Deniability for protocols with shared state with applications to Anonymous Authentication

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

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

ABSTRACT

In this work we propose a different approach to solve the problem of *anonymous authentication* (Boneh et al. CCS 1999). Instead of the classical solution that uses *ring signatures*, we directly combine an an *anonymous channel* with *deniable authentication* without contradicting anonymity. The classical solution provides anonymity with respect to a set of authorized users, our solution provides anonymity with respect to authorized and no authorized (but maybe authorized for other group) users at the price of a different form of anonimity.

We develop new tools to prove deniability of protocols in the *Generalized Universal Composability* framework (GUC, Canetti et al. TCC 2007). In particular we define deniability for multi party protocols, and show that deniable protocols are exactly those protocols that realize the ideal functionality $\mathcal{F}_{\mathsf{den}}$. This result enable us to prove deniability of many UC and GUC functionalities.

We use our new tools to design a functionality for *Anonymous Authenticated Channels* and design a protocol that realizes it in the GUC framework.

General Terms

Theory

1. INTRODUCTION

Consider an online forum where users can post sensible data on a server and may be concerned about being linked to their posts. Additionally the server requieres to give different relevance to posts by sorting the list of posts using the users reputation, measured by some non-integer number. Arguably, we would like to implement an *Anonymous Authentication* protocol [1], where users can authenticate to a server maintaining their anonymity.

The typical solution to the anonymous authentication problem is such that "the authentication protocol carried out between the user and the server does not identify the user" [1]. Instead it verifies that the user belongs to some *authorized group*, in our case the group of users with the same reputation. Note that the user will be anonymous only with respect to other users with the same reputation.

Lindell [19] defined two security requirements for an anonymous authentication protocol:

1. Secure authentication: No unauthorized user should be able to fool the server into granting it access (except with very small probability).
2. Anonymity: The server should not know which user it is interacting with.

But what if the reputation is sufficiently "fine grained"? We can even assume that each user have a different reputation and thus the authorized group is of size 1. In such a case, it is unavoidable that the server will learn the user's identity. Can we still provide some some meaningful form of anonymity to the user? Specifically, can we guarantee that the user will be anonymous with respect to other parties, not to the server but other parties than the server (eg. other users, other servers, or any external entity)?

Note that in Lindell's definition, with respect to anonymity the "enemy" of the user is the server. Indeed, when the server is malicious it will be trying to learn which user it is authenticating. We assume that the user's "enemy" is not the server but any other party. In our setting the user is not worried about what will the server learn, but is worried about what an eavesdropper or user will learn (maybe based on information received from a malicious server).

We allow users to authenticate themselves to the server (we violate the anonymity requirement). But we assure that the only entities who are aware of the identities executing the protocol are only the user and the server. Any other entity (including an adversary monitoring the communication) should not be aware of the identities participating in the protocol.

We construct our solution by combining an anonymous channel and deniable authentication. Deniable authentication allows the server to be convinced of the authenticity of a received message, but it avoids to convince any other party of the authenticity of a received message. The anonymous channel hide the identities to any party other than the prover and the server.

We prove the security of our protocol in the *Generalized Universal Composability* framework [8] because it allows to capture *deniability* [12] and also our protocol inherits strong security guarantees such as *composability*.

1.1 Related Work

Deniable authentication was first defined by Dwork et al. in their seminal work on *Concurrent Zero-Knowledge*. Later Dodis et al introduced the notion of online deniability restricted to the cases of authentication, identification, and Key Exchange [12], and Zero Knowledge [14]. They showed that for each of these tasks deniability is equivalent to GUC-realize the corresponding ideal functionalities. As a consequence their definition implies security under general concurrent composition.

Many protocols for deniable authentication and identification (e.g. [15, 16, 23, 22, 20, 12]), deniable key exchange (e.g. [24, 27]), and deniable zero knowledge [21, 14] have been proposed. All of them fall in one of the two categories: (offline) deniable and online-deniable.

Most of the proposed protocols proposed before [12] are offline deniable (with the sole exception of HMQV [23] as noted by Dodis et al. [12]).

Dodis et al. presented an online-deniable authentication with respect to static adversaries [12], which is the underlying authentication procedure that our protocol for anonymous authentication uses. They also showed the impossibility of online-deniable authentication with respect to adaptive adversaries, and this result is the reason for restricting our protocol to static adversaries.

