Online Deniability for Multiparty Protocols with Applications to Externally Anonymous Authentication

Alonso González-Ulloa and Alejandro Hevia

Dept. of Computer Science, University of Chile
Blanco Encalada 2120, 3er piso, Santiago, Chile
alogonza@ing.uchile.cl, ahevia@dcc.uchile.cl

Abstract. In the problem of *anonymous authentication* (Boneh et al. CCS 1999), a sender wishes to authenticate a message to a given recipient in a way that preserves anonymity: the recipient does not know the identity of the sender and only is assured that the sender belongs to some authorized set. Although solutions for the problem exist (for example, by using ring signatures, e.g. Naor, Crypto 2002), they provide no security when the anonymity set is a singleton. This work is motivated by the question of whether there is any type of anonymity possible in this scenario. It turns out that we can still protect the identity of *all senders* (authorized or not) if we shift our concern from preventing the identity information be revealed to the recipient to preventing it could be revealed to an external entity, other than the recipient. We define a natural functionality which provides such guarantees and we denote it by $\mathcal{F}_{\mathsf{eaa}}$ for *externally anonymous authenticated channel*.
We argue that any realization of $\mathcal{F}_{\mathsf{eaa}}$ must be deniable in the sense of Dodis et al. TCC 2009. To prove the deniability of similar primitives, previous work defined *ad hoc* notions of deniability for each task, and then each notion was showed equivalent to realizing the primitive in the *Generalized Universal Composability* framework (GUC, Canetti et al. TCC 2007). Instead, we put forward the question of whether deniability can be defined independently from any particular task. We answer this question in the affirmative providing a natural extension of the definition of Dodis et al. for arbitrary multiparty protocols. Furthermore, we show that a protocol satisfies this definition if an only if it realizes the ideal functionality $\mathcal{F}_{\mathsf{den}}$ in the GUC framework. This result enables us to prove that most GUC functionalities we are aware of (and their realizations) are deniable.
We conclude by applying our results to the construction of a deniable protocol that realizes $\mathcal{F}_{\mathsf{eaa}}$.

1 Introduction

Consider an online forum where users can post sensitive data on a server and may be concerned about being linked to their posts – a health forum, for example. In order to organize posts by relevance, the server may want to rank posts according to user reputation, which we assume is measured by some arbitrary number. Arguably, the forum's owners would like to implement an *anonymous authentication* protocol [2], where users can be authenticated to a server while maintaining their anonymity.

In the typical solution to the anonymous authentication problem "the authentication protocol carried out between the user and the server does not identify the user" [2]. Instead, the server verifies that the user belongs to some *authorized group*, such as the group of users with the same reputation. But what if the reputation is sufficiently "fine grained"? On the worst case, if each user has a different reputation level then the authorized group is of size 1. In such a case, it is unavoidable that the server will learn the user's identity. Even worse, any party may learn the user's identity since the security notion does not rule out public verification of the authentication process. Clearly, anonymous authentication does not suffice for our goal.

Can we still provide some meaningful form of anonymity for the user? It turns out that the answer to this question is positive if we are willing to consider a different form of anonymity.

Suppose we are no longer concerned about whether the server would identify the user but whether an eavesdropper – or any party *other than the server* – can be certain that a given user indeed participated. Moreover, we would like to preserve anonymity even in the case the server is malicious and may use any strategy to *prove* to the external party that the given user indeed participated in the protocol. We seek guarantees that no such server can succeed and call this goal *externally anonymous authentication* (EAA). More precisely, an EAA protocol should satisfy the following two requirements

Secure authentication: No user should be able to fool the server about its identity (except with very small probability).

External anonymity: Users can not be linked to their messages by parties other than the server, even when the server is malicious.

Yet, rigorously formalizing the above is tricky. In order to gain some intuition, it is helpful to abstract the desired properties using a "functionality" such as those used in secure function evaluation definitions [29]. A good starting point is the functionality for ideally authenticated channels $\mathcal{F}_{\mathsf{auth}}$ [4], which allows a sender S to transmit an authenticated message to a receiver R. The functionality is essentially $\mathcal{F}_{\mathsf{auth}}(m) = (m, m)$, where the input is given by S, the first output is R's output, and the second output is the adversary's output. In our multiparty setting, with users $P_1, P_2, \ldots, P_n$ and server S, it becomes $\mathcal{F}_{\mathsf{auth}}(m_1, m_2, \ldots, m_n) = ((m_1, m_2, \ldots, m_n), (m_1, m_2, \ldots, m_n))$. Certainly, $\mathcal{F}_{\mathsf{auth}}$ provides secure authentication but no anonymity. In order to provide anonymity, we must hide the parties identities. We do so by setting the adversary's output to the lexicographically sorted list $\mathsf{Sort}(m_1, m_2, \ldots, m_n)$. Consequently, the semantics of the EAA primitive can be captured by

$$\mathcal{F}_{\mathsf{eaa}}(m_1, m_2, \ldots, m_n) \stackrel{\text{def}}{=} ((m_1, \ldots, m_n), \, \mathtt{Sort}(m_1, \ldots, m_n)) \, .$$

The fact that the server's output contains a message m in the i-th position is not a proof of the participation of the i-th user since the server's output can be *produced without the participation of i-th user.*

This last property of $\mathcal{F}_{\mathsf{eaa}}$ is captured by the concept of *deniability* (originally put forward by Dwork et al. [16] and refined by Dodis et al. [11, 13]). Roughly speaking, a protocol is deniable if the server nor any other participant can prove that a particular party participated in the protocol.

Can we realize $\mathcal{F}_{\mathsf{eaa}}$ with some protocol π while preserving deniability? Here the cryptographic framework where security is proven becomes crucial. Indeed, it has been shown that a proper framework needs to be used in order to preserve deniability [6, 11, 14, 22]. In fact, the Generalized Universal Composability (GUC) framework from Canetti et al. [6] was specifically proposed to preserve deniability, although this work does not formalize a "general" notion for this concept.

So far, it has been shown that the GUC framework captures deniability only for *specific tasks*, via the following approach: Given a task T, define a deniability experiment for T and then show that a protocol satisfies this experiment if and only it realizes the ideal functionality for T in the GUC framework. This approach have been successfully done for *authentication* and *key exchange* by Dodis et al. [11], and for *zero-knowledge* by Dodis et al. [13].

Defining Deniability for Arbitrary Multiparty protocols: Instead of repeating the previous approach for external anonymous authentication, we put forward the question of whether deniability can be defined independently from any particular task. We answer this question in the affirmative by providing a natural extension of the definition of Dodis et al. for arbitrary multiparty protocols.

To gain some intuition on our definition, we recall (a simplified version of) the deniable zero-knowledge definition of Dodis et al. [13]. There, a prover P proves a true statement x to a verifier V in a setting where both of them are part of a network environment which includes some trusted parties or setup (for example, PKI). Now, let π be a protocol that implements a deniable zero-knowledge proof from P to V. The protocol is required to be complete, sound, and zero-knowledge. According to Dodis et al. a protocol π is an online deniable zero-knowledge protocol if, when $x \in L$, it can be simulated only with access to the statement x and the public information of trusted parties, but without participation of P nor V.

Interestingly, even if one abstracts out the properties which are specific to the *zero-knowledge* task (completeness, soundness, and zero-knowledge), we still get a meaningful security property. Namely: "*A protocol π is deniable if and only if it can be simulated given only its inputs and public information, but without participation of any honest party.*"

Note that neither *correctness* nor *privacy* are required from a protocol with this property, and therefore, a protocol achieving it might be trivial to construct. But, in that case, the protocol will probably not realize any interesting task. Deniable protocols become interesting when they also realize some non-trivial functionality.

We formalize *online deniability* by proposing an experiment – similar to the one by Dodis et al. [11, 14] – which, contrarily to those works, does not require privacy nor correctness. We show that an arbitrary multiparty protocol satisfies our deniability definition if an only if it realizes a fixed ideal functionality $\mathcal{F}_{\mathsf{den}}$ in the GUC framework. Since a functionality is a special case of a protocol, this result also enables us to prove that most GUC functionalities we are aware of (and their realizations) are deniable. Yet, not all the functionalities are deniable, as we discuss below.

We gain further confidence in our definition by noting that a similar characterization of *Bi-deniability in the LUC framework* [9] can be captured with the LUC equivalent of $\mathcal{F}_{\mathsf{den}}$. In the process, we take the opportunity to correct a small mistake in their notion, as we explain in next section.

The Non-Triviality of Our Definition of Deniability: It is easy to see that there are protocols that cannot be deniable. Consider one that uses a UF-CMA public-key signature scheme [18] where a signature for an honest user and her verification key are public, cannot be deniable – any simulation from the inputs only will contradict the UF-CMA property. On the other hand, the protocols proposed in [11, 13] can be shown deniable under our definition. This provides some support to our characterization. If we consider functionalities, on the other hand, the landscape is not so clear. One functionality that appears to be *not deniable* is $\mathcal{F}_{\mathsf{keia}}$ [11]. Whether we can characterize the class of meaningful *non-deniable* functionalities is an interesting open problem.

