, Z_i)$, the following holds

$$\Pr\left[a_X = 0 \wedge a_{Z_i}, \cdots, a_{z_q} \neq 0\right] \geq \frac{1}{q\sqrt{k}}$$

[14] The precise definition of programmable hash functions is not usefull here.

[15] We use multiplicative notation exceptionnaly to stick to the original statement.

if $(q^2 \cdot l)$ is much smaller than the group order[16], where the probability is taken over the trapdoor t that was produced along with K'.
Now set $\mathbb{G} \coloneqq \mathbb{G}_1$ of known order p, $g \coloneqq g_1$, $h \coloneqq g_1$ (or any other generator), $l \coloneqq k$ and $q \coloneqq Q$ the maximum number of Frank queries. Remark that the challenger of Game $\mathsf{G}_{2.1}$ chooses the $b^{(i)}$ in the same way as $\mathsf{PHF.TrapGen}$ the a_i (given that $a_0 = -1$) ; and that $b^{(h)}$ (resp. $b^{(h^*)}$) are computed the in the same way as $\mathsf{PHF.TrapEval}(t, h)$ (resp. $\mathsf{PHF.TrapEval}(t, h^*)$). Finally, recall that the $b^{(i)}$ are perferctly hidden from the adversary (the challenger does not send them yet) and that since Game G_1, the hash h^* is different from all previous hash h, so the probability that the all the $b^{(h)}$ (at most Q) are non zero and that $b^{(h^*)} = 0$ is greater than $\frac{1}{Q\sqrt{k}}$, which concludes the proof. □

A.5 Sender Binding for $\mathsf{AMF_{EC}}$

Theorem 11 (Sender Binding). *If* DDH *assumption holds in* $\mathbb{G}_1$ *and if the* $\mathcal{L}_1$-$\mathsf{19\text{-}ExtKerMDH}$ *assumption holds in* $\mathbb{G}_2$ *then the AMF scheme* $\mathsf{AMF_{EC}}$ *is sender binding. More precisely, if* $\mathcal{A}$ *is an adversary against the sender binding security of* $\mathsf{AMF_{EC}}$ *then there exist adversaries* $\mathcal{H}$ *and* $\mathcal{H}'$ *against the* DDH *assumption in* $\mathbb{G}_1$ *and the* $\mathcal{L}_1$-$\mathsf{19\text{-}ExtKerMDH}$ *assumption in* $\mathbb{G}_2$ *such that*

$$\mathsf{Adv}^{\mathsf{s\text{-}Bind}}_{\mathsf{AMF_{EC}}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\mathcal{H}) + \mathsf{Adv}^{\mathsf{ExtKerMDH}}_{\mathbb{G}_2}(\mathcal{H}')$$

Note that the running time of $\mathcal{H}$*, and* $\mathcal{H}'$ *is about that of* $\mathcal{A}$*.*

Proof. Similarly as before, the goal of the proof is to start from the original $\mathsf{s\text{-}Bind}$ game and to drift toward a game in which the adversary produces a valid proof for a statement not in $\mathcal{L}'$. In this proof, $\mathcal{A}$ is an adversary against the $\mathsf{s\text{-}Bind}$ security of $\mathsf{AMF_{EC}}$ and we call *challenger* the PPT algorithm that interact with it.

Game G_0. Let game G_0 be the original $\mathsf{s\text{-}Bind}$ game. The receiver and judge key pairs are generated honestly, in particular the two public keys only contain encryptions of 0 and 1, and td_j encrypts 0. The oracles answer Verify and Judge queries using the genuine algorithms.

$$\Pr[\mathsf{G}_0 \Rightarrow 1] = \Pr\left[\mathsf{s\text{-}Bind}^{\mathcal{A}}_{\mathsf{AMF_{EC}}} \Rightarrow 1\right] \tag{25}$$

Game G_1. In game G_1, the encrypted plaintext $a_r^{(0)}$ in $C_r^{(0)}$ is swapped from 1 to 0 during key generation. In other words, the receiver public key now only contains encryptions of 0. Remark that the receiver keys are only used by the Verify oracle which solely uses the receiver decryption key dk_s. A reduction to the $\mathsf{MC\text{-}IND\text{-}CPA}$ security of ElGamal PKE yields

$$|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_0 \Rightarrow 1]| \leq 2 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\mathcal{H}) \tag{26}$$

for a suitable DDH adversary $\mathcal{H}$.

Conclusion. In game G_1 the adversary sends a valid forgery Σ^* for (pk_s^*, m^*) accepted by Verify but rejected by Judge. However, we have

$$\left(\mathsf{ek}_s^*, \mathsf{ek}_r, \mathsf{ek}_j, \mathsf{td}_j, (C_s^{(i)^*}, C_r^{(i)}, C_j^{(i)})_{0 \leq i \leq k}, c_r^*, c_j^*, h^*\right) \notin \mathcal{L}'$$

since:

- td_j is a genuine encryption of 0 so Equation $(0')$ must be false.
- $\mathsf{Dec}(\mathsf{dk}_r, {c_r}^*) = 1$, so there is no ${r_r}^* \in \mathbb{Z}_p$ such that ${c_r}^* = \mathsf{Enc}(\mathsf{ek}_r, 0; {r_r}^*)$ and therefore Conjunction $(2')$ must be false.
- $\mathsf{Dec}(\mathsf{dk}_j, {c_j}^*) = 0$, so there is no ${r_j}^* \in \mathbb{Z}_p$ such that ${c_j}^* = \mathsf{Enc}(\mathsf{ek}_j, 1; {r_j}^*)$ and therefore Conjunctions $(1')$ and $(4')$ must be false.
- $\forall i \in [0, k], a_r^{(i)} = 0$, so there is no $(\alpha^*, t^*) \in \mathbb{Z}_p^2$ such that

$$\left(C_r^{(0)} + \sum_{h^*[i]=1} C_r^{(i)}\right) \cdot \alpha^* = \mathsf{Enc}(\mathsf{ek}_r, 1; t^*).$$

[16] This bound is not specifically required in the statement of the Theorem but prevents the a_i from circling the entire $\mathbb{Z}_p$, and is satisfied in most relevant scenarios.

A reduction to the computational soundness of ZAP yields

$$\Pr[\mathsf{G}_1 \Rightarrow 1] \leq \mathsf{Adv}^{\mathsf{ExtKerMDH}}_{\mathbb{G}_2}(\mathcal{H}') \tag{27}$$

for a suitable $\mathcal{L}_1$-19-ExtKerMDH adversary $\mathcal{H}'$.

Combining Equations (25) to (27) yields the result stated in Theorem 11. □

A.6 Deniability Properties for $\mathsf{AMF}_{\mathsf{EC}}$

Theorem 12 (Deniability Properties). *If the* DDH *assumption holds in* $\mathbb{G}_1$*, then the AMF scheme* $\mathsf{AMF}_{\mathsf{EC}}$ *achieves universal deniability, and receiver compromise deniability. Furthermore, if* $\mathcal{A}$ *is an adversary against the universal deniability (resp. receiver compromise deniability) security of* $\mathsf{AMF}_{\mathsf{EC}}$*, then there exist adversaries* $\mathcal{I}$ *(resp.* $\mathcal{I}'$*) against the* DDH *assumption in* $\mathbb{G}_1$ *such that*

$$\mathsf{Adv}^{\mathsf{UnivDen}}_{\mathsf{AMF}_{\mathsf{EC}}}(\mathcal{A}) \leq 8 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\mathcal{I}) \quad \left(\textit{resp. } \mathsf{Adv}^{\mathsf{RecCompDen}}_{\mathsf{AMF}_{\mathsf{EC}}}(\mathcal{A}) \leq 6 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\mathcal{I}')\right)$$

Note that the running time of $\mathcal{I}$ *and* $\mathcal{I}'$ *is about that of* $\mathcal{A}$*.*

Moreover, $\mathsf{AMF}_{\mathsf{EC}}$ *achieves unconditional judge compromise deniability (i.e.,* $\mathsf{Adv}^{\mathsf{JdgCompDen}}_{\mathsf{AMF}_{\mathsf{EC}}}(\mathcal{A}) = 0$*).*

Proof. The unconditional judge compromise deniability comes from the fact that the only game hop necessary is a swap of witness in the ZAP proof, which is perfectly witness-indistinguishable.

The security proofs for the other deniability properties of $\mathsf{AMF}_{\mathsf{EC}}$ are very similar to the ones of $\mathsf{AMF}_{\mathsf{GC}}$, so we only justify receiver compromise deniability here.

Let $\mathcal{A}$ be an adversary against the receiver compromise deniability property of $\mathsf{AMF}_{\mathsf{EC}}$. The goal of the proof is to go from $\mathsf{G}_0 := \mathsf{RecCompDen}^{\mathcal{A},0}_{\mathsf{AMF}_{\mathsf{EC}}}$ (reply to queries with the Frank algorithm) to the $\mathsf{G}_5 := \mathsf{RecCompDen}^{\mathcal{A},1}_{\mathsf{AMF}_{\mathsf{EC}}}$ (reply with the RForge algorithm) with little difference in the success probability of $\mathcal{A}$ at every step.