Canetti and Vald [9, 10] introduced the notion of *Bi-Deniability* with many similarities to the definition of deniability of Dodis et al. [12] and with our work. They showed that a protocol is Bi-Deniable if and only if it LUC-realizes $\mathcal{F}_{\overline{\mathsf{auth}}}$, the LUC authentication functionality. We note that $\mathcal{F}_{\overline{\mathsf{auth}}}$ requires *correctness*, if the sender sends m the receiver receives m, but in deniability this is not the case. We discuss about their work in Sect. 3.2.

Another meaning for deniability is that of *Deniable encryption* as defined in [3], where a Deniable encryption scheme should allow the receiver (or the sender) to deny that the content of a cyphertext is a given plaintext. In contrast, we would say that an encryption scheme $\mathcal{E}$ is deniable when a party executing $\mathcal{E}$ can deny that the execution is taking place.

The modern study of anonymous channels was started in [11] with mix-nets. More recently, in [26] Wikström introduced a UC-secure mixnets with respect to static adversaries. His construction relies on a honest majority of *mixers* and a distributed Distributed Key Generation protocol, which relies on a Common Refference String.

Ishai et al. introduced an anonymity functionality Anon which provides *sender anonymity* [17]. That is, the adversary only learns the multiset of all received messages for each receiver. A strengthened version of Anon is one where the adversary only learns the multiset of all sent messages, and thus this strengthened version of Anon achieves *sender and receiver anonymity*. Our ideal functionality for anonymous authenticated channels $\mathcal{F}_{\mathsf{aac}}$ can be viewed as the "conjunction" of the strengthened version of Anon and the authenticated channels functionality $\mathcal{F}_{\mathsf{auth}}$ from [5].

The main goal of Ishai et al. was to construct secure channels from anonymous channels. They observed that two parties can choose random values r_1 and r_2 and communicate each other those values using the anonymous channel. Each party can compute the secret bit $r_1 > r_2$, but any other party is unaware of this bit. They pointed out that there is a drawback in this approach, an adversary can take advantage of the anonymity and send another random value r^* such that parties may finally compute $r_1 > r^*$ or $r^* > r_2$. A simple solution to this is to use an authentication mechanism, but they pointed out that this approach fails because most authentication tasks are *non-repudiable* and therefore contradict the privacy of the secret bit. But this is not true for our functionality $\mathcal{F}_{\mathsf{aac}}$, therefore it can be used to implement secure channels using the "flawed" approach discarded by Ishai et al.

Most work on the anonymous authentication literature relies on *Ring Signature* protocols in order to make the user identify anonymously as member of an authorized group [18, 1, 13]. As we mentioned earlier, ring signatures don't provide anonymity guarantees when the size of the ring is 1.

Naor proposed the notion of *Deniable Ring Signatures* which combines deniability and ring signatures [20]. As noted by Dodis et al. their protocols can not be online deniable as long as verifiers do not register public keys [12]. We also note that in our setting deniable ring signature do not necessarily provide anonymity between users of different rings (in our example users with different reputations).

1.2 Our Contribution

We generalize the *online deniable authentication* definition from [12] in the form of *online deniability*. We show that a protocol is online deniable if and only if it GUC realizes the ideal functionality $\mathcal{F}_{\mathsf{den}}$. We note that this result also applies to Bi-deniability and the LUC framework [10, 9]. We correct a mistake from [10] and show that a protocol is Bi-deniable if and only if the protocol LUC realizes $\mathcal{F}_{\overline{\mathsf{den}}}$, the LUC equivalent for $\mathcal{F}_{\mathsf{den}}$.

We also show that most ideal functionalities are deniable by noting that most functionalities are subroutine respecting (for example $\mathcal{F}_{\mathsf{auth}}$, $\mathcal{F}_{\mathsf{zk}}$, and $\mathcal{F}_{\mathsf{com}}$), and showing that a subroutine respecting protocol GUC-emulates $\mathcal{F}_{\mathsf{den}}$. Our result implies the all previous results on deniability we are aware of: deniability for GUC-secure Zero Knowledge protocols [14, Thm.8], deniability for GUC-secure authentication [12, Prop.1], and deniability of GUC-secure key exchange and identification from [12].

We apply our results on deniability to construct a seemingly paradoxical functionality: one that combines anonymous channels and authenticated channels. That is, if a party receives a message, she is convinced of the integrity of the message and the identity of the sender, but no other parties that the sender or the receiver are not aware of that.

To prove the security of the protocol, we use the fact that the Multi Decisional Diffie-Hellman assumption is a consequence of the Decisional Diffie-Hellman assumption [2]. We give a better bound on the adversarial advantage than the one given in [2].