An external anonymous authentication protocol: We conclude applying the definitional tools to our motivating problem. Concretely, we formulate a (not simplified) functionality for externally anonymous authenticated channel $\mathcal{F}_{\mathsf{eaa}}$, and we show that there is a simple yet effective construction. Our construction combines an anonymous channel and the deniable authentication protocol secure against static adversaries of Dodis et al. [11] with some modifications. We prove that our construction is secure under the Decisional Diffie-Hellman (DDH) assumption. A technical byproduct of this proof is an improved bound on the reduction from the so-called *Multi Decisional Diffie Hellman* (Multi-DDH) to DDH. This result may be of independent interest.

1.1 Related Work

Deniability, Offline and Online: Deniable authentication was first defined by Dwork et al. in their seminal work on *Concurrent Zero-Knowledge* [16]. Later, Dodis et al. refined the notion, introducing the concept of *online* deniability. They define and study the concept in the context of authentication, identification, and Key Exchange [11], and Zero Knowledge [14]. They showed that, for each of these tasks, deniability is equivalent to GUC-realizing the corresponding ideal functionalities. As consequence, their definition implies security under general concurrent composition.

Several protocols that achieve deniability exist for different tasks: deniable authentication and identification (e.g. [11, 15–17, 21, 23, 24]), deniable key exchange (e.g. [25, 28]), and deniable zero knowledge [14, 22]. All of them, however, fall in one of the two categories: (offline) deniable and online-deniable. Most of the protocols proposed before [11, 14] are offline deniable (with the sole exception of HMQV [24] as noted by Dodis et al. [11]). In our work, we focus on the strictly stronger requirement of online deniability.

Online Deniable Authentication: Dodis et al. presented an online-deniable authentication with respect to static adversaries [11]. They also showed the impossibility of the same task with respect to adaptive adversaries. Our paper builds on these results, first by using a modified version of their protocol as the underlying authentication procedure in our anonymous authentication protocol, and second, by restricting our adversarial model to static adversaries.

Anonymous Authentication: Most work on the anonymous authentication literature relies on *ring signatures* in order to make the user identify anonymously as member of an authorized group [2, 12, 20]. Unfortunately, ring signatures cannot provide meaningful anonymity guarantees when the ring size is one. Naor also proposed the notion of *deniable ring signatures* which combines deniability and ring signatures [21]. However, as noted by Dodis et al., Naor's protocol cannot be online deniable as long as verifiers do not register public keys [11]. We also note that, in our setting, deniable ring signature do not necessarily provide anonymity among users of different rings (in our example users with different reputations).

Bi-Deniability and Deniable Encryption: Canetti and Vald [9, 10] introduced the notion of *Bi-Deniability* with many similarities to the definition of deniability of Dodis et al. [11] as well as to the one of our work. Their definition is motivated by a different problem (capturing collusion-freeness and game-theoretic solution concepts) in a UC variant called Local Universal Composability (LUC). They showed that a two-party protocol is Bi-Deniable if and only if it LUC-realizes $\mathcal{F}_{\overline{\mathsf{auth}}}$, the LUC authentication functionality (defined in [9, 10]).

Unfortunately, there is a minor technical issue in their result that we discovered when comparing their notion with ours. Functionality $\mathcal{F}_{\overline{\mathsf{auth}}}$ requires *a specific correctness property*: if the sender provides m to the protocol, the receiver receives the same m. This is not the case in other deniable protocols (zero-knowledge for example) so the their proposed characterization is incorrect. We conclude that the right characterization is not $\mathcal{F}_{\overline{\mathsf{auth}}}$ but $\mathcal{F}_{\overline{\mathsf{den}}}$, the LUC version of our $\mathcal{F}_{\mathsf{den}}$. A more detailed description of the problem and a way to patch this issue is discussed in the full version of this work.

Another meaning for deniability is that of *deniable encryption* (e.g. [7]). A deniable encryption scheme must allow the receiver (or the sender) to deny that the content of a ciphertext is a given plaintext. In contrast, in our definition an encryption scheme $\mathcal{E}$ is essentially deniable if a party executing $\mathcal{E}$ can deny at all that the execution is taking place.

SIMPLIFYING CONSTRUCTIONS: Ishai et al. [19] introduced an anonymity functionality Anon (which provides *sender and receiver anonymity*) in order to construct secure channels from anonymous channels. Interestingly, our ideal functionality for anonymous authenticated channels $\mathcal{F}_{\mathsf{eaa}}$ can be used to significantly simplify their construction. Due to space restrictions, we defer this discussion to the full version of this work.

1.2 Our Contribution

In this work, we define the concept of *online deniability* independently of any specific task. We also provide an alternative characterization by showing that a protocol is online deniable if and only if it GUC realizes a given ideal functionality $\mathcal{F}_{\mathsf{den}}$. We note that this result also applies to Bi-deniability in the LUC framework [9, 10].

We extend the characterization of deniable protocols (and functionalities) by showing that most ideal functionalities are deniable. Our result implies all previous results on deniability we are aware of: deniability for GUC-secure Zero Knowledge protocols [13, Thm.8], deniability for GUC-secure authentication [11, Prop.1], and deniability of GUC-secure key exchange and identification from [11].

Finally, we apply our results on deniability to realize the functionality that motivated this work, which combines anonymous and authenticated channels. To prove the security of the protocol, we use the fact that the Multi-Decisional Diffie-Hellman ($\mathsf{Multi\text{-}DDH}$) assumption follows from the Decisional Diffie-Hellman (DDH) assumption [3]. In the process, we give an improved bound on the $\mathsf{Multi\text{-}DDH}$ vs. DDH relation: the tightness of our reduction is linear compared to quadratic from the best known result [3].

1.3 Organization

We briefly review UC and GUC frameworks in Section 2. We then define deniability, provide a characterization of deniable protocols on Section 3. Finally, in Section 4, we conclude by defining an ideal functionality for externally anonymous authenticated channel $\mathcal{F}_{\mathsf{eaa}}$, designing a protocol for it, and formally proving it GUC realizes $\mathcal{F}_{\mathsf{eaa}}$ in the $\bar{\mathcal{G}}_{\mathsf{krk}}$-hybrid model.

2 Preliminaries

MODEL: We define and prove the security guarantees of our protocol in the Generalized Universal Composability (GUC) framework as described in [6]. We also use GUC to provide an alternative characterization of deniability. Since GUC is a generalization of the Universal Composability (UC) framework [5], we briefly and informally outline both of them here. A more detailed exposition can be found in [4, 5] for UC and [6, 26] for GUC.

THE UNIVERSAL COMPOSABILITY FRAMEWORK (UC): In the UC framework, the desired properties of cryptographic protocols are defined in terms of tasks or *functionalities*. A functionality is a "trusted third party" that obtains inputs directly from the parties, performs certain instructions on these inputs, and provides the appropriate outputs back to the parties. A protocol securely implements a given cryptographic task if running the protocol against a realistic (i.e. real-life) adversary "emulates" the execution of an ideal process. In the ideal process, the task is computed by the trusted party directly interacting with the parties against a very limited adversary called the *ideal* *adversary.* The notion of "emulation" involves a distinguisher $\mathcal{Z}$ which not only provides the inputs to the parties and sees their outputs but also interacts with the adversary, with the goal of telling whether it is interacting with a real protocol and the real-life adversary, or with the functionality and the ideal-adversary. Good emulation means no such environment is successful. See details and proofs in [5]. We denote $\mathsf{EXEC}_{\mathcal{Z},\mathcal{A},\pi}(k)$ the distribution of the output of environment $\mathcal{Z}$ when executed with adversary $\mathcal{A}$, protocol π, and security parameter k. If the protocol assumes the presence of some functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_m$, the output of the environment is denoted by $\mathsf{EXEC}^{\mathcal{F}_1,\ldots,\mathcal{F}_m}_{\mathcal{Z},\mathcal{A},\pi}(k)$ or simply $\mathsf{EXEC}^{\mathcal{F}_1,\ldots,\mathcal{F}_m}_{\mathcal{Z},\mathcal{A},\pi}$, when considering the associated family of distributions. Similarly, in the ideal world, the output of $\mathcal{Z}$ executed with $\mathcal{S}$ and the ideal protocol with functionality $\mathcal{F}$ is denoted $\mathsf{EXEC}^{\mathcal{F}}_{\mathcal{Z},\mathcal{S},\mathsf{IDEAL}_{\mathcal{F}}}$.

The main advantage of UC security is that it composes, that is, the security of any protocol is maintained even when the protocol is being executed concurrently with other, possibly adversarially chosen, protocols. But there is a restriction, the protocol must be **subroutine respecting**. This means that the protocol itself and all its subroutines do not provide any input or output to any other protocol. In other words, the protocol and the subroutines called by the protocol are independent of all other protocols and cannot share state with other protocols.

THE GENERALIZED UC FRAMEWORK (GUC): A UC-secure protocol must be *subroutine respecting*, otherwise the UC-theorem may not be true. Furthermore, a subroutine respecting protocol is unrealistic when *setup assumptions* are required (which is the case of more interesting functionalities [8]), as noted by Canetti et al. [6].

In the GUC framework [6], subroutine respecting protocols are extended so they can be $\bar{\mathcal{G}}$-*subroutine respecting*: protocols can be subroutine respecting except that are allowed to call the *shared functionality* $\bar{\mathcal{G}}$. The GUC framework models $\bar{\mathcal{G}}$-subroutine respecting protocols by allowing the environment to impersonate dummy parties connected to the shared functionality. This small change lets the environment simulate protocols that share state with the analyzed protocol.