Game G_1. In game G_1, the judge key pair is generated with the trapdoor open: instead of encrypting 0 in td_j, the challenger encrypts 1 using randomness r_{td}. A reduction to the MC-IND-CPA security of ElGamal PKE yields

$$|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_0 \Rightarrow 1]| \leq 2 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\mathcal{I}'_1) \tag{28}$$

for a suitable DDH adversary $\mathcal{I}'_1$.

Game G_2. In game G_2, the Frank queries are answered using the trapdoor witness r_{td}. Because ZAP is perfectly witness-indistinguishable,

$$\Pr[\mathsf{G}_2 \Rightarrow 1] = \Pr[\mathsf{G}_1 \Rightarrow 1]. \tag{29}$$

Game G_3. In game G_3, the plaintext inside c_j is swapped from 1 to 0, but the ZAP proof is still computed using the trapdoor witness r_{td}. A reduction to the MC-IND-CPA security of ElGamal PKE yields

$$|\Pr[\mathsf{G}_3 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1]| \leq 2 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\mathcal{I}'_3) \tag{30}$$

for a suitable DDH adversary $\mathcal{I}'_3$.

Game G_4. In game G_4, the Frank queries are answered using the witness derived from the receiver secret key since the later passed the well-formed check and since c_j now encrypts 0. Because ZAP is perfectly witness-indistinguishable,

$$\Pr[\mathsf{G}_4 \Rightarrow 1] = \Pr[\mathsf{G}_3 \Rightarrow 1]. \tag{31}$$

Game G_5. In game G_5, the judge key pair is generated with td_j encrypting 0 again. A reduction to the MC-IND-CPA security of ElGamal PKE yields

$$|\Pr[\mathsf{G}_5 \Rightarrow 1] - \Pr[\mathsf{G}_4 \Rightarrow 1]| \leq 2 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\mathcal{I}'_5) \tag{32}$$

for a suitable DDH adversary $\mathcal{I}'_5$.

Collecting all differences of probabilities yields the result stated in Theorem 12. □

B Other Security Notions for AMF

This section summarizes the other security notions for AMF schemes.

B.1 AMF with Strong Authentication

In addition to AMF with strong deniability, Tyagi et al. considered AMF with strong authentication [TGL+19b, Appendix B]. To provide a stronger authenticity property, they define *strong unforgeability*, which guarantees signatures are unforgeable even if the judge's secret key is known to the adversary. However, such a strong unforgeability contradicts judge compromise deniability as they mentioned. Thus, they introduced new deniability notions called *weak judge compromise deniability* and *judge-receiver compromise deniability*, which are weaker than judge compromise deniability. The formal definition of them are stated below.

Strong unforgeability prevents the judge from impersonating a sender to a receiver. Note that strong unforgeability is not implied by receiver binding and sender binding.

Definition 27 (Strong Unforgeability). *We say that an AMF scheme* AMF *is* strong unforgeable*, if for any efficient adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{sUnforge}}_{\mathsf{AMF}}(\mathcal{A}) := \Pr\left[\mathsf{sUnforge}^{\mathcal{A}}_{\mathsf{AMF}} \Rightarrow 1\right] \leq \mathsf{negl}(\lambda),$$

where the game $\mathsf{sUnforge}^{\mathcal{A}}_{\mathsf{AMF}}$ *is defined in the top of Figure 12.*

Remark 6. Strictly speaking, Tyagi et al. did not give a formal definition of strong authentication property. In [TGL+19b, Appendix B], they said "Strong authentication is a stronger unforgeability notion ensuring that the receiver will not accept judge forgeries. Informally, it can be considered as an extension of the unforgeability definition in which the adversary knows the judge secret key." Regarding the statement "the adversary knows the judge secret key," there are two possible scenarios: either the adversary generates the secret key itself, or an honestly-generated secret key is leaked to the adversary. The subsequent work by Huang et al. [HLZW24] has adopted the latter scenario. In this paper, we adopt the former because it gives strong security guarantee.

To make deniability notions compatible with strong unforgeability, Tyagi et al. weakens the judge compromise deniability by giving less secret keys to the distinguisher or more to the forging algorithm, resulting in two deniability notions. The first notion is weak judge compromise deniability. It indicates that parties who know the judge's secret key can forge a signature, indistinguishable from honestly-generated signature against the distinguisher who knows the sender's and the receiver's secret keys only. Since the distinguisher does not know the judge's secret key, this judge compromise deniability is weaker than Definition 24.

Definition 28 (Weak Judge Compromise Deniability). *We say that an AMF scheme* AMF *achieves* weak judge compromise deniability*, if for any efficient adversary* $\mathcal{A}$*, it holds that*

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{wJdgCompDen}}_{\mathsf{AMF}}(\mathcal{A})\\ &:= \left|\Pr\left[\mathsf{wJdgCompDen}^{\mathcal{A},0}_{\mathsf{AMF}} \Rightarrow 1\right] - \Pr\left[\mathsf{wJdgCompDen}^{\mathcal{A},1}_{\mathsf{AMF}} \Rightarrow 1\right]\right|\\ &\leq \mathsf{negl}(\lambda),\end{aligned}$$

where the game $\mathsf{wJdgCompDen}^{\mathcal{A},b}_{\mathsf{AMF}}$ *is defined in the bottom left of Figure 12.*

The second notion is judge-receiver compromise deniability. It indicates that parties who know both the receiver's and judge's secret keys can forge a signature, indistinguishable from honestly-generated signature to a distinguisher who knows the sender's, the receiver's and the judge's secret keys. Formally, the following new forging algorithm is introduced.

- $\mathsf{JRForge}(\mathsf{pk}_s, \mathsf{sk}_r, \mathsf{sk}_j, \mathsf{m}) \to \Sigma$: The probabilistic $\mathsf{JRForge}$ algorithm takes a judge secret key sk_j, a receiver secret key sk_r and sender public key pk_s and a message m as input and outputs a signature Σ.

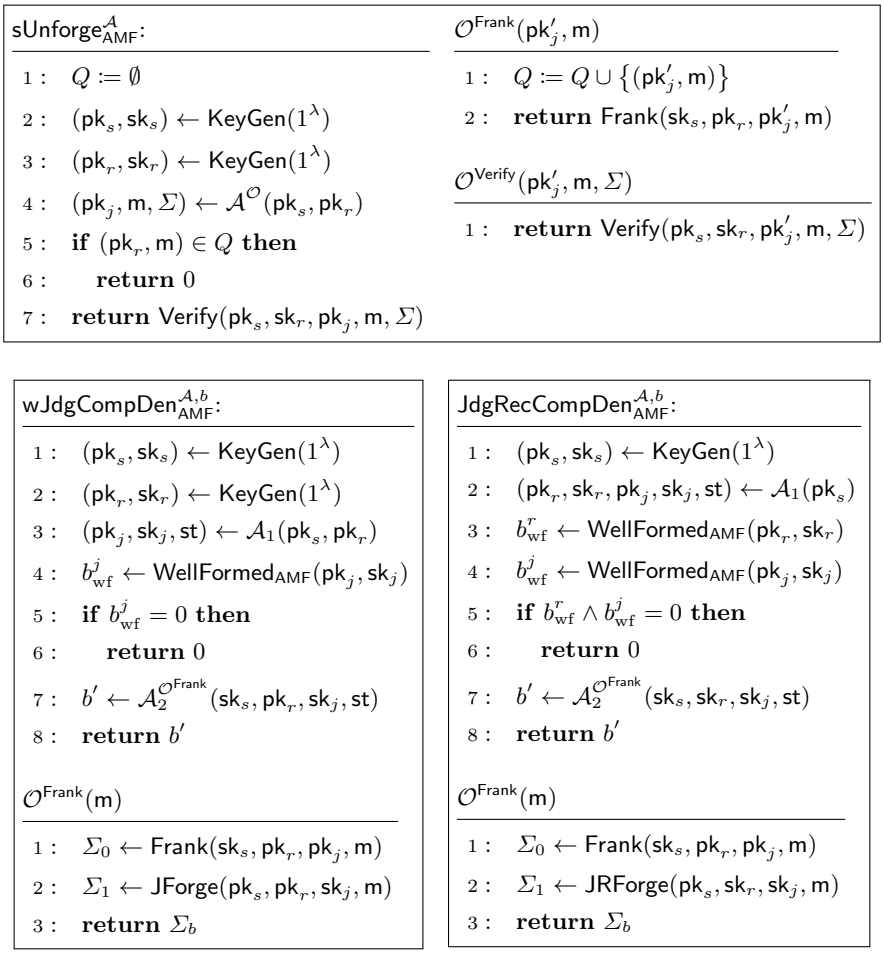


Fig. 12: Security games for AMF schemes with strong authentication: strong unforgeability (top), weak judge compromise deniability (bottom left), and judge-receiver compromise deniability (bottom right).