1.3 Organization

We introduce the UC and GUC frameworks in section 2. Next we define deniability an show our result that characterizes deniable protocols on Sect. 3.

In section 4 we recall the mixnet of Wikström. In section 5 we define our ideal functionality for anonymous autheticated channel $\mathcal{F}_{\mathsf{aac}}$ and we desgin a protocol that GUC realizes $\mathcal{F}_{\mathsf{aac}}$.

2. CRYPTOGRAPHIC MODEL

We define and prove the security guarantees of our protocol in the Generalized Universal Composability (GUC) framework. The GUC framework [8] is a generalization of the Universal composability framework [6]. Both frameworks models concurrent protocols, but the GUC framework also models concurrent protocols sharing state between them.
We first review the UC because GUC builds on UC with some minor, but significant, modifications.

2.1 The Universal Composability framework (UC)

The UC framework provides a methodology for a modularized designing and modularized proving the security of distributed multiparty cryptographic protocols, even when composed with arbitrary other protocols. The approach of UC is, given a real world protocol, abstract the security one want the protocol have into a idealized secure protocol. Then, it must be shown that executing the real or the idealized protocol is essentially the same, and thus the real protocol is as secure as the idealized protocol.
A multiparty real world protocol π is distributed between parties $P_1, \ldots, P_n$. The protocol π can be monitored and some participants can have "undesired" behavior, attempting with the desired security of the protocol. All the possible undesired behavior is executed by one machine, the real adversary $\mathcal{A}$. In an execution of π the adversary can eavesdrop and manipulate all the data sent from one party to another, and can corrupt some parties and then execute arbitrary code on them.
On the other hand, given a functionality $\mathcal{F}$, we consider the ideal protocol denoted by $\mathsf{IDEAL}_{\mathcal{F}}$. $\mathcal{F}$ is known as an *ideal functionality* and in an execution of $\mathsf{IDEAL}_{\mathcal{F}}$ is executed by a *trusted party*. $\mathcal{F}$ does all the computation of $\mathsf{IDEAL}_{\mathcal{F}}$ because each party P_i $i \in \{1, \ldots, n\}$ only passes his input to the ideal functionality and returns unchanged all it receives from $\mathcal{F}$. We say that P_i executes the *dummy party protocol* and and that P_i is a *dummy party*, denoted by $\tilde{P}_i$. In an execution of the protocol $\mathsf{IDEAL}_{\mathcal{F}}$ an ideal adversary $\mathcal{S}$ cannot eavesdrop the data sent between a party P_i and $\mathcal{F}$, because is sent by a secure direct link between P_i and $\mathcal{F}$. $\mathcal{S}$ is allowed to stop or drop exchanged data between some party and $\mathcal{F}$, and is allowed to communicate with $\mathcal{F}$ (indicating some influence or leak to/from $\mathcal{F}$).
Concurrent execution with arbitrary other protocols is given by interaction between a protocol (real or ideal) and a special machine $\mathcal{Z}$ known as the environment. It is considered that all the externals protocols are executed by the environment $\mathcal{Z}$ and the inputs received by each party are given by the environment. The environment also haves direct communication with the adversary ($\mathcal{A}$ or $\mathcal{S}$). When the protocol finish, the output of each party is received by $\mathcal{Z}$ and then $\mathcal{Z}$ outputs a bit.

Definition 1. The distribution that follows the output of an environment $\mathcal{Z}$, executed with adversary $\mathcal{A}$, protocol π and security parameter k, is denoted by $\mathsf{EXEC}_{\mathcal{Z},\mathcal{A},\pi}(k)$.

To execute ideal protocols we must consider *hybrid models*. A protocol executed in the $\mathcal{F}_1, \ldots, \mathcal{F}_m$-hybrid model is a protocol where each party have access to ideal functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_m$. We denote the output of $\mathcal{Z}$ executed with $\mathcal{A}$, protocol π in the $\mathcal{F}_1, \ldots, \mathcal{F}_m$-hybrid model and security parameter k by $\mathsf{EXEC}^{\mathcal{F}_1,\ldots,\mathcal{F}_m}_{\mathcal{Z},\mathcal{A},\pi}(k)$. In the ideal world, the output of $\mathcal{Z}$ executed with $\mathcal{S}$ and the protocol $\mathsf{IDEAL}_{\mathcal{F}}$ is equal to $\mathsf{EXEC}^{\mathcal{F}}_{\mathcal{Z},\mathcal{S},\mathsf{IDEAL}_{\mathcal{F}}}$.