The definitions of execution and emulation in GUC are almost identical to those of UC but notation changes. From now on, we denote by EXEC both the UC-execution as well as the GUC-execution, and by UC-emulation we also refer to GUC-emulation. Analogously to UC, it is possible to prove a composition theorem for $\bar{\mathcal{G}}$-subroutine respecting protocols.

3 Online Deniability

In this section, we generalize the notions of deniability for specific tasks available in the literature, namely *online authentication deniability*, *online deniable key-exchange* [11], and *online deniable zero-knowledge* [13], to any task. We call our definition simply *online deniability.* Our definition follows that in [11, 13] but it does not require correctness nor privacy.

The lack of correctness implies that, from a definitional standpoint, we do not commit to any specific correctness property. In contrast, in deniable authentication an honest receiver is required to output the same message sent by an honest sender. Similarly, in deniable zero-knowledge, both completeness and soundness are required. The lack of privacy implies that we do not guarantee any privacy property in a deniable protocol, in contrast to deniable zero-knowledge where the witness remains hidden to any other party than the prover.

THE PLAYERS: We consider a judge $\mathcal{J}$, an informant $\mathcal{I}$, misinformant $\mathcal{M}$, and a global setup functionality $\bar{\mathcal{G}}$. Also, we assume there are parties $P_1, \ldots, P_n$ running an *arbitrary* distributed protocol π.[1] All parties can communicate with a shared functionality $\bar{\mathcal{G}}$ (which may model, for example, availability of a PKI) or with $\mathcal{I}$. The entities $\mathcal{J}$, $\mathcal{I}$ and $\mathcal{M}$ have also access to the public interface of $\bar{\mathcal{G}}$, and also to the secret interface but only in the case of corrupted parties.

THE REAL WORLD: In the *real world*, the first entity activated is the judge $\mathcal{J}$. It then activates $\mathcal{I}$ while providing $\mathcal{I}$ with the input of each party participating in π. Upon activation, party $\mathcal{I}$ forwards (possibly different) inputs to the actual parties, witnesses (monitors) the execution of protocol π, and interact with $\mathcal{J}$ (possibly sending evidence that an execution of π is taking place). The informant $\mathcal{I}$ can adaptively corrupt parties if instructed by $\mathcal{J}$. When it does so, the entire internal state of the corrupted party is revealed to $\mathcal{I}$, who takes control of the corrupted party. Finally, at some point of the execution $\mathcal{J}$ is activated, outputs a single bit and halts. Given an integer k, the security parameter, we denote by $\mathsf{RealDen}^{\bar{\mathcal{G}}}_{\mathcal{J},\mathcal{I},\pi}(k)$ the output of $\mathcal{J}$ executed in the real world with informant $\mathcal{I}$, shared functionality $\bar{\mathcal{G}}$, and protocol π.

THE SIMULATED WORLD: In the *simulated world*, $\mathcal{J}$ is the first party activated. Then it activates $\mathcal{M}$ with the inputs of each party. Contrarily to the real world, in the simulated world honest parties cannot be activated by $\mathcal{M}$, and thus actual parties never executes the protocol π. Nonetheless, $\mathcal{M}$ is also able to adaptively corrupt parties. Corruption works exactly as in the real world. Finally, at some point of the execution $\mathcal{J}$ is activated, outputs a single bit and halts. Given an integer k, the security parameter, we denote by $\mathsf{SimDen}^{\bar{\mathcal{G}}}_{\mathcal{J},\mathcal{M}}(k)$ the output of a judge $\mathcal{J}$ executed in the simulated world with misinformant $\mathcal{M}$, and setup functionality $\bar{\mathcal{G}}$.

Definition 1. *Let π be an arbitrary $\bar{\mathcal{G}}$-subroutine respecting multiparty protocol. We say that protocol π is online deniable if for all judge $\mathcal{J}$ and all informant $\mathcal{I}$ there exists a misinformant $\mathcal{M}$ such that* $\mathsf{RealDen}^{\bar{\mathcal{G}}}_{\mathcal{J},\mathcal{I},\pi} \approx \mathsf{SimDen}^{\bar{\mathcal{G}}}_{\mathcal{J},\mathcal{M}}$.

3.1 Online deniability in GUC

We now show that deniability in the real world can be seen as a syntactic transformation of the GUC real world experiment – any environment and adversary can be simulated with a judge and an informant, and vice versa. The proof is in Appendix A.1.

Lemma 1. *For each judge $\mathcal{J}$ and for each informant $\mathcal{I}$ there exists an environment $\mathcal{Z}^{\mathcal{J}}$ and an adversary $\mathcal{A}^{\mathcal{I}}$ such that for all protocol π executed in the $\bar{\mathcal{G}}$-hybrid model* $\mathsf{RealDen}^{\bar{\mathcal{G}}}_{\mathcal{J},\mathcal{I},\pi} \equiv \mathsf{EXEC}^{\bar{\mathcal{G}}}_{\mathcal{Z}^{\mathcal{J}},\mathcal{A}^{\mathcal{I}},\pi}$. *Conversely, for all environment $\mathcal{Z}$ and for all adversary $\mathcal{A}$ there exists a judge $\mathcal{J}^{\mathcal{Z}}$ and an informant $\mathcal{I}^{\mathcal{A}}$ such that* $\mathsf{EXEC}^{\bar{\mathcal{G}}}_{\mathcal{Z},\mathcal{A},\pi} \equiv \mathsf{RealDen}^{\bar{\mathcal{G}}}_{\mathcal{J}^{\mathcal{Z}},\mathcal{I}^{\mathcal{A}},\pi}$

The functionality $\mathcal{F}_{\mathsf{den}}$, defined in figure 1, provides all the necessary information to run a misinformant, and thus a simulator can perfectly simulate the misinformant. We now show a result equivalent to the one in [11].

Theorem 1. *A $\bar{\mathcal{G}}$-subroutine respecting protocol π is online deniable if and only if it GUC-realizes the ideal functionality $\mathcal{F}_{\mathsf{den}}$ in the $\bar{\mathcal{G}}$-hybrid model.*

[1] Also, we allow algorithm $\mathcal{J}$ to take an auxiliary input $z \in \{0,1\}^*$ and to run in polynomial time with respect to $|z|$ and k. Similarly, due to some technicalities, we require the informant and misinformant to be probabilistic polynomial time in the sense of UC adversaries [4, version 2013].

The ideal functionality $\mathcal{F}_{\mathsf{den}}$ running with parties $\tilde{P}_1, \ldots, \tilde{P}_n$ proceeds as follows:

1. When x_i is received from party $\tilde{P}_i$, send $(\mathtt{Input}, \tilde{P}_i, x)$ to the adversary $\mathcal{S}$.
2. When $(\mathtt{Output}, \tilde{P}_i, y)$ is received from the adversary $\mathcal{S}$, send y to $\tilde{P}_i$.

Fig. 1. The ideal functionality $\mathcal{F}_{\mathsf{den}}$

Proof. We first prove the first implication, namely: if π is online deniable then π GUC-emulates $\mathcal{F}_{\mathsf{den}}$. Let $\mathcal{Z}$ be an environment and $\mathcal{A}$ be an adversary. By lemma 1, there exists a judge $\mathcal{J}^{\mathcal{Z}}$ that simulates $\mathcal{Z}$ such that $\mathsf{RealDen}^{\bar{\mathcal{G}}}_{\mathcal{J}^{\mathcal{Z}},\mathcal{I}^{\mathcal{A}},\pi} \equiv \mathsf{EXEC}^{\bar{\mathcal{G}}}_{\mathcal{Z},\mathcal{A},\pi}$. By deniability of π, there exists a misinformant $\mathcal{M}$ such that $\left|\Pr\left[\mathsf{RealDen}^{\bar{\mathcal{G}}}_{\mathcal{J}^{\mathcal{Z}},\mathcal{I}^{\mathcal{A}},\pi} = 1\right] - \Pr\left[\mathsf{SimDen}^{\bar{\mathcal{G}}}_{\mathcal{J}^{\mathcal{Z}},\mathcal{M}} = 1\right]\right| = \epsilon$, where ϵ is negligible in the security parameter.

The ideal adversary $\mathcal{S}^{\mathcal{M}}$ with access to the ideal functionality $\mathcal{F}_{\mathsf{den}}$ proceeds as follows:

1. Simulate $\mathcal{M}$.
2. When $(\mathtt{Input}, \tilde{P}_i, x)$ is received from $\mathcal{F}_{\mathsf{den}}$, send $(\tilde{P}_i, x)$ to $\mathcal{M}$.
3. When $\mathcal{M}$ sends $(\tilde{P}_i, y)$, send $(\mathtt{Output}, \tilde{P}_i, y)$ to $\mathcal{F}_{\mathsf{den}}$.
4. When $\mathcal{M}$ corrupts party P_i, party $\tilde{P}_i$ is corrupted and the internal state of $\tilde{P}_i$ is revealed to $\mathcal{M}$.
5. When x is received from $\mathcal{Z}$, send x to $\mathcal{M}$.
6. When x is received from $\mathcal{M}$, send x to $\mathcal{Z}$.