Judge-receiver compromise deniability is weaker than Definition 22 because the forgery algorithm for judge-receiver compromise deniability uses both the receiver's and the judge's secret keys.

Definition 29 (Judge-Receiver Compromise Deniability). *We say that an AMF scheme* AMF *achieves* judge-receiver compromise deniability*, if for any efficient adversary $\mathcal{A}$, it holds that*

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{JdgRecCompDen}}_{\mathsf{AMF}}(\mathcal{A})\\ &\coloneqq \left|\Pr\left[\mathsf{JdgRecCompDen}^{\mathcal{A},0}_{\mathsf{AMF}} \Rightarrow 1\right] - \Pr\left[\mathsf{JdgRecCompDen}^{\mathcal{A},1}_{\mathsf{AMF}} \Rightarrow 1\right]\right|\\ &\leq \mathsf{negl}(\lambda),\end{aligned}$$

where the game $\mathsf{JdgRecCompDen}^{\mathcal{A},b}_{\mathsf{AMF}}$ *is defined in the bottom right of Figure 12.*

The security requirements for AMF schemes with strong authentication are summarized as follows.

Definition 30 (AMF Schemes with Strong Authentication). *We say that an AMF scheme* AMF *is a* secure AMF scheme with strong authentication *if* AMF *satisfies correctness, strong unforgeability, sender binding, receiver binding, universal deniability, receiver compromise deniability, weak judge compromise deniability and receiver-judge compromise deniability.*

Remark 7. Lai et al. formalized the security of asymmetric group message franking (AGMF) with strong authentication in mind (cf. [LZH+23, Remark 5]). However, judge compromise deniability in

$\mathsf{Unframe}^{\mathcal{A}}_{\mathsf{AMF}}$:

1 : $Q \coloneqq \emptyset$
2 : $(\mathsf{pk}_s, \mathsf{sk}_s) \leftarrow \mathsf{KeyGen}(1^\lambda)$
3 : $(\mathsf{pk}_r, \mathsf{sk}_r, \mathsf{pk}_j, \mathsf{sk}_j, \mathsf{m}, \Sigma) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Frank}}}(\mathsf{pk}_s)$
4 : $b^r_{\mathrm{wf}} \leftarrow \mathsf{WellFormed}_{\mathsf{AMF}}(\mathsf{pk}_r, \mathsf{sk}_r)$
5 : $b^j_{\mathrm{wf}} \leftarrow \mathsf{WellFormed}_{\mathsf{AMF}}(\mathsf{pk}_j, \mathsf{sk}_j)$
6 : **if** $b^r_{\mathrm{wf}} \wedge b^j_{\mathrm{wf}} = 0$ **then**
7 : **return** 0
8 : **if** $(\mathsf{pk}_r, \mathsf{pk}_j, \mathsf{m}) \in Q$ **then**
9 : **return** 0
10 : $b_r \leftarrow \mathsf{Verify}(\mathsf{pk}_s, \mathsf{sk}_r, \mathsf{pk}_j, \mathsf{m}, \Sigma)$
11 : $b_j \leftarrow \mathsf{Judge}(\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{sk}_j, \mathsf{m}, \Sigma)$
12 : **return** $b_r \wedge b_j$

$\mathcal{O}^{\mathsf{Frank}}(\mathsf{pk}'_r, \mathsf{pk}'_j, \mathsf{m})$

1 : $Q \coloneqq Q \cup \{(\mathsf{pk}'_r, \mathsf{pk}'_j, \mathsf{m})\}$
2 : $\Sigma \leftarrow \mathsf{Frank}(\mathsf{sk}_s, \mathsf{pk}'_r, \mathsf{pk}'_j, \mathsf{m})$
3 : **return** Σ

Fig. 13: Security games for AMF schemes: unframeability.

their paper [LZH+23, Definition 9] corresponds to weak judge compromise deniability, and they did not define judge-receiver compromise deniability for AGMF. Therefore, while their AGMF scheme can be used as an AMF scheme, it is not a secure AMF scheme with strong authentication in the sense of Definition 30.

B.2 AMF with Unframeability

Unframeability [HLZW24] guarantees that only the sender is able to produce a signature acceptable to both the receiver and the judge. In other words, an adversary who corrupts both the receiver's secret key and the judge's secret key cannot forge such a valid signature. Therefore, this property enhances accountability.

Definition 31 (Unframeability). *We say that an AMF scheme* AMF *is* unframeable *if, for any efficient adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{Unframe}}_{\mathsf{AMF}}(\mathcal{A}) \coloneqq \Pr\left[\mathsf{Unframe}^{\mathcal{A}}_{\mathsf{AMF}} \Rightarrow 1\right] \leq \mathsf{negl}(\lambda),$$

where the game $\mathsf{Unframe}^{\mathcal{A},b}_{\mathsf{AMF}}$ *is defined in Figure 13.*

Remark 8. Previous unframeability definition [HLZW24] generates these keys honestly and the corresponding secret keys are given to the adversary. We consider unframeability where both the receiver key pair and the judge key pair is generated by the adversary for strong security guarantee. However, we restrict the adversary to output well-formed keys. This well-formedness check is necessary to prove our generic construction based on PKE. This is because an adversary could produce a receiver (resp. judge) key pair with unrelated encryption and decryption keys, and then produce a forgery in which c_r (resp. c_j) is an encryption of 0 for the encryption key but decrypts to 1 using the decryption key. We thus have to assume that this check is possible in the underlying PKE scheme. We leave it as an open problem whether it is possible to construct an AMF scheme that satisfies unframeability without well-formedness check.

It is easy to see that unframeability contradicts judge compromise deniability and judge-receiver compromise deniability. The former prevents the judge and receiver from forging signatures acceptable for Verify and Judge while the latter two allow such forgeries as a desired functionality. Thus, weak judge compromise deniability is compatible with unframeability.

In summary, the security requirements for AMF schemes with unframeability are summarized as follows.

Definition 32 (AMF Schemes with Unframeability). *We say that an AMF scheme* AMF *is a* secure AMF scheme with unframeability *if* AMF *satisfies correctness, strong unforgeability, sender binding, receiver binding, universal deniability, receiver compromise deniability, weak judge compromise deniability and unframeability.*

C Generic Construction of AMF with Unframeability

We can obtain an AMF scheme with unframeability based on our generic framework in Section 4 with a few modifications. First, remark that in AMF with unframeability, the receiver must reject a signature produced by JForge because of strong unforgeability, thus c_r should encrypt 0. But then, to prove weak judge compromise deniability, the bit in c_r has to be flipped while the adversary still generates sk_j, so the ZAP trapdoor has to be set up differently. Unlike AMF with strong deniability, all deniability games now generate the key pair of the sender and of either the receiver or the judge. At the same time, no accountability adversary generates the judge key pair, so we now update the trapdoor accordingly (see below). This change makes the duplicated ciphertext c'_σ useless since the valid ZAP proofs can now be generated with the new trapdoor in every deniability game. Also, because of sender binding, RForge encrypts σ using the judge orphan key (see Section 1.2) which makes strong unforgeability hard to prove since a forgery accepted by Verify can be produced with a valid signature for the sender verification key (encrypted with $\widetilde{\mathsf{ek}_j}$) or with a valid signature for the receiver verification key (encrypted with *missing* $\widetilde{\mathsf{ek}_s}$). Therefore, all algorithms shall generate c_σ (encryption of σ using the sender orphan encryption key) and c'_σ (encryption 0 using the judge orphan encryption key except RForge which encrypts σ). This way, knowing the sender orphan *or* judge orphan decryption key is enough to recover σ in RForge output (note that in $\mathsf{RecCompDen}$ the adversary only generates the receiver key pair). Finally, in the unframeability game, the judge public key is adversarially chosen, which implies that soundness of a ZAP proof using judge randomness does not hold anymore. A simple fix for this is to include two ZAP proofs in every signature: one generated with the sender ZAP randomness and the other with the judge ZAP randomness. Therefore, we can construct the AMF scheme $\mathsf{AMF}_{\mathsf{Unframe}}$ like $\mathsf{AMF}_{\mathsf{GC}}$ in Figure 6 except that:

- JForge now encrypts 0 in c_r (and Disjunction (4) is updated accordingly in relation $\mathcal{R}$),
- in every signing algorithm, the message m is concatenated with both the receiver and judge public keys (instead of only the former one) because of the definition of forbidden forgeries in both strong unforgeability and unframeability,
- the trapdoor for switching witnesses is changed: Equation (0) evaluates to true if $\mathsf{td}_s = \mathsf{Enc}(\mathsf{ek}_s, 1; r^s_{\mathsf{td}}) \wedge \Big(\mathsf{td}_r = \mathsf{Enc}(\mathsf{ek}_r, 1; r^r_{\mathsf{td}}) \vee \mathsf{td}_j = \mathsf{Enc}(\mathsf{ek}_j, 1; r^j_{\mathsf{td}})\Big)$, thus the syntax of a witness for $\mathcal{R}$ has to be updated to include the randomness associated with the trapdoor td of all three parties,
- c_σ always encrypts σ using sender orphan encryption key and c'_σ encrypts 0 or σ using judge orphan encryption key depending on the signing algorithm (and $\mathcal{R}$ is updated accordingly), and
- a ZAP proof generated with the sender randomness is added to the signature.