Fig. 2. The ideal adversary $\mathcal{S}^{\mathcal{M}}$

Consider now the ideal adversary $\mathcal{S}^{\mathcal{M}}$, with (oracle) access to $\mathcal{M}$, as defined in figure 2. Note that in both, $\mathsf{SimDen}^{\bar{\mathcal{G}}}_{\mathcal{J}^{\mathcal{Z}},\mathcal{M}}$ and $\mathsf{EXEC}^{\bar{\mathcal{G}},\mathcal{F}_{\mathsf{den}}}_{\mathcal{Z},\mathcal{S}^{\mathcal{M}},\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{den}}}}$, $\mathcal{Z}$ is "hardwired" with $\mathcal{M}$. The only difference is that in $\mathsf{EXEC}^{\bar{\mathcal{G}},\mathcal{F}_{\mathsf{den}}}_{\mathcal{Z},\mathcal{S}^{\mathcal{M}},\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{den}}}}$ the inputs are directly given by $\mathcal{Z}$ to the parties, but in $\mathsf{SimDen}^{\bar{\mathcal{G}}}_{\mathcal{J}^{\mathcal{Z}},\mathcal{M}}$ the inputs are forwarded from $\mathcal{J}^{\mathcal{Z}}$ to the parties by $\mathcal{M}$. In both cases the input given to each party is the same, and consequently the view of $\mathcal{Z}$ is the same. Therefore $\mathsf{SimDen}^{\bar{\mathcal{G}}}_{\mathcal{J}^{\mathcal{Z}},\mathcal{M}} \equiv \mathsf{EXEC}^{\bar{\mathcal{G}},\mathcal{F}_{\mathsf{den}}}_{\mathcal{Z},\mathcal{S}^{\mathcal{M}},\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{den}}}}$, which implies that $\left|\Pr\left[\mathsf{EXEC}^{\bar{\mathcal{G}}}_{\mathcal{Z},\mathcal{A},\pi} = 1\right] - \Pr\left[\mathsf{EXEC}^{\bar{\mathcal{G}},\mathcal{F}_{\mathsf{den}}}_{\mathcal{Z},\mathcal{S}^{\mathcal{M}}} = 1\right]\right| = \epsilon$. Therefore, π GUC-emulates $\mathcal{F}_{\mathsf{den}}$.

The other direction is similar. Suppose that π GUC-emulates $\mathcal{F}_{\mathsf{den}}$ in the $\bar{\mathcal{G}}$-hybrid model. Let $\mathcal{J}$ be a judge and $\mathcal{I}$ be an informant, by lemma 1 there exists an environment $\mathcal{Z}^{\mathcal{J}}$ and an adversary $\mathcal{A}^{\mathcal{I}}$ such that $\mathsf{RealDen}^{\bar{\mathcal{G}}}_{\mathcal{J},\mathcal{I},\pi} \equiv \mathsf{EXEC}^{\bar{\mathcal{G}}}_{\mathcal{Z}^{\mathcal{J}},\mathcal{A}^{\mathcal{I}}}$. By hypothesis there exists a simulator $\mathcal{S}$ such that the advantage of $\mathcal{Z}^{\mathcal{J}}$ distinguishing between π and $\mathcal{F}_{\mathsf{den}}$ is negligible. Similarly to the first implication, a misinformant $\mathcal{M}^{\mathcal{S}}$ can simulate $\mathcal{S}$ for $\mathcal{J}$. This is indistinguishable from $\mathcal{I}$ to $\mathcal{J}$, and therefore π is online deniable.

A Sufficient Condition for Online Deniability: It seems that deniability is a concern not important per se, it becomes important when one wishes it to be added to some existent task, say, GUC-realizing some functionality $\mathcal{F}$. It appears that a large class of functionalities are indeed deniable ($\mathcal{F}_{\mathsf{auth}}$ and $\mathcal{F}_{\mathsf{zk}}$ are examples). Indeed, this is simply a consequence of the fact that most ideal functionalities are subroutine respecting.

Corollary 1. *Let π be a subroutine respecting protocol, then π is online deniable. Furthermore, if ρ is $\bar{\mathcal{G}}$-subroutine respecting but only have access to the public interface of $\bar{\mathcal{G}}$ (that is the interface that is accessible by the adversary), then ρ is also online deniable.*

Proof. The view of a subroutine respecting protocol is completely determined by its inputs and randomness, thus π can be perfectly simulated only with access to the inputs returned by $\mathcal{F}_{\mathsf{den}}$. Thus π GUC-emulates $\mathcal{F}_{\mathsf{den}}$ and, by Theorem 1, π must be deniable. Clearly, this simulation can easily be extended to the case the protocol requires access to the shared functionality $\bar{\mathcal{G}}$ so the adversary only sees the public interface.

By the transitivity of the GUC-emulation relation, we remark that a proof that a protocol GUC-realizes some deniable ideal functionality also implies that the protocol is deniable.

4 Externally Anonymous Authenticated Channels

4.1 The $\mathcal{F}_{\mathsf{eaa}}$ ideal functionality

An "anonymous authenticated channel" should allow parties to send authenticated messages to any other party without revealing their identities to anyone except the receiver. Since the receiver knows the sender identity, we call this variant *external anonymity*. We formally define an externally-anonymous authenticated channel via an ideal functionality called $\mathcal{F}_{\mathsf{eaa}}$ (fig. 3) which requires the shared functionality $\bar{\mathcal{G}}_{\mathsf{krk}}$ (fig. 5). Note that the functionality $\mathcal{F}_{\mathsf{eaa}}$ reveals just the value of each sent message to the adversary but not the identities related to those messages. This holds while the receiver of the message is not corrupt – but even in that situation the information revealed by $\mathcal{F}_{\mathsf{eaa}}$ is completely simulatable by *any one*. This holds because functionality $\mathcal{F}_{\mathsf{eaa}}$ is indeed online deniable, guaranteed by corollary 2.

Functionality $\mathcal{F}_{\mathsf{eaa}}$ parameterized by an integer κ, running with shared functionality $\bar{\mathcal{G}}_{\mathsf{krk}}$, parties $P_1, \dots, P_n$, and adversary $\mathcal{S}$, proceeds as follows.

Initialization: Initialize multisets $M_j \leftarrow \emptyset$ for each $j \in \{1, \dots, n\}$ and $M \leftarrow \emptyset$.
Message reception: Suppose $(\texttt{Send}, m, j)$ is received from $\tilde{P}_i$, do:
1. If $\tilde{P}_i$ or $\tilde{P}_j$ are not registered in $\bar{\mathcal{G}}_{\mathsf{krk}}$ or $i = j$, then send $\perp$ to $\tilde{P}_i$.
2. Else, send $(\texttt{Sent}, \tilde{P}_i)$ to $\mathcal{S}$ and $(\texttt{Sent}, m, j)$ to $\tilde{P}_i$, and let $M_j \leftarrow M_j \uplus \{(m, i)\}$ and $M \leftarrow M \uplus \{m\}$ honest.

Message delivery: Once that $|M| = \kappa$, for each $j \in \{1, \dots, n\}$ send $(\texttt{Messages}, M_j)$ to $\tilde{P}_j$, and send $(\texttt{Messages}, M)$ to $\mathcal{S}$.

Fig. 3. The ideal functionality $\mathcal{F}_{\mathsf{eaa}}$

The integer κ statically fixes the number of exchanged messages per session. We include this parameter in order to define a condition that triggers the delivery of messages. In order to realize $\mathcal{F}_{\mathsf{eaa}}$, we will require that the underlying anonymity functionality also fixes the number of exchanged messages per session to κ.

Note that $\mathcal{F}_{\mathsf{eaa}}$ requires that senders and receivers have been registered on $\bar{\mathcal{G}}_{\mathsf{krk}}$. This condition means that registration on $\bar{\mathcal{G}}_{\mathsf{krk}}$ is required but not provided by $\mathcal{F}_{\mathsf{eaa}}$.

Corollary 2. *The functionality $\mathcal{F}_{\mathsf{eaa}}$ is online deniable.*

Proof. Note that $\mathcal{F}_{\mathsf{eaa}}$ is $\bar{\mathcal{G}}_{\mathsf{krk}}$-subroutine respecting, but all information needed from $\bar{\mathcal{G}}_{\mathsf{krk}}$ is whether or not a party is registered. Certainly, this information is public and thus $\mathcal{F}_{\mathsf{eaa}}$ is online deniable as consequence of Corollary 1.

4.2 Primitives and setup assumptions

In order realize $\mathcal{F}_{\mathsf{eaa}}$, we require an anonymous channel functionality $\mathcal{F}_{\mathsf{anon}}$ and a global PKI modeled by the shared functionality $\bar{\mathcal{G}}_{\mathsf{krk}}$.

Anonymous Channels: Anonymous channels allow users to exchange messages without revealing their identities. We define an anonymous channel functionality $\mathcal{F}_{\mathsf{anon}}$ in fig. 4. We note that $\mathcal{F}_{\mathsf{anon}}$ can be realized, for example, using the universal composable mixnet proposed by Wikström [27].

The Ideal functionality $\mathcal{F}_{\mathsf{anon}}$, parameterized by an integer κ, running parties $P_1, \ldots, P_n$, and ideal adversary $\mathcal{S}$

1. Initialize a list $L \leftarrow \emptyset$.
2. When $(\texttt{Send}, m)$ is received from P_i, append m to the list L. Then, hand $(P_i, \texttt{Sent})$ to $\mathcal{S}$.
3. Once that $|L| = \kappa$, sort the list L lexicographically to form a list L', and hand $(\texttt{Output}, L')$ to $\mathcal{S}$ and to P_i, for $i = 1$ to k.

Fig. 4. The functionality $\mathcal{F}_{\mathsf{anon}}$.

The Key Registration with Knowledge Functionality: In this section, we recall the Φ-Key Registration with Knowledge Functionality [11]. We require that honest parties retrieve secrets keys only using our protocol SIGMIX, defined in Sect. 4.4. In other words, $\Phi = \{\mathsf{SIGMIX}\}$.

Parameterized by a security parameter λ, a protocol (or, more generally, a list of protocols) Φ, and a (deterministic) key generation function Gen, shared functionality $\bar{\mathcal{G}}_{\mathsf{krk}}$ proceeds as follows when running with parties $P_1, \ldots, P_n$:

Registration: When receiving a message $(\texttt{Register})$ from an honest party P_i that has not previously registered, sample $r \stackrel{R}{\leftarrow} \{0,1\}^\lambda$ then compute $(PK_i, SK_i) \leftarrow \mathsf{Gen}^\lambda(r)$ and record the tuple (P_i, PK_i, SK_i).

Corrupt Registration: When receiving a message $(\texttt{Register}, r)$ from a corrupt party P_i that has not previously registered, compute $(PK_i, SK_i) \leftarrow \mathsf{Gen}^\lambda(r)$ and record the tuple (P_i, PK_i, SK_i).

Public Key Retrieval: When receiving a message $(\texttt{Retrieve}, P_i)$ from any party P_j (where $i = j$ is allowed), if there is a previously recorded tuple of the form (P_i, PK_i, SK_i), then return (P_i, PK_i) to P_j . Otherwise return $(P_i, \perp)$ to P_j .

Secret Key Retrieval: When receiving a message $(\texttt{RetrieveSecret}, P_i)$ from a party P_i that is either corrupt or honestly running the protocol code for Φ, if there is a previously recorded tuple of the form (P_i, PK_i, SK_i) then return (P_i, PK_i, SK_i) to P_i. In all other cases, return $(P_i, \perp)$.

Fig. 5. The Φ-Key Registration with Knowledge shared functionality [11].

4.3 Realizing $\mathcal{F}_{\mathsf{eaa}}$

A first attempt to realize $\mathcal{F}_{\mathsf{eaa}}$ is to combine $\mathcal{F}_{\mathsf{anon}}$ with the GUC-secure authentication protocol with respect to static adversaries of Dodis et al. [11]. In their protocol, P_i sends an authenticated message m to P_j by attaching a MAC $\sigma = \mathsf{MAC}_{k_{i,j}}(m)$. The *symmetric shared secret key* $k_{i,j} = k_{j,i}$

can be non interactively computed by P_i using P_j's public key and P_i's secret key (both keys registered in $\bar{\mathcal{G}}_{\mathsf{krk}}$), and it can be also computed by P_j with P_i's public key and P_j's secret key. Note that a corrupted receiver P_j may not convince a third party about the identity of the sender of (m, σ). Indeed, given that the key $k_{i,j}$ used to produce σ can also be produced by P_j, (m, σ) can be obtained without participation of P_i when P_j is corrupted.

We will argue that this approach fails. First, note that nothing in the security definition of a MAC (UF-CMA) prevents the existence of an algorithm Check which, given two MACs values $\sigma = \mathsf{MAC}_k(m)$ and $\sigma' = \mathsf{MAC}_{k'}(m')$, returns 1 if and only if $k = k'$.[2]

Fix an execution where party P_1 sends $\mathtt{0}||\sigma_1 = \mathsf{MAC}_{k_{1,2}}(\mathtt{0})$ to P_2 and then P_2 sends $\mathtt{1}||\sigma_2 = \mathsf{MAC}_{k_{1,2}}(\mathtt{1})$ to P_1. By anonymity, the only information leaked to the adversary should be the set of sent messages $\{\mathtt{0}, \mathtt{1}\}$ and the fact that P_1 sent a message and then P_2 sent another message. The same information is leaked in a similar execution with the only exception that P_2 sends $\mathtt{1}||\sigma_2 = \mathsf{MAC}_{k_{1,3}}(\mathtt{1})$ to another party P_3. An adversary can distinguish these executions by executing $\mathsf{Check}(\sigma_1, \sigma_2)$; in the first execution it should return 1 and in the second 0.

We fix this problem by attaching to the message m the evaluation of a *Variable Input-Pseudo Random Function* (VI-PRF) [1] $E_{k_{i,j}}$ on m. Therefore, if P_i wants to anonymously send an authenticated message m to P_j, P_i simply anonymously sends $m||\sigma$ where $\sigma = E_{k_{i,j}}(m)$. Note that now the simulator is no longer concerned about the keys used to generate σ; it can generate σ by simply picking a fresh random value. Unfortunately, this simulation procedure will be only successful if *the same message is never sent twice to the same receiver*. Indeed, if the same message is sent twice to the same receiver, the simulator can not decide whether to attach the same random value twice or two independently chosen random values.

We can easily get rid of this assumption requiring the senders to attach a *random nonce* to each message, and requiring the receivers to never accept the same message from the same receiver. [3] Note that in a real implementation, this can be also achieved by attaching, for example, the current time to each message.

4.4 The SIGMIX protocol

The SIGMIX protocol runs in the $\mathcal{F}_{\mathsf{anon}}, \bar{\mathcal{G}}_{\mathsf{krk}}$-hybrid model and with static adversaries. We fix the key generation algorithm Gen of $\bar{\mathcal{G}}_{\mathsf{krk}}$ with an algorithm that, using randomness r, sample a random element x from $\mathbb{Z}_q$ and return the pair (g^x, x), where g is the generator of a cyclic group G_q of order q. It is stressed that any other protocol using $\bar{\mathcal{G}}_{\mathsf{krk}}$ might share public keys with SIGMIX. We consider the functionality $\mathcal{F}_{\mathsf{anon}}$ as a traditional UC ideal functionality, meaning that each instance of $\mathcal{F}_{\mathsf{anon}}$ is local to each calling protocol.

The SIGMIX protocol is described in figure 6.

4.5 Proof of security

Before we prove the security of SIGMIX, we bound the relation between the hardness of the *Multi Decisional Diffie-Hellman* assumption and the (standard) *Decisional Diffie-Hellman* assumption [3].

[2] Such a scheme can be easily constructed in the *Random Oracle Model*. Simply attach $H(k)$ to the MAC, i.e. $\mathsf{MAC}'_k(m) = H(k)||\mathsf{MAC}_k(m)$. The value $H(k)$ does not help a forger to break MAC' as long as $H(k)$ is a random value independent from k. Given two MACs $h||\sigma = H(k)||\mathsf{MAC}_k(m)$ and $h'||\sigma' = H(k')||\mathsf{MAC}_{k'}(m')$, the algorithm Check returns 1 if and only if $h = h'$.

[3] In the proof of security we make a stronger requirement to simplify the proof. However, using a random nonce also allows us to get rid of this stronger assumption.

The protocol SIGMIX^{κ}, running with parties $P_1, \dots, P_n$ in the $\mathcal{F}_{\mathsf{anon}}, \bar{\mathcal{G}}_{\mathsf{krk}}$-hybrid model, proceeds as follows:

Sender P_i**:** Each sender P_i and proceeds as follows:

1. Wait for input $(\texttt{Send}, m, j)$.
2. If P_i or P_j are not registered on $\bar{\mathcal{G}}_{\mathsf{krk}}$, or $i = j$, return $\perp$. Compute $k_{i,j} \leftarrow y_j^{x_i}$, and compute $\sigma \leftarrow E_{k_{i,j}}(m)$, where x_i is P_i's secret key and y_j is P_j's public key.
3. Hand $(\texttt{Send}, m||\sigma)$ to $\mathcal{F}_{\mathsf{anon}}$ and return $(\texttt{Sent}, m, j)$.

Receiver P_j**:** Each receiver P_j proceeds as follows:

1. Wait for an input $(\texttt{Output}, L)$ from $\mathcal{F}_{\mathsf{anon}}$.
2. Retrieve $y_1, \dots, y_n$, the public keys of all parties participating in the protocol, and retrieve x_j, P_j's secret key, from $\bar{\mathcal{G}}_{\mathsf{krk}}$. For each $i \in \{1, \dots, n\}$, if $y_i \neq \perp$ and $x_j \neq \perp$ compute the shared secret $k_{i,j} \leftarrow y_i^{x_j}$, otherwise $k_{i,j} \leftarrow \perp$.
3. Let the multiset $M_j \leftarrow \emptyset$. For each $i \in \{1, \dots, n\}$ and each $(m||\sigma) \in L$, if $k_{i,j} \neq \perp$ and $\sigma = E_{k_{i,j}}(m)$, then $M_j \leftarrow M_j \uplus \{(m, i)\}$. Return $(\texttt{Messages}, M_j)$.

Fig. 6. The protocol SIGMIX.

Proposition 1. *Let G_q be a cyclic group where the* DDH *assumption holds, then the* Multi-DDH *assumption also holds, that is:* $(\{g^{x_i}\}_{i=1}^n, \{g^{x_i x_j}\}_{i=1,j>i}^n) \overset{c}{\approx} (\{g^{x_i}\}_{i=1}^n, \{g^{r_{i,j}}\}_{i=1,j>i}^n)$*, where $x_i \in_R G_q$, $r_{i,j} \in_R G_q$ for all i and j. Specifically, for each adversary D attacking* Multi-DDH *there exists an adversary D' attacking* DDH *such that* $n \cdot \mathrm{Adv}_{D'}^{\mathrm{DDH}}(k) \geq \mathrm{Adv}_{D}^{\mathrm{MDDH}}(k)$.