Figure 14 shows the modifed AMF scheme $\mathsf{AMF}_{\mathsf{Unframe}}$. We now use a ZAP proof system for the NP language $\mathcal{L}_{\mathsf{Unframe}}$ defined by the relation $\mathcal{R}_{\mathsf{Unframe}}$ defined as follows:

$$
\begin{aligned}
\mathcal{R}_{\mathsf{Unframe}} \coloneqq \{\ & (\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{pk}_j, c_\sigma, c'_\sigma, c_r, c_j, m_{r,j}), (\sigma, r_\sigma, r'_\sigma, r_r, r_j, r^s_{\mathsf{td}}, r^r_{\mathsf{td}}, r^j_{\mathsf{td}}) \quad \text{such that} \\
& \Big[\mathsf{td}_s = \mathsf{Enc}(\mathsf{ek}_s, 1; r^s_{\mathsf{td}}) \wedge \Big(\mathsf{td}_r = \mathsf{Enc}(\mathsf{ek}_r, 1; r^r_{\mathsf{td}}) \vee \mathsf{td}_j = \mathsf{Enc}(\mathsf{ek}_j, 1; r^j_{\mathsf{td}})\Big)\Big] \\
\vee\ & \Big[\mathsf{SIG.Verify}(\mathsf{vk}_s, m_{r,j}, \sigma) \wedge c_\sigma = \mathsf{Enc}(\widetilde{\mathsf{ek}_s}, \sigma; r_\sigma) \wedge c_r = \mathsf{Enc}(\mathsf{ek}_r, 1; r_r) \wedge c_j = \mathsf{Enc}(\mathsf{ek}_j, 1; r_j)\Big] \\
\vee\ & [c_r = \mathsf{Enc}(\mathsf{ek}_r, 0; r_r) \wedge c_j = \mathsf{Enc}(\mathsf{ek}_j, 0; r_j)] \\
\vee\ & \Big[\mathsf{SIG.Verify}(\mathsf{vk}_r, m_{r,j}, \sigma) \wedge c_\sigma = \mathsf{Enc}(\widetilde{\mathsf{ek}_s}, \sigma; r_\sigma) \wedge c'_\sigma = \mathsf{Enc}(\widetilde{\mathsf{ek}_j}, \sigma; r'_\sigma) \wedge c_r = \mathsf{Enc}(\mathsf{ek}_r, 1; r_r) \wedge c_j = \mathsf{Enc}(\mathsf{ek}_j, 0; r_j)\Big] \\
\vee\ & \Big[\mathsf{SIG.Verify}(\mathsf{vk}_j, m_{r,j}, \sigma) \wedge c_\sigma = \mathsf{Enc}(\widetilde{\mathsf{ek}_s}, \sigma'; r_\sigma) \wedge c_r = \mathsf{Enc}(\mathsf{ek}_r, 0; r_r) \wedge c_j = \mathsf{Enc}(\mathsf{ek}_j, 1; r_j)\Big]\}
\end{aligned}
$$

with $\mathsf{pk}_s = (\mathsf{vk}_s, \mathsf{ek}_s, \widetilde{\mathsf{ek}_s}, \mathsf{zr}_s, \mathsf{td}_s)$, $\mathsf{pk}_r = (\mathsf{vk}_r, \mathsf{ek}_r, \widetilde{\mathsf{ek}_r}, \mathsf{zr}_r, \mathsf{td}_r)$, $\mathsf{pk}_j = (\mathsf{vk}_j, \mathsf{ek}_j, \widetilde{\mathsf{ek}_j}, \mathsf{zr}_j, \mathsf{td}_j)$ and $m_{r,j} = m||\mathsf{pk}_r||\mathsf{pk}_j$.

We can prove that $\mathsf{AMF}_{\mathsf{Unframe}}$ is secure AMF scheme with unframeability. Correctness proof is trivial, and sender binding and receiver binding, universal deniability, and receiver compromise deniability can be proved similarly to $\mathsf{AMF}_{\mathsf{GC}}$. We note that, in the proof of sender binding, a forged

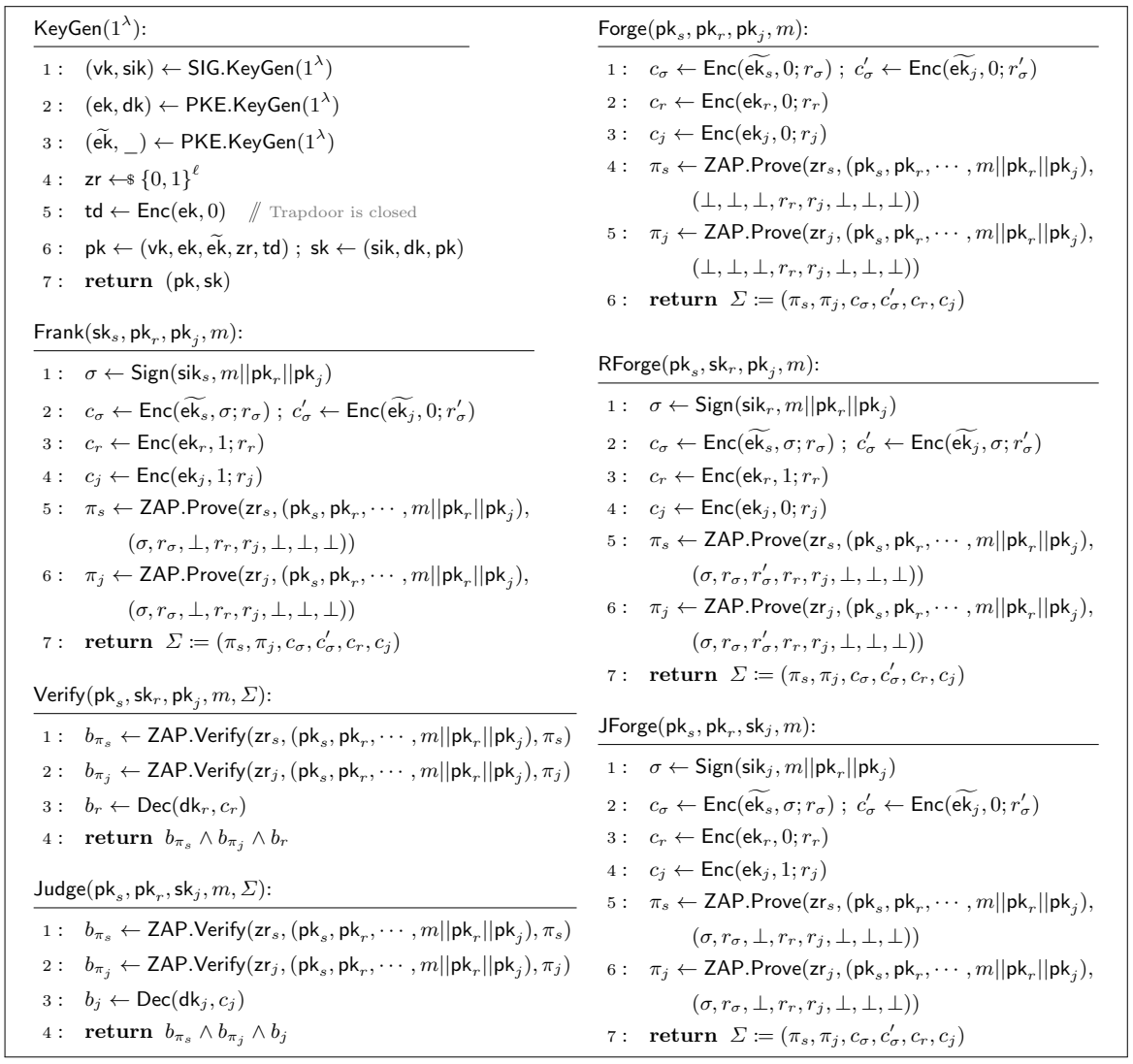


Fig. 14: Algorithms of our generic construction $\mathsf{AMF}_{\mathsf{Unframe}}$: The encryption randomness r_σ, r'_σ, r_r and r_j are assumed to be generated honestly when needed. The symbol $\bot$ represents the universal void witness.

signature is extracted from the ciphertext c'_σ; in the proof of universal deniability and receiver compromise deniability, the witnesses of ZAP are swapped using the trapdoors r^s_{td} and r^j_{td} since the sender and judge key pairs are honestly generated.

Thus, we only sketch the proofs for strong unforgeability, unframeability and weak judge compromise deniability.

Theorem 13 (Strong Unforgeability for $\mathsf{AMF}_{\mathsf{Unframe}}$). *If* SIG *is* EUF-CMA *secure, if* PKE *is perfectly correct and if* ZAP *is adaptively sound then the AMF scheme* $\mathsf{AMF}_{\mathsf{Unframe}}$ *is strong unforgeable.*

Proof (Sketch). The proof is almost identical to the proof of Theorem 4 except that the challenge signature key in EUF-CMA game of SIG is embedded into the sender's or receiver's public key. The soundness of the ZAP generated using the sender randomness zr_s guarantees that a forged AMF signature accepted by Verify contains a valid signature under the sender's or receiver's verification key, encrypted in c_σ with the sender orphan encryption key. Since the sender key pair is generated by the reduction, the latter knows the corresponding decryption key, and can extract a forged signature from the ciphertext. □

Theorem 14 (Unframeability for $\mathsf{AMF}_{\mathsf{Unframe}}$). *If* SIG *is* EUF-CMA *secure, if* PKE *is perfectly correct and if* ZAP *is adaptively sound then the AMF scheme* $\mathsf{AMF}_{\mathsf{Unframe}}$ *is unframeable.*

Proof (Sketch). The proof is almost identical to the proof of Theorem 4 except that the challenge signature key in EUF-CMA game is embedded into the sender's public key. The soundness of the ZAP generated using the sender randomness zr_s guarantees that a forged signature accepted by both Verify and Judge must contain a valid signature under the sender's verification key, encrypted in c_σ with the sender's orphan key. Since the sender key pair is generated by the reduction, the latter knows the corresponding decryption key, and can extract a forged signature from the ciphertext. □

Theorem 15 (Weak Judge Compromise Deniability for $\mathsf{AMF}_{\mathsf{Unframe}}$). *If* PKE *is* MC-IND-CPA *secure, and if* ZAP *is statistically witness indistinguishable then the AMF scheme* $\mathsf{AMF}_{\mathsf{Unframe}}$ *achieves weak judge compromise deniability.*

Proof (Sketch). The proof is identical to the proof of Theorem 7 (with receiver and judge swapped) since receiver compromise deniability and weak judge compromise deniability are symmetrical. In particular, contrary to the proof of Theorem 8, the witnesses are swapped using the trapdoors r^s_{td} and r^r_{td} since the challenger generates the sender and receiver key pairs. □

From Theorems 3 to 7 and 13 to 15, we conclude the following corollary.

Corollary 4. *If* PKE *is perfectly correct,* MC-IND-CPA *secure and* MC-IK-CPA *secure, if* SIG *is correct and* EUF-CMA *secure, and if* ZAP *is correct, witness indistinguishable and adaptively sound then the AMF scheme* $\mathsf{AMF}_{\mathsf{Unframe}}$ *is a secure AMF scheme with unframeability.*

D AMF from Multi-Designated Verifier Signature

In [TGL+19b, Section G], Tyagi et al. discussed an AMF scheme based on designated verifier signatures (DVS) and designated verifier proof systems. They pointed out that achieving judge compromise deniability seems difficult for such an AMF scheme and achieving accountability is challenging because the accountability game provides verification oracles, whereas DVS's unforgeability games do not incorporate such oracles. After [TGL+19b] was published, Damgård et al. [DHM+20] and its follow-up works [MPR22b, CHMR23, HYH25] provided formalization for multi-designated verifier signature (MDVS), a generalization of DVS allowing the signer to specify multiple designated verifiers. Since the receiver and the judge in the AMF setting can be viewed as designated verifiers, MDVS looks more suitable to construct an AMF scheme than DVS. In addition, the MDVS's definitions in [DHM+20, CHMR23, HYH25] provided verification oracles, which seems to be sufficient for the purpose of constructing AMF schemes.

In this section, we discuss the relationship between AMF and MDVS, and show that a simple generic construction of AMF from MDVS is a secure AMF scheme with strong authentication. This is the first AMF scheme with strong authentication satisfying all the security notions defined by Tyagi et al. Besides, we show that it does not satisfy judge compromise deniability and unframeability. Our analysis reveals that these two properties make a significant difference between MDVS and AMF.

D.1 Recap Multi-Designated Verifier Signature

We first recall the syntax and security notions of multi-designated verifier signature (MDVS). We follow the existing works [DHM+20,CHMR23,HYH25], but consider a syntax without Setup algorithm for the compatibility with AMF schemes.

Definition 33 (Multi-Designated Verifier Signature). *A multi-designated verifier signature scheme* MDVS *consists of five algorithms.*

- $\mathsf{SKGen}(1^\lambda) \to (\mathsf{spk}, \mathsf{ssk})$: *The probabilistic signer key generation algorithm takes a security parameter* 1^λ *as input and outputs a pair of public and secret keys for a* signer $(\mathsf{spk}, \mathsf{ssk})$[17].
- $\mathsf{VKGen}(1^\lambda) \to (\mathsf{vpk}, \mathsf{vsk})$: *The probabilistic verifier key generation algorithm takes a security parameter* 1^λ *as input and outputs a pair of public and secret keys for a* verifier $(\mathsf{vpk}, \mathsf{vsk})$[17].
- $\mathsf{Sign}(\mathsf{ssk}, \mathcal{D}, \mathsf{m}) \to \sigma$: *The probabilistic signing algorithm takes a signer's secret key* ssk*, a designated set (a set of designated verifiers' public keys)* $\mathcal{D} = \{\mathsf{vpk}_j\}_{j\in[|\mathcal{D}|]}$*, and a message* m *as input and outputs a signature* σ*.*
- $\mathsf{Vrfy}(\mathsf{spk}, \mathsf{vsk}, \mathcal{D}, \mathsf{m}, \sigma) \to \{0,1\}$: *The deterministic private verification algorithm takes a signer's public key* spk*, a verifier's secret key* vsk*, a designated set* $\mathcal{D}$*, a message* m*, and a signature* σ *as input and outputs* 1 *or* 0*.*
- $\mathsf{Sim}(\mathsf{spk}, \mathcal{D}, \mathcal{C}, \mathsf{m}) \to \sigma$: *The probabilistic signature simulation algorithm takes* spk*, a designated set* $\mathcal{D}$*, a set of verifiers' secret keys* $\mathcal{C} = \{\mathsf{vsk}_j\}_{j\in[|\mathcal{C}|]}$ *(*$0 \le |\mathcal{C}| \le |\mathcal{D}|$*), and a message* m *as input and outputs a signature* σ*.*

We assume anyone can check the validity of a given verifier's key pair $(\mathsf{vpk}, \mathsf{vsk})$ with an efficient algorithm $\mathsf{WellFormed}_{\mathsf{vpk}}$.

Correctness guarantees the designated verifiers accept signatures generated by a signer.

Definition 34 (Correctness). *We say that an MDVS scheme* MDVS *is* correct *if for any* $\mu = \mathsf{poly}(\lambda)$*, any* $1^\lambda \leftarrow \mathsf{Setup}(1^\lambda)$*, any signer key* $(\mathsf{spk}, \mathsf{spk}) \leftarrow \mathsf{SKGen}(1^\lambda)$*, all verifier keys* $(\mathsf{vpk}_i, \mathsf{vpk}_i) \leftarrow \mathsf{VKGen}(1^\lambda)$ $(i \in [\mu])$*, and any message* m*, it holds that* $\mathsf{Vrfy}(\mathsf{spk}, \mathsf{vsk}_i, \mathcal{D}, \mathsf{m}, \sigma) = 1$ *with overwhelming probability, where* $\mathcal{D} := \{\mathsf{vpk}_j\}_{j\in[\mu]}$ *and* $\sigma \leftarrow \mathsf{Sign}(\mathsf{ssk}, \mathcal{D}, \mathsf{m})$*.*

The main security properties of MDVS scheme are consistency, unforgeability, and off-the-record: Consistency ensures that a malicious signer cannot create ill-formed signatures such that an uncorrupted verifier accepts but another an uncorrupted verifier rejects (here, "uncorrupted" means the secret key of the verifier is not known to the adversary). Unforgeability ensures designated verifiers cannot forge signatures from a signer to another designated verifier without knowledge of either secret keys. Off-the-record ensures that a signature created from $\mathsf{MDVS.Sign}$ is indistinguishable from a simulated signature from $\mathsf{MDVS.Sim}$ even with access to the signer's and the designated verifiers' secret keys that is used as the input for $\mathsf{MDVS.Sim}$.

These properties are formally defined as follows. We here define a slightly stronger version of them than the existing works [DHM+20,CHMR23,HYH25], which seem to be sufficient for AFM. For example, the adversary against the unforgeability game is allowed to include malicious designated verifier's keys to the forged signature and the signing and verification oracles accept maliciously-generated verifier's keys.