This linear bound on the advantages is tighter than the quadratic bound given in [3] which, to the best of our knowledge, is the best bound known for Multi-DDH. The proof is in Appendix A.2.

The security of SIGMIX is guaranteed by the following theorem which is proven in Appendix A.3.

Theorem 2. *Suppose that $E : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^t$ is a VI-PRF and that* DDH *holds in G_q, then* SIGMIX *GUC-realizes the ideal functionality $\mathcal{F}_{\mathsf{eaa}}$ in the $\bar{\mathcal{G}}_{\mathsf{krk}}$-hybrid model with respect to environments and adversaries that do not play replay attacks. Concretely, let n be the number of participants and k the security parameter of an execution of* SIGMIX*. Then, for all environment $\mathcal{Z}$ and for all adversary $\mathcal{A}$ there exist a simulator $\mathcal{S}$, a* DDH *distinguisher $\mathcal{D}_{\mathsf{DDH}}$, and a distinguisher $\mathcal{D}_{\mathsf{PRF}}$ such that for all k large enough*

$$n \cdot \mathrm{Adv}_{\mathcal{D}_{\mathsf{DDH}}}^{\mathrm{DDH}}(k) + (\kappa+1)\mathrm{Adv}_{E,\mathcal{D}_{\mathsf{PRF}}}^{\mathrm{PRF}}(k) + \frac{\kappa}{2^t} \geq \left| \begin{matrix} \Pr\left[\mathsf{EXEC}_{\mathcal{Z},\mathcal{A},\mathsf{SIGMIX}}^{\bar{\mathcal{G}}_{\mathsf{krk}},\mathcal{F}_{\mathsf{anon}}}(k) = 1\right] - \\ \Pr\left[\mathsf{EXEC}_{\mathcal{Z},\mathcal{S},\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{eaa}}}}^{\bar{\mathcal{G}}_{\mathsf{krk}},\mathcal{F}_{\mathsf{eaa}}}(k) = 1\right] \end{matrix} \right|$$

References

1. M. Bellare, R. Canetti, and H. Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *FOCS*, pages 514–523. IEEE Computer Society, 1996.
2. D. Boneh and M. Franklin. Anonymous authentication with subset queries. In *6th ACM Conference on Computer and Communications Security (CCS '99)*, pages 113–119, New York, Nov. 1999. ACM Press.
3. E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group diffie-hellman key exchange under standard assumptions. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '02, pages 321–336, London, UK, UK, 2002. Springer-Verlag.
4. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000.
5. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In B. Werner, editor, *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS-01)*, pages 136–147, Los Alamitos, CA, Oct. 14–17 2001. IEEE Computer Society.

6. R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with global setup. In S. P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85. Springer, 2007.
7. R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In B. S. Kaliski Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 90–104. Springer-Verlag, 17–21 Aug. 1997.
8. R. Canetti, E. Kushilevitz, and Y. Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In E. Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 68–86. Springer, 2003.
9. R. Canetti and M. Vald. Universally composable security with local adversaries. In I. Visconti and R. D. Prisco, editors, *SCN*, volume 7485 of *Lecture Notes in Computer Science*, pages 281–301. Springer, 2012.
10. R. Canetti and M. Vald. Universally composable security with local adversaries. Cryptology ePrint Archive, Report 2012/117, 2012. http://eprint.iacr.org/.
11. Y. Dodis, J. Katz, A. Smith, and S. Walfish. Composability and on-line deniability of authentication. In O. Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2009.
12. Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 609–626. Springer, 2004.
13. Y. Dodis, V. Shoup, and S. Walfish. Efficient constructions of composable commitments and zero-knowledge proofs. 2008. http://www.shoup.net/papers/gucc.pdf.
14. Y. Dodis, V. Shoup, and S. Walfish. Efficient constructions of composable commitments and zero-knowledge proofs. In D. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 515–535. Springer, 2008.
15. C. Dwork, M. Naor, and A. Sahai. Concurrent zero knowledge. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC-98)*, pages 409–418, New York, May 23–26 1998. ACM Press.
16. C. Dwork and A. Sahai. Concurrent zero-knowledge: Reducing the need for timing constraints. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 442–457. Springer, 1998.
17. I. Goldberg, B. Ustaoglu, M. V. Gundy, and H. Chen. Multi-party off-the-record messaging. In E. Al-Shaer, S. Jha, and A. D. Keromytis, editors, *ACM Conference on Computer and Communications Security*, pages 358–368. ACM, 2009.
18. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988, April.
19. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography from anonymity. In *FOCS*, pages 239–248. IEEE Computer Society, 2006.
20. J. Kilian and E. Petrank. Identity escrow. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 169–185. Springer, 1998.
21. M. Naor. Deniable ring authentication. In M. Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 481–498. Springer, 2002.
22. R. Pass. On deniability in the common reference string and random oracle model. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 316–337. Springer, 2003.
23. M. D. Raimondo and R. Gennaro. New approaches for deniable authentication. In V. Atluri, C. Meadows, and A. Juels, editors, *ACM Conference on Computer and Communications Security*, pages 112–121. ACM, 2005.
24. M. D. Raimondo, R. Gennaro, and H. Krawczyk. Secure off-the-record messaging. In V. Atluri, S. D. C. di Vimercati, and R. Dingledine, editors, *WPES*, pages 81–89. ACM, 2005.
25. M. D. Raimondo, R. Gennaro, and H. Krawczyk. Deniable authentication and key exchange. In A. Juels, R. N. Wright, and S. D. C. di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 400–409. ACM, 2006.
26. S. Walfish. *Enhanced security models for network protocols*. PhD thesis, New York, NY, USA, 2008. AAI3310580.
27. D. Wikström. A universally composable mix-net. In M. Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 317–335. Springer, 2004.
28. A. C. Yao and Y. Zhao. Deniable internet key exchange. In *Applied Cryptography and Network Security*, pages 329–348. Springer, 2010.
29. A. C.-C. Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164. IEEE Computer Society, 1982.

A Proofs

A.1 Proof of Lemma 1

Proof. Let $\mathcal{J}$ be a judge and $\mathcal{I}$ be an informant. Then we can define the environment $\mathcal{Z}^{\mathcal{J}}$ and an adversary $\mathcal{A}^{\mathcal{I}}$, such that the output of $\mathcal{Z}^{\mathcal{J}}$ is equal to the output of $\mathcal{J}$.

The environment $\mathcal{Z}^{\mathcal{J}}$ simulates $\mathcal{J}$ with a direct link to $\mathcal{I}$. This is done by appending the prefix "`Judge-Inform` :" to each message sent by $\mathcal{J}$ to $\mathcal{I}$, then the new message is sent to $\mathcal{A}^{\mathcal{I}}$. Symmetrically, when "`Inform-Judge` :m" is received from $\mathcal{A}^{\mathcal{I}}$, m is forwarded to $\mathcal{J}$ as coming from $\mathcal{I}$. When $(\texttt{Input}, P_i, x)$ is received from $\mathcal{A}^{\mathcal{I}}$, it activates party P_i with input x. When party P_i produces output y, $\mathcal{Z}^{\mathcal{J}}$ sends $(\texttt{Output}, P_i, y)$ to $\mathcal{A}^{\mathcal{I}}$.

The adversary $\mathcal{A}^{\mathcal{I}}$ simulates $\mathcal{I}$ with a direct link to $\mathcal{J}$ symmetrically as it is made by $\mathcal{Z}^{\mathcal{J}}$. When $\mathcal{I}$ produces input x for party P_i, $\mathcal{A}^{\mathcal{I}}$ sends $(\texttt{Input}, P_i, x)$ to $\mathcal{Z}^{\mathcal{J}}$. When $(\texttt{Output}, P_i, y)$ is received from $\mathcal{Z}^{\mathcal{J}}$, $\mathcal{I}$ is informed that P_i outputs y_i. When $\mathcal{I}$ corrupts party P_i, $\mathcal{A}^{\mathcal{I}}$ also corrupts P_i and reveals the internal state of P_i to $\mathcal{I}$. Clearly, the simulation of $\mathcal{J}$ and $\mathcal{I}$ is perfect, and thus the output of $\mathcal{Z}^{\mathcal{J}}$ follows the same distribution that follows the output of $\mathcal{J}$.

The other direction is similar. Let $\mathcal{Z}$ be a judge and $\mathcal{A}$ be an adversary. The judge $\mathcal{J}^{\mathcal{Z}}$ redirects input that $\mathcal{Z}$ sends to the parties to the informant. The informant $\mathcal{I}^{\mathcal{A}}$ simulates the adversary $\mathcal{A}$ and starts an execution of π following the instructions of $\mathcal{J}^{\mathcal{Z}}$. The simulation of $\mathcal{Z}$ and $\mathcal{A}$ is also perfect, and thus the output of $\mathcal{J}^{\mathcal{Z}}$ follows the same distribution that follows the output of $\mathcal{Z}$.