Definition 35 (Consistency). *We say that an MDVS scheme* MDVS *has* consistency *if, for any efficient adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{Con}}_{\mathsf{MDVS}}(\mathcal{A}) := \Pr\left[\mathsf{Con}^{\mathcal{A}}_{\mathsf{MDVS}} \Rightarrow 1\right] \le \mathsf{negl}(\lambda),$$

where the game $\mathsf{Con}^{\mathcal{A}}_{\mathsf{MDVS}}$ *is defined in Figure 15.*

Definition 36 (Unforgeability). *We say that an MDVS scheme* MDVS *is* unforgeable *if, for any efficient adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{Unf}}_{\mathsf{MDVS}}(\mathcal{A}) := \Pr\left[\mathsf{Unf}^{\mathcal{A}}_{\mathsf{MDVS}} \Rightarrow 1\right] \le \mathsf{negl}(\lambda),$$

where the game $\mathsf{Unf}^{\mathcal{A}}_{\mathsf{MDVS}}$ *is defined in Figure 15.*

[17] Assume the secret key contains the corresponding public key.

Definition 37 (Off-The-Record). *We say that an MDVS scheme* MDVS *is* off-the-record *if, for any efficient adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{OTR}}_{\mathsf{MDVS}}(\mathcal{A}) \coloneqq \left|\Pr\left[\mathsf{OTR}^{\mathcal{A},0}_{\mathsf{MDVS}} \Rightarrow 1\right] - \Pr\left[\mathsf{OTR}^{\mathcal{A},1}_{\mathsf{MDVS}} \Rightarrow 1\right]\right| \leq \mathsf{negl}(\lambda),$$

where the game $\mathsf{Unf}^{\mathcal{A}}_{\mathsf{MDVS}}$ *is defined in Figure 15.*

D.2 Generic Construction of AMF from MDVS

We can construct an AMF scheme from an MDVS scheme directly and naturally. The pseudocode for the AMF scheme $\mathsf{AMF}_{\mathsf{MDVS}}$ is provided in Figure 16. We can show that $\mathsf{AMF}_{\mathsf{MDVS}}$ is a secure AMF scheme with strong authentication (cf. Definition 30)[18].

- Strong unforgeability and receiver binding follow from the unforgeability of MDVS. Recall that the unforgeability of MDVS guarantees that the adversary, who corrupt some of verifiers, cannot create a signature accepted by uncorrupted verifiers. Thus, the sUnforge adversary who does not know the sender and the receiver's secret keys cannot forge a signature accepted by the receiver even if it knows the judge secret key. Similarly, the r-Bind adversary who does not know the sender and the judge's secret keys cannot forge a signature accepted by the judge.
- Sender binding follows from the consistency of MDVS. Since the s-Bind adversary does not know both the receiver's and judge's secret keys, consistency guarantees that the adversary cannot create a signature that is accepted by the receiver but is not by the judge, even if the adversary chooses the sender's key by itself.
- Universal deniability, receiver compromise deniability, weak judge compromise deniability and judge-receiver compromise deniability of $\mathsf{AMF}_{\mathsf{MDVS}}$ are direct consequences of the OTR property of MDVS. The OTR property ensures that when the verifiers' secret keys used by MDVS.Sim and the verifiers' secret keys known by the distinguisher are identical, simulated signatures and real signatures are indistinguishable. We can observe that these four deniability games have this situation.

It is worth noting that $\mathsf{AMF}_{\mathsf{MDVS}}$ does not satisfy judge compromise deniability nor unframeability. According to the unforgeability of MDVS, designated verifiers can distinguish real signatures and simulated signatures if the simulated signature are generated without their secret keys. In $\mathsf{AMF}_{\mathsf{MDVS}}$, only the judge's secret key vsk_j is used for signature generation, but the JdgCompDen adversary knows both vsk_r and vsk_j. Thus, the adversary can distinguish whether the signature is generated by MDVS.Sign with ssk_s or MDVS.Sim with vsk_j by using vsk_r. Furthermore, the JRForge algorithm described in Figure 16 can produce a signature (for any message) accepted by honest Verify and Judge algorithms, thus $\mathsf{AMF}_{\mathsf{MDVS}}$ does not satisfy unframeability.

E On Multi-Challenge IK-CPA Security

For completeness, we show that the standard single-challenge IK-CPA security implies MC-IK-CPA security.

Theorem 16. *Let* PKE *be a PKE scheme. Then, for any adversary* $\mathcal{A}$ *against* MC-IND-CPA *security who makes* q_E *challenge queries, there exists an adversary* $\mathcal{B}$ *against* IK-CPA *security such that*

$$\mathsf{Adv}^{\mathsf{MC\text{-}IK\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) = q_E \cdot \mathsf{Adv}^{\mathsf{IK\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B}).$$

Note that the running time of $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

[18] Strictly speaking, $\mathsf{AMF}_{\mathsf{MDVS}}$ only achieves a weak form of strong unforgeability where the judge key pair is honestly-generated and the corresponding secret key is leaked to the adversary because the unforgeability of MDVS holds when user key pairs are honestly-generated.

$\mathsf{Con}^{\mathcal{A}}_{\mathsf{MDVS}}$

$L_{\mathsf{SPK}}, L_{\mathsf{SSK}}, L_{\mathsf{VPK}}, L_{\mathsf{VSK}}, L_{\mathsf{Sig}}, L_{\mathsf{Ver}} \coloneqq \emptyset$
$(\mathsf{spk}^*, \mathcal{D}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{SPK}}, \mathcal{O}^{\mathsf{SSK}}, \mathcal{O}^{\mathsf{VPK}}, \mathcal{O}^{\mathsf{VSK}}, \mathcal{O}^{\mathsf{Sig}}, \mathcal{O}^{\mathsf{Ver}}}(1^\lambda)$
if $\forall \mathsf{vpk} \in \mathcal{D}^* : (\mathsf{vpk}, \mathsf{vsk}) \in L_{\mathsf{VPK}}$
  $\wedge\, \exists \mathsf{vpk}_0^*, \mathsf{vpk}_1^* \in \mathcal{D}^* :$
    $\mathsf{vpk}_0^* \neq \mathsf{vpk}_1^* \wedge \mathsf{vpk}_0^* \notin L_{\mathsf{VSK}} \wedge \mathsf{vpk}_1^* \notin L_{\mathsf{VSK}}$
    $\wedge\, \mathsf{Vrfy}(\mathsf{spk}^*, \mathsf{vsk}_0^*, \mathcal{D}^*, \mathsf{m}^*, \sigma^*) = 0$
    $\wedge\, \mathsf{Vrfy}(\mathsf{spk}^*, \mathsf{vsk}_1^*, \mathcal{D}^*, \mathsf{m}^*, \sigma^*) = 1$ **then**
  return 1
else return 0

$\mathsf{Unf}^{\mathcal{A}}_{\mathsf{MDVS}}$:

$L_{\mathsf{SPK}}, L_{\mathsf{SSK}}, L_{\mathsf{VPK}}, L_{\mathsf{VSK}}, L_{\mathsf{Sig}}, L_{\mathsf{Ver}} \coloneqq \emptyset$
$(\mathsf{spk}^*, \mathcal{D}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{SPK}}, \mathcal{O}^{\mathsf{SSK}}, \mathcal{O}^{\mathsf{VPK}}, \mathcal{O}^{\mathsf{VSK}}, \mathcal{O}^{\mathsf{Sig}}, \mathcal{O}^{\mathsf{Ver}}}(1^\lambda)$
if $(\mathsf{spk}^*, \mathsf{ssk}^*) \in L_{\mathsf{SPK}} \wedge \mathsf{spk}^* \notin L_{\mathsf{SSK}}$
  $\wedge\, (\mathsf{spk}^*, \mathcal{D}^*, \mathsf{m}^*) \notin L_{\mathsf{Sig}}$
  $\wedge\, \exists \mathsf{vpk}^* \in \mathcal{D}^* :$
    $(\mathsf{vpk}^*, \mathsf{vsk}^*) \in L_{\mathsf{VPK}} \wedge \mathsf{vpk}^* \notin L_{\mathsf{VSK}}$
    $\wedge\, \mathsf{Vrfy}(\mathsf{spk}^*, \mathsf{vsk}^*, \mathcal{D}^*, \mathsf{m}^*, \sigma^*) = 1$
  return 1
else return 0

$\mathsf{OTR}^{\mathcal{A},b}_{\mathsf{MDVS}}$:

$L_{\mathsf{SPK}}, L_{\mathsf{SSK}}, L_{\mathsf{VPK}}, L_{\mathsf{VSK}}, L_{\mathsf{Sig}}, L_{\mathsf{Ver}} \coloneqq \emptyset$
$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathsf{spk}^*, \mathcal{D}^*, \mathcal{C}^*, \mathsf{m}^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{SPK}}, \mathcal{O}^{\mathsf{SSK}}, \mathcal{O}^{\mathsf{VPK}}, \mathcal{O}^{\mathsf{VSK}}, \mathcal{O}^{\mathsf{Sig}}, \mathcal{O}^{\mathsf{Ver}}}(1^\lambda)$
$\mathcal{C}^*_{\mathsf{vpk}} \coloneqq \{\mathsf{vpk} : (\mathsf{vpk}, \mathsf{vsk}) \in \mathcal{C}^*\}$
$\mathcal{C}^*_{\mathsf{vsk}} \coloneqq \{\mathsf{vsk} : (\mathsf{vpk}, \mathsf{vsk}) \in \mathcal{C}^*\}$
if $\exists (\mathsf{vpk}, \mathsf{vsk}) \in \mathcal{C}^* : \mathsf{WellFormed}_{\mathsf{vpk}}(\mathsf{vpk}, \mathsf{vsk}) = 0$ **then**
  return 1
if $(\mathsf{spk}^*, *) \notin L_{\mathsf{SPK}} \vee \mathcal{C}^*_{\mathsf{vpk}} \not\subseteq \mathcal{D}^*$ **then**
  return 1
$\mathsf{ssk}^* \leftarrow L_{\mathsf{SPK}}[\mathsf{spk}^*]$
$\sigma_0^* \leftarrow \mathsf{Sign}(\mathsf{ssk}^*, \mathcal{D}^*, \mathsf{m}^*)$
$\sigma_1^* \leftarrow \mathsf{Sim}(\mathsf{spk}^*, \mathcal{D}^*, \mathcal{C}^*_{\mathsf{vsk}}, \mathsf{m}^*)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{SPK}}, \mathcal{O}^{\mathsf{SSK}}, \mathcal{O}^{\mathsf{VPK}}, \mathcal{O}^{\mathsf{VSK}}, \mathcal{O}^{\mathsf{Sig}}, \mathcal{O}^{\mathsf{Ver}}}(\sigma_b^*)$
if $\exists \mathsf{vpk}^* \in \mathcal{D}^* \setminus \mathcal{C}^*_{\mathsf{vpk}} :$
  $\mathsf{vpk}^* \in L_{\mathsf{VSK}} \vee (\mathsf{spk}^*, \mathsf{vpk}^*, \mathcal{D}^*, \mathsf{m}^*, \sigma_b) \notin L_{\mathsf{Ver}}$ **then**
  return 1
return b'

$\mathcal{O}^{\mathsf{SPK}}()$

$(\mathsf{spk}, \mathsf{ssk}) \leftarrow \mathsf{SKGen}(1^\lambda)$
$L_{\mathsf{SPK}} \coloneqq L_{\mathsf{SPK}} \cup \{(\mathsf{spk}, \mathsf{ssk})\}$
return spk

$\mathcal{O}^{\mathsf{SSK}}(\mathsf{spk})$

if $(\mathsf{spk}, *) \notin L_{\mathsf{SPK}}$ **then**
  return $\perp$
$L_{\mathsf{SSK}} \coloneqq \mathsf{spk}$
$\mathsf{ssk} \leftarrow L_{\mathsf{SPK}}[\mathsf{spk}]$
return ssk

$\mathcal{O}^{\mathsf{VPK}}()$

$(\mathsf{vpk}, \mathsf{vsk}) \leftarrow \mathsf{VKGen}(1^\lambda)$
$L_{\mathsf{VPK}} \coloneqq L_{\mathsf{VPK}} \cup \{(\mathsf{vpk}, \mathsf{vsk})\}$
return vpk

$\mathcal{O}^{\mathsf{VSK}}(\mathsf{vpk})$

if $(\mathsf{vpk}, *) \notin L_{\mathsf{VPK}}$ **then**
  return $\perp$
$L_{\mathsf{VSK}} \coloneqq L_{\mathsf{VSK}} \cup \{\mathsf{vpk}\}$
$\mathsf{vsk} \leftarrow L_{\mathsf{VPK}}[\mathsf{vpk}]$
return vsk

$\mathcal{O}^{\mathsf{Sig}}(\mathsf{spk}, \mathcal{D}, \mathsf{m})$

if $(\mathsf{spk}, *) \notin L_{\mathsf{SPK}}$ **then**
  return $\perp$
$\mathsf{ssk} \leftarrow L_{\mathsf{SPK}}[\mathsf{spk}]$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{ssk}, \mathcal{D}, \mathsf{m})$
$L_{\mathsf{Sig}} \coloneqq L_{\mathsf{Sig}} \cup \{(\mathsf{spk}, \mathcal{D}, \mathsf{m})\}$
return σ

$\mathcal{O}^{\mathsf{Ver}}(\mathsf{spk}, \mathsf{vpk}, \mathcal{D}, \mathsf{m}, \sigma)$

if $(\mathsf{vpk}, *) \notin L_{\mathsf{VPK}}$ **then**
  return $\perp$
$\mathsf{vsk} \leftarrow L_{\mathsf{VPK}}[\mathsf{vpk}]$
$b \leftarrow \mathsf{Vrfy}(\mathsf{spk}, \mathsf{vsk}, \mathcal{D}, \mathsf{m}, \sigma)$
$L_{\mathsf{Ver}} \coloneqq L_{\mathsf{Ver}} \cup \{(\mathsf{spk}, \mathsf{vpk}, \mathcal{D}, \mathsf{m}, \sigma)\}$
return b

Fig. 15: Security games for multi-designated verifier signatures.

$\mathsf{Frank}(\mathsf{sk}_s, \mathsf{pk}_r, \mathsf{pk}_j, \mathsf{m})$

1 : $\mathcal{D} \coloneqq \{\mathsf{vpk}_r, \mathsf{vpk}_j\}$
2 : **return** $\mathsf{MDVS.Sign}(\mathsf{ssk}_s, \mathcal{D}, \mathsf{m})$

$\mathsf{Forge}(\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{pk}_j, \mathsf{m})$

1 : $\mathcal{D} \coloneqq \{\mathsf{vpk}_r, \mathsf{vpk}_j\}$
2 : **return** $\mathsf{MDVS.Sim}(\mathsf{spk}_s, \mathcal{D}, \emptyset, \mathsf{m})$

$\mathsf{RForge}(\mathsf{pk}_s, \mathsf{sk}_r, \mathsf{pk}_j, \mathsf{m})$

1 : $\mathcal{D} \coloneqq \{\mathsf{vpk}_r, \mathsf{vpk}_j\}$
2 : **return** $\mathsf{MDVS.Sim}(\mathsf{spk}_s, \mathcal{D}, \{\mathsf{vsk}_r\}, \mathsf{m})$

$\mathsf{JForge}(\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{sk}_j, \mathsf{m})$

1 : $\mathcal{D} \coloneqq \{\mathsf{vpk}_r, \mathsf{vpk}_j\}$
2 : **return** $\mathsf{MDVS.Sim}(\mathsf{spk}_s, \mathcal{D}, \{\mathsf{vsk}_j\}, \mathsf{m})$

$\mathsf{JRForge}(\mathsf{pk}_s, \mathsf{sk}_r, \mathsf{sk}_j, \mathsf{m})$

1 : $\mathcal{D} \coloneqq \{\mathsf{vpk}_r, \mathsf{vpk}_j\}$
2 : **return** $\mathsf{MDVS.Sim}(\mathsf{spk}_s, \mathcal{D}, \{\mathsf{vsk}_r, \mathsf{vsk}_j\}, \mathsf{m})$

$\mathsf{KeyGen}(1^\lambda)$

1 : $(\mathsf{spk}, \mathsf{ssk}) \leftarrow \mathsf{MDVS.SKGen}(1^\lambda)$
2 : $(\mathsf{vpk}, \mathsf{vsk}) \leftarrow \mathsf{MDVS.VKGen}(1^\lambda)$
3 : $\mathsf{pk} \coloneqq (\mathsf{spk}, \mathsf{vpk})$
4 : $\mathsf{sk} \coloneqq (\mathsf{ssk}, \mathsf{vsk}, \mathsf{pk})$
5 : **return** $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Verify}(\mathsf{pk}_s, \mathsf{sk}_r, \mathsf{pk}_j, \mathsf{m}, \Sigma)$

1 : $\mathcal{D} \coloneqq \{\mathsf{vpk}_r, \mathsf{vpk}_j\}$
2 : **return** $\mathsf{MDVS.Vrfy}(\mathsf{spk}_s, \mathsf{vsk}_r, \mathcal{D}, \mathsf{m}, \Sigma)$

$\mathsf{Judge}(\mathsf{pk}_s, \mathsf{pk}_r, \mathsf{sk}_j, \mathsf{m}, \Sigma)$

1 : $\mathcal{D} \coloneqq \{\mathsf{vpk}_r, \mathsf{vpk}_j\}$
2 : **return** $\mathsf{MDVS.Vrfy}(\mathsf{spk}_s, \mathsf{vsk}_j, \mathcal{D}, \mathsf{m}, \Sigma)$

Fig. 16: The AMF scheme $\mathsf{AMF}_{\mathsf{MDVS}}$ based on MDVS.