A.2 Proof of Proposition 1

Proof. The proof is based on an hybrid argument, where in each hybrid $\boldsymbol{\chi}_\ell$ the shared keys of parties $P_1, \ldots, P_\ell$ are randomly chosen and other shared keys don't. That is

$$\boldsymbol{\chi}_\ell \stackrel{\text{def}}{=} \left((\{g^{x_i}\}_{i=1}^{n}), \left(\{g^{r_{i,j}}\}_{i=1,j=i}^{\ell,\ell}\right), \left(\{g^{x_i x_j}\}_{i=1,j=\ell+1}^{n,n}\right)\right)$$

Where $x_i \in_R G_q$ and $r_{i,j} \in_R G_q$ for all $i, j \in 1, \ldots, n$. Let D be an adversary that attacks $\mathsf{Multi\text{-}DDH}$. In figure 7 we describe an adversary that chooses a random $\ell \in \{1, \ldots, n\}$ and, if D' breaks $\mathsf{Multi\text{-}DDH}$, it distinguish between hybrids $\boldsymbol{\chi}_\ell$ and $\boldsymbol{\chi}_{\ell+1}$.

Clearly, if $z = xy$ then $\boldsymbol{\gamma}_3 = \left(\{g^{\delta_i xy}\}_{i=1}^{\ell}\right)$ and thus $\boldsymbol{\chi} = \boldsymbol{\chi}_\ell$. Otherwise, if $z \in_R G_q$ then $\boldsymbol{\chi} = \boldsymbol{\chi}_{\ell+1}$. Then, the advantage of D' is given by $\frac{1}{n}\mathrm{Adv}_D^{\mathrm{MDDH}}(k)$. From the above, proving the indistinguishability between χ_1 and χ_n is straightforward.

A.3 Proof of Theorem 2 (sketch)

Proof. The proof proceeds through the indistinguishability of 4 games: $\mathsf{Game}_{\texttt{real}}$, $\mathsf{Game}_{\texttt{rand\_keys}}$, $\mathsf{Game}_{\texttt{rand}}$, and $\mathsf{Game}_{\texttt{ideal}}$.

Let $\mathcal{Z}$ be an environment and $\mathcal{A}$ an adversary. $\mathsf{Game}_{\texttt{real}}$ consist of an execution of SIGMIX with environment $\mathcal{Z}$ and adversary $\mathcal{A}$ in the real world. $\mathsf{Game}_{\texttt{rand\_keys}}$ is the same as $\mathsf{Game}_{\texttt{real}}$ except that instead of executing SIGMIX, the protocol $\mathsf{SIGMIX}_{\texttt{rand\_keys}}$ is executed. $\mathsf{SIGMIX}_{\texttt{rand\_keys}}$ is almost equal to SIGMIX, except that for each pair of honest parties P_i and P_j, $i \neq j$, the shared keys $k_{i,j}$ and $k_{j,i}$ are replaced with a random $r_{i,j} \in G_q$. $\mathsf{Game}_{\texttt{rand}}$ is the same as $\mathsf{Game}_{\texttt{rand\_keys}}$ except that, instead of $\mathsf{SIGMIX}_{\texttt{rand\_keys}}$, the protocol $\mathsf{SIGMIX}_{\texttt{rand}}$ is executed. $\mathsf{SIGMIX}_{\texttt{rand}}$ is almost equal to $\mathsf{SIGMIX}_{\texttt{rand\_keys}}$, except that each call to E with shared key $k_{i,j}$, where both P_i and P_j are

On input (g^x, g^y, g^z) the adversary D' does the following:

1. $\ell \xleftarrow{R} \{1, \ldots, n\}$.
2. Compute public keys for parties in $\{P_1, \ldots, P_\ell\}$:
   (a) $\delta_1 \leftarrow 1$.
   (b) $\delta_i \xleftarrow{R} G_q$ for $i = 2$ to ℓ.
   (c) $g^{x_i} \leftarrow (g^x)^{\delta_i}$ for $i = 1$ to ℓ.
3. Compute the public key of party $P_{\ell+1}$, that is $g^{x_{\ell+1}} \leftarrow g^y$.
4. Compute secret keys for parties not in $\{P_1, \ldots, P_\ell\}$, that is $x_i \xleftarrow{R} G_q$ for $i = \ell + 2$ to n.
5. Let γ_1 bet the set of all public keys, $\boldsymbol{\gamma}_1 \leftarrow (\{g^{x_i}\}_{i=1}^n)$.
6. Compute the shared keys for parties in $\{P_1, \ldots, P_\ell\}$:
   (a) $r_{i,j} \xleftarrow{R} G_q$ for $i = 1$ to ℓ and $j = i$ to ℓ.
   (b) $\boldsymbol{\gamma}_2 \leftarrow \left(\{g^{r_{i,j}}\}_{i=1,j=i}^{\ell,\ell}\right)$.
7. Compute the shared key between all parties in $\{P_1, \ldots, P_\ell\}$ and $P_{\ell+1}$:
   (a) $g^{r_{i,\ell+1}} \leftarrow (g^z)^{\delta_i}$ for $i = 1$ to ℓ.
   (b) $\boldsymbol{\gamma}_3 \leftarrow \left(\{g^{r_{i,\ell+1}}\}_{i=1}^{\ell}\right)$.
8. Compute the shared keys for which at least one exponent is known:
   (a) $g^{x_i x_j} \leftarrow (g^x)^{\delta_i x_j}$ for $i = 1$ to ℓ and $j = \ell + 2$ to n.
   (b) $\boldsymbol{\gamma}_4 \leftarrow \left(\{g^{x_i x_j}\}_{i=1,j=\ell+2}^{\ell,n}\right)$.
   (c) $\boldsymbol{\gamma}_5 \leftarrow \left(\{g^{x_i x_j}\}_{i=\ell+2,j=i}^{n,n}\right)$.
9. $\boldsymbol{\chi} \leftarrow (\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\gamma}_3, \boldsymbol{\gamma}_4, \boldsymbol{\gamma}_5)$.
10. Run $D(\boldsymbol{\chi})$ and output whatever it outputs.

Fig. 7. Adversary D' attacking DDH

honest, is replaced by a call to a completely random function F. $\mathsf{Game}_{\mathtt{ideal}}$ consist of an execution of $\mathcal{F}_{\mathsf{eaa}}$ with environment $\mathcal{Z}$ and the simulator $\mathcal{S}$ (defined in figure 8) in the ideal world. We let the output of each game be the output of the environment.

Is not hard to see that indistinguishability between $\mathsf{Game}_{\mathtt{real}}$ and $\mathsf{Game}_{\mathtt{rand\_keys}}$ follows directly from the hardness of Multi-DDH. Also, by proposition 1, there exist an adversary $\mathcal{D}_{\mathsf{DDH}}$ such that $n \cdot \mathrm{Adv}^{\mathrm{DDH}}_{\mathcal{D}_{\mathsf{DDH}}}(k) \geq \left|\Pr[\mathsf{Game}_{\mathtt{real}}] - \Pr[\mathsf{Game}_{\mathtt{rand\_keys}}]\right|$.

The indistinguishability between $\mathsf{Game}_{\mathtt{rand\_keys}}$ and $\mathsf{Game}_{\mathtt{rand}}$ is based on an hybrid argument. For each $i \in \{0, \ldots, \kappa\}$, we define the hybrid $\mathsf{Game}^i_{\mathtt{rand}}$ such that the l-th shared key, $l \leq i$, is replaced by a randomly chosen key. While the l-th shared key, $l > i$ is chosen as in $\mathsf{Game}_{\mathtt{rand\_keys}}$. By a standard hybrid argument we get that there exist a distinguisher $\mathcal{D}_{\mathsf{PRF}}$ such that $(\kappa + 1) \cdot \mathrm{Adv}^{\mathrm{PRF}}_{E,\mathcal{D}_{\mathsf{PRF}}}(k) \geq \left|\Pr\left[\mathsf{Game}_{\mathtt{rand\_keys}} = 1\right] - \Pr\left[\mathsf{Game}_{\mathtt{rand}} = 1\right]\right|$.

Let $I_{\mathcal{A}} \subseteq \{1, \ldots, n\}$ be the set indexes of parties corrupted by $\mathcal{A}$. The ideal adversary $\mathcal{S}^{\mathcal{A}}$ is described in figure 8, and it simulates the execution of $\mathsf{SIGMIX}_{\mathtt{rand}}$ only with access to $\mathcal{F}_{\mathsf{eaa}}$.

Since the values of messages sent by honest senders remains unknown to $\mathcal{S}^{\mathcal{A}}$ until all messages have been sent, when $\mathcal{F}_{\mathsf{eaa}}$ informs that a party $\tilde{P}_j$ (honest) sent some message, $\mathcal{S}^{\mathcal{A}}$ cheats the simulated $\mathcal{A}$ making $\mathcal{F}_{\mathsf{anon}}$ tells $\mathcal{A}$ that some message have been sent by $\tilde{P}_j$. When the set of messages M is revealed to $\mathcal{S}^{\mathcal{A}}$, it silently instruct the simulated parties to send the messages to $\mathcal{F}_{\mathsf{anon}}$. We will show that this seems indistinguishable from a real execution to $\mathcal{A}$.

Note that the view of $\mathcal{A}$ consist of the lexicographically ordered list L, the responses from $\mathcal{F}_{\mathsf{anon}}$, and the responses from $\bar{\mathcal{G}}_{\mathsf{krk}}$. Also note that the responses from $\mathcal{F}_{\mathsf{anon}}$ and the responses from $\bar{\mathcal{G}}_{\mathsf{krk}}$ are the same and in the same order in both $\mathsf{Game}_{\mathtt{ideal}}$ and $\mathsf{Game}_{\mathtt{rand}}$. Therefore, the only possible difference might be in L.