Proof. To prove the theorem, we rely on hybrid arguments [MF21, Section 3.2.2]. We consider the following sequence of games from G_0 to $\mathsf{G}_{\mathsf{q_E}}$.

Game G_0. This is identical to $\mathsf{MC\text{-}IK\text{-}CPA}^{\mathcal{A},0}_{\mathsf{PKE}}$ game. The encryption oracle returns $\mathsf{Enc}(\mathsf{ek}^*_0, \mathsf{m})$ for the received m.

Game G_i *for* $i \in \{1, \ldots, q_E - 1\}$. This is identical to $\mathsf{G}_{\mathsf{i}-1}$ except that for queries from 1 to i, the encryption oracle returns $\mathsf{Enc}(\mathsf{ek}^*_1, \mathsf{m})$ for the received m instead of $\mathsf{Enc}(\mathsf{ek}^*_0, \mathsf{m})$.

Game $\mathsf{G}_{\mathsf{q_E}}$. This is identical to $\mathsf{MC\text{-}IK\text{-}CPA}^{\mathcal{A},1}_{\mathsf{PKE}}$ game. The encryption oracle returns $\mathsf{Enc}(\mathsf{ek}^*_1, \mathsf{m})$ for the received m.

By definition, we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{MC\text{-}IK\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) &= \left|\Pr\left[\mathsf{MC\text{-}IK\text{-}CPA}^{\mathcal{A},0}_{\mathsf{PKE}} \Rightarrow 1\right] - \Pr\left[\mathsf{MC\text{-}IK\text{-}CPA}^{\mathcal{A},1}_{\mathsf{PKE}} \Rightarrow 1\right]\right| \\
&= \left|\Pr[\mathsf{G}_0 \Rightarrow 1] - \Pr[\mathsf{G}_{\mathsf{q_E}} \Rightarrow 1]\right| \\
&= \left|\sum_{i=1}^{q_E} (\Pr[\mathsf{G}_{\mathsf{i}-1} \Rightarrow 1] - \Pr[\mathsf{G}_\mathsf{i} \Rightarrow 1])\right|
\end{aligned}$$

Next, we will construct an $\mathsf{IK\text{-}CPA}$ adversary $\mathcal{B}$ as follows. Upon receiving two encryption keys ek^*_0 and ek^*_1, $\mathcal{B}$ first chooses $i \in [q_E]$ at random, Then it simulates $\mathsf{MC\text{-}IK\text{-}CPA}$ game against $\mathcal{A}$ by implementing the encryption oracle as follows. Let c be the counter that counts the number of encryption queries:

- If $c < i$, it returns $\mathsf{Enc}(\mathsf{ek}^*_1, \mathsf{m})$.
- If $c = i$, it sends m to its encryption oracle and receives ct^*. Then, it returns ct^* to $\mathcal{A}$.
- If $c > i$, it returns $\mathsf{Enc}(\mathsf{ek}^*_0, \mathsf{m})$.

Finally, $\mathcal{A}$ outputs a bit b. Then, $\mathcal{B}$ outputs b as its final output.

When $\mathcal{B}$ runs $\mathsf{IK\text{-}CPA}^{\mathcal{A},0}_{\mathsf{PKE}}$, $\mathcal{A}$'s view is identical to game $\mathsf{G}_{\mathsf{i}-1}$ and $\mathcal{B}$'s output has the same distribution as $\mathcal{A}$ in $\mathsf{G}_{\mathsf{i}-1}$; When $\mathcal{B}$ runs $\mathsf{IK\text{-}CPA}^{\mathcal{A},1}_{\mathsf{PKE}}$, $\mathcal{A}$'s view is identical to game G_i and $\mathcal{B}$'s output has

the same distribution as $\mathcal{A}$ in $\mathsf{G_i}$. Therefore, we have

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{IK\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B}) \\
&= \left|\Pr\left[\mathsf{IK\text{-}CPA}^{\mathcal{B},0}_{\mathsf{PKE}} \Rightarrow 1\right] - \Pr\left[\mathsf{IK\text{-}CPA}^{\mathcal{B},1}_{\mathsf{PKE}} \Rightarrow 1\right]\right| \\
&= \left|\sum_{j=1}^{q_E}\left(\Pr\left[\mathsf{IK\text{-}CPA}^{\mathcal{B},0}_{\mathsf{PKE}} \Rightarrow 1 \wedge i = j\right] - \Pr\left[\mathsf{IK\text{-}CPA}^{\mathcal{B},1}_{\mathsf{PKE}} \Rightarrow 1\right] \wedge i = j\right)\right| \\
&= \left|\sum_{j=1}^{q_E}\Pr[i=j]\left(\Pr\left[\mathsf{IK\text{-}CPA}^{\mathcal{B},0}_{\mathsf{PKE}} \Rightarrow 1 | i = j\right] - \Pr\left[\mathsf{IK\text{-}CPA}^{\mathcal{B},1}_{\mathsf{PKE}} \Rightarrow 1 | i = j\right]\right)\right| \\
&= \frac{1}{q_E}\left|\sum_{j=1}^{q_E}\left(\Pr\left[\mathsf{IK\text{-}CPA}^{\mathcal{B},0}_{\mathsf{PKE}} \Rightarrow 1 | i = j\right] - \Pr\left[\mathsf{IK\text{-}CPA}^{\mathcal{B},1}_{\mathsf{PKE}} \Rightarrow 1 | i = j\right]\right)\right| \\
&= \frac{1}{q_E}\left|\sum_{j=1}^{q_E}\left(\Pr[\mathsf{G_{j-1}} \Rightarrow 1] - \Pr[\mathsf{G_j} \Rightarrow 1]\right)\right| \\
&= \frac{1}{q_E}\mathsf{Adv}^{\mathsf{MC\text{-}IK\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A})
\end{aligned}$$

We conclude that

$$\mathsf{Adv}^{\mathsf{MC\text{-}IK\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) = q_E \cdot \mathsf{Adv}^{\mathsf{IK\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B}).$$

□

Table of Contents

Asymmetric Message Franking in the Plain Model: Generic and Efficient Constructions 1
Milan Gonzalez-Thauvin and Keitaro Hashimoto
1 Introduction 1
1.1 Our Contributions 3
1.2 Technical Overview 3
1.3 More Related Work 7
1.4 Paper Organization 8
2 Preliminaries 8
2.1 Notations 8
2.2 Public-Key Encryption 8
2.3 Signature Scheme 9
2.4 The ZAP Proof System 10
2.5 Collision-Resistant Hash Functions 11
2.6 Preliminaries for Efficient Constructions 11
3 Recap Asymmetric Message Franking 13
3.1 Syntax 13
3.2 Security Definitions 14
4 Generic Construction of AMF in the Plain Model 17
4.1 Proposed Construction $\mathsf{AMF}_{\mathsf{GC}}$ 17
4.2 Security Proofs for $\mathsf{AMF}_{\mathsf{GC}}$ 18
4.3 Instantiations of Generic Construction 23
5 Efficient Construction of AMF from Pairings 23
5.1 Proposed Construction $\mathsf{AMF}_{\mathsf{EC}}$ 23
5.2 Security Proofs for $\mathsf{AMF}_{\mathsf{EC}}$ 24
6 Asymmetric Message Franking from Plain LWE 28
6.1 Recap ZAP for $\mathsf{NP} \cap \mathsf{coNP}$ 28
6.2 Proposed Construction $\mathsf{AMF}'_{\mathsf{GC}}$ 29
6.3 Security Proofs for $\mathsf{AMF}'_{\mathsf{GC}}$ 30
6.4 Instantiation of $\mathsf{AMF}'_{\mathsf{GC}}$ 32
A Missing Proofs for $\mathsf{AMF}_{\mathsf{GC}}$ and $\mathsf{AMF}_{\mathsf{EC}}$ 34
A.1 Sender Binding for $\mathsf{AMF}_{\mathsf{GC}}$ 34
A.2 Receiver Compromise Deniability for $\mathsf{AMF}_{\mathsf{GC}}$ 35
A.3 Judge Compromise Deniability for $\mathsf{AMF}_{\mathsf{GC}}$ 37
A.4 Probability That the Challenger Aborts in Game $\mathsf{G}_{2.1}$ (Receiver Binding of $\mathsf{AMF}_{\mathsf{EC}}$) . . 38
A.5 Sender Binding for $\mathsf{AMF}_{\mathsf{EC}}$ 39
A.6 Deniability Properties for $\mathsf{AMF}_{\mathsf{EC}}$ 40
B Other Security Notions for AMF 41
B.1 AMF with Strong Authentication 41
B.2 AMF with Unframeability 43
C Generic Construction of AMF with Unframeability 44
D AMF from Multi-Designated Verifier Signature 46
D.1 Recap Multi-Designated Verifier Signature 47
D.2 Generic Construction of AMF from MDVS 48
E On Multi-Challenge IK-CPA Security 48