Let $L_{\mathtt{sim}}$ be the list in $\mathsf{Game}_{\mathtt{ideal}}$ and $L_{\mathtt{real}}$ be the list in $\mathsf{Game}_{\mathtt{rand}}$. We distinguish five type of elements in both lists , $(\mathtt{corrupt}, \mathtt{corrupt})$, $(\mathtt{corrupt}, \mathtt{honest})$, $(\mathtt{honest}, \mathtt{corrupt})$, $(\mathtt{honest}, \mathtt{honest})$, and `malformed`. Where $(t_1, t_2) \in \{\mathtt{corrupt}, \mathtt{honest}\}^2$, means that the sender is of type t_1 and the receiver is of type t_2, and `malformed` means that the element is not of the form $m||E_{k_{i,j}}(m)$, for some shared key between P_i and P_j, and $i \in I_{\mathcal{A}}$ or $j \in I_{\mathcal{A}}$. Note that `malformed` and $(\mathtt{corrupt}, t)$

Ideal adversary $\mathcal{S}$, running with parties $\tilde{P}_1, \ldots, \tilde{P}_n$ and executed in the $\mathcal{F}_{\mathsf{eaa}}, \bar{\mathcal{G}}_{\mathsf{krk}}$-hybrid model, proceeds as follows:

Initialization: Initially do the following:
1. Corrupt party $\tilde{P}_i$, for each $i \in I_{\mathcal{A}}$.
2. Start a simulated execution of $\mathsf{Game}_{\mathtt{rand}}$ with a dummy environment $\mathcal{Z}'$ and a dummy functionality $\mathcal{F}_{\mathsf{anon}}$.
3. Initialize multisets $M'_j \leftarrow \emptyset$, for each $j \in I_{\mathcal{A}}$, and $L \leftarrow \emptyset$.

Simulation of links $(\mathcal{Z}', \mathcal{A})$ and $(P_i, \bar{\mathcal{G}}_{\mathsf{krk}})$, $i \in I_{\mathcal{A}}$:
1. If m is received from $\mathcal{Z}$, then instruct $\mathcal{Z}'$ to send m to $\mathcal{A}$.
2. If m is sent from $\mathcal{A}$ to $\mathcal{Z}'$, then send m to $\mathcal{Z}$.
3. If P_j is instructed by $\mathcal{A}$ to send $(\mathtt{Register}, r)$ to $\bar{\mathcal{G}}_{\mathsf{krk}}$ and $\tilde{P}_j$ has not previously registered, instruct $\tilde{P}_j$ to send $(\mathtt{Register}, r)$ to $\bar{\mathcal{G}}_{\mathsf{krk}}$ and record the tuple $(x_i, y_i) \leftarrow (r, g^r)$.

Simulation of corrupt parties: When P_i, $i \in I_{\mathcal{A}}$, sends $m||\sigma$ to $\mathcal{F}_{\mathsf{anon}}$ do:
1. Let $L \leftarrow L \uplus \{m||\sigma\}$ and instruct $\mathcal{F}_{\mathsf{anon}}$ to send $(P_j, \mathtt{Sent})$ to $\mathcal{A}$.
2. If $\sigma = E_{y_j^{x_{i'}}}(m)$ for some registered public key y_j, $j \in \{1, \ldots, n\}$, and some registered secret key $x_{i'}$, $i' \in I_{\mathcal{A}}$, then send $(\mathtt{Send}, m, i', j)$ to $\tilde{P}_{i'}$. If $j \in I_{\mathcal{A}}$, then $M'_j \leftarrow M'_j \uplus \{(m, i')\}$.

Simulation of honest parties: If $(\tilde{P}_i, \mathtt{Sent})$, $i \notin I_{\mathcal{A}}$, is received from $\mathcal{F}_{\mathsf{eaa}}$, then instruct $\mathcal{F}_{\mathsf{anon}}$ to send $(P_i, \mathtt{Sent})$ to $\mathcal{A}$.

End of simulation: Once that $(\mathtt{Messages}, M)$ is received from $\mathcal{F}_{\mathsf{eaa}}$ and for each $j \in I_{\mathcal{A}}$ $(\mathtt{Messages}, M_j)$ is sent to $\tilde{P}_j$, do:
1. Let $M' \leftarrow M \setminus \left(\biguplus_{j \in I_{\mathcal{A}}} \biguplus_{(m,l) \in M_j} \{m\}\right)$.
2. For each $m \in M'$, let $\sigma \xleftarrow{R} \{0,1\}^t$ and $L \leftarrow L \uplus \{m||\sigma\}$.
3. For each $(m, i) \in M_j \setminus M'_j$, let $k_{i,j} \leftarrow y_i^{x_j}$ and $L \leftarrow L \uplus \{m||E_{k_{i,j}}(m)\}$.
4. Sort L lexicographically and proceed with $\mathcal{F}_{\mathsf{anon}}$ as if L were the final list of received messages.

Fig. 8. The ideal adversary $\mathcal{S}$

are both in $L_{\mathtt{sim}}$ and $L_{\mathtt{real}}$, given that such elements only comes from corrupted parties. Therefore, they must be added to $L_{\mathtt{sim}}$ in line 1 of the "Simulation of corrupt parties" stage. A message of the type $(\mathtt{honest}, \mathtt{corrupt})$ is added to $L_{\mathtt{real}}$ if an only if the corresponding message appears on $M_j \setminus M'_j$, for some $j \in I_{\mathcal{A}}$. Therefore, it must be added to $L_{\mathtt{sim}}$ in line 3 of the "End of simulation" stage. Note that, given that the sender is honest, the message is exactly the same message added in $L_{\mathtt{real}}$. Finally a message of the type $(\mathtt{honest}, \mathtt{honest})$ is added to $L_{\mathtt{real}}$ if and only if the correspondent message can be found in M'. Therefore, it must be added to $L_{\mathtt{sim}}$ in line 2 of the "End of simulation" stage. Note that, given that both the sender and the receiver are honest, the subset of messages of type $(\mathtt{honest}, \mathtt{honest})$ in $L_{\mathtt{real}}$ must be of the form $\{m_1||F(m_1), \ldots, m_s||F(m_s)\}$, where $s \in \mathbb{N}$ and F is a completely random function. The assumption of environments that do not send the same message twice implies that, if $i \neq j$, then $m_i \neq m_j$ and thus $F(m_i)$ and $F(m_j)$ are independent randomly chosen strings. Therefore, the messages added to $L_{\mathtt{real}}$ in line 2 follows exactly the same distribution of correspondent messages in $L_{\mathtt{sim}}$. We conclude that the view of $\mathcal{A}$ in $\mathsf{Game}_{\mathtt{rand}}$ is exactly the same as in $\mathsf{Game}_{\mathtt{ideal}}$.

The only possible difference on the view of $\mathcal{Z}$ is due to a difference in the output of an honest party. Specifically, this difference is possible because, in the "Simulation of corrupted parties" stage, some messages $m||\sigma$ could not have been sent to $\mathcal{F}_{\mathsf{eaa}}$ in line 2 of the "Simulation of corrupt parties" stage. This set of dropped messages may contain forgery, i.e. messages created by $\mathcal{A}$ that are accepted by an honest P_j as coming from an honest P_i.

Therefore, if $\mathcal{A}$ does not forges, then the the output of $\mathcal{Z}$ in $\mathsf{Game}_{\mathtt{rand}}$ must be the same in $\mathsf{Game}_{\mathtt{ideal}}$. Let "$\mathcal{A}$ forges" be the event where an honest party accepts a message sent by $\mathcal{A}$ as coming from another honest party. Then $\Pr\left[\mathsf{Game}_{\mathtt{rand}} = 1 | \overline{\mathcal{A} \text{ forges}}\right] = \Pr\left[\mathsf{Game}_{\mathtt{ideal}} = 1 | \overline{\mathcal{A} \text{ forges}}\right]$. By the *fundamental lemma of game playing* we get that $|\Pr[\mathsf{Game}_{\mathtt{rand}} = 1] - \Pr[\mathsf{Game}_{\mathtt{ideal}} = 1]| \leq \Pr[\mathcal{A} \text{ forges}]$.

In the event "$\mathcal{A}$ forges" there is at least one $m||\sigma$ such an honest P_j accept m as coming from an honest P_i, which means that $\sigma = F(m)$. By the assumption of adversaries that do not play replay attacks, $m||\sigma$ should be different of any $m'||\sigma'$ previously seen by $\mathcal{A}$. Moreover, it must be that

$m \neq m'$ for all previously $m'||\sigma'$ seen by $\mathcal{A}$, otherwise it must be that $\sigma \neq \sigma' = F(m') = F(m)$, because $m'||\sigma'$ was honestly sent, and thus $m||\sigma$ can not be accepted by an honest party.

Given that $m \neq m'$, for all previously $m||\sigma$ seen by $\mathcal{A}$, it must be that $\mathcal{A}$ have never seen $F(m)$. Therefore, $F(m)$ is randomly chosen independently from σ and thus $\Pr\left[F(m) = \sigma\right] = 1/2^t$. Given that $\mathcal{A}$ can send a most κ messages, it must hold that $\Pr\left[\mathcal{A} \text{ forges}\right] \leq \kappa \cdot 2^{-t}$.

Combining the results, we get that SIGMIX GUC-emulates $\mathcal{F}_{\mathsf{eaa}}$.