Definition 2. We say that a protocol π UC-emulates ρ if for all environment $\mathcal{Z}$ and all adversary $\mathcal{A}$ there exist an ideal adversary $\mathcal{S}$ such that

$$\mathsf{EXEC}_{\mathcal{Z},\mathcal{A},\pi} \approx \mathsf{EXEC}_{\mathcal{Z},\mathcal{S},\rho}.$$

When ρ is the protocol $\mathsf{IDEAL}_{\mathcal{F}}$ executed in the $\mathcal{F}$-hybrid model we say that π UC-realizes $\mathcal{F}$, and thus π is UC-secure.
The main advantage of UC security is that it composes, that is, the security of a protocol is maintained even when the protocol is being executed with other, possibly adversarially chosen, protocols. But there is a restriction, the protocol must be *subroutine respecting*. A subroutine respecting protocol 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 can't share state with other protocols.
Composition of UC security is guaranteed by the *composition theorem* (Theorem 1).

THEOREM 1. *[6] Suppose that subroutine respecting protocol π UC-emulates ρ (also is subroutine respecting). Let ϕ be a protocol that (possibly) makes calls to π and let $\phi^{\pi/\rho}$ be the same protocol ϕ except that all calls to π are replaced by calls to ρ. Then the protocol $\phi^{\pi/\rho}$ UC-emulates π.*

2.2 The Generalized UC framework (GUC)

As we said before, a UC-secure protocol must be *subroutine respecting*. If this restriction is not accomplished the the UC-theorem could not hold.
One scenario where is not realistic to assume that protocols are subroutine respecting is when setup assumptions are needed in order to UC-emulate the ideal functionality. For example, in the case of zero knowledge is known that is impossible to realize the zero knowledge ideal functionality without some setup assumption, like the common reference string (CRS), when only the minority of the parties is honest [4].
In UC the CRS is modeled as an ideal functionality $\mathcal{F}_{\mathsf{crs}}$ that publicizes a random string to all parties, and zero knowledge protocols are proved to UC-realize the ideal zero knowledge functionality in the $\mathcal{F}_{\mathsf{crs}}$-hybrid model.
Such ideal functionality (a trusted party) is obvious hard to obtain in the real world, and consequently is unrealistic to consider a local copy of the trusted setup for each zero knowledge protocol session. In fact it must be considered shared between the many zero knowledge protocols and possibly other protocols.
In the case of a shared $\mathcal{F}_{\mathsf{crs}}$ used for realizing a ZK protocol is shown in [21] that it leads to loose of the natural deniability of a ZK protocol. Furthermore in [28] is shown that it also could lead to the loose of general composability and the "Proof of knowledge" property.
In GUC protocols can be not subroutine respecting but $\bar{\mathcal{G}}$-*subroutine respecting*, and it means that the protocol is subroutine respecting except that is allowed to call the *shared*

functionality $\bar{\mathcal{G}}$.
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 makes able to the environment to simulate protocols sharing state with the analyzed protocol.
The definitions of execution and emulation in GUC are almost identical to the definitions of UC, but the names are distinct. From now on with EXEC we refer to both, UC-execution and GUC-execution, and with UC-emulation we refer also to GUC-emulation. Similarly as in UC is possible to demonstrate a composition theorem for $\bar{\mathcal{G}}$-subroutine respecting protocols.

3. DENIABILITY IN THE GUC FRAMEWORK

Deniable authentication was first defined by Dwork, Nahor and Sahai in [15]. Several modifications and generalizations have been made since then, and here we consider the definition given by Dodis, Katz, Smith and Walfish in [12], since their definition applies to a concurrent and distributed setting.
Roughly speaking an authentication protocol is deniable if nobody can prove that a particular session of the protocol is taking place or have ever took place. In [12] is shown that such property can be achieved considering an on line judge who must decide who is he talking to: an informant who is observing a real session of the authentication protocol, or a misinformant without access to the real session of the protocol but still try to convince the judge that the session is taking place. The protocol is said to be an online deniable authentication protocol if for all judge and all informants there exist a mis informant such that the judge can't distinguish from the informant and the misinformant with overwhelming probability.
In the full version of [12] is demonstrated that this notion is equivalent to GUC-realize the ideal functionality $\mathcal{F}_{\mathsf{auth}}$. They pointed out that a protocol GUC-realizing a functionality $\mathcal{F}$ is as deniable as $\mathcal{F}$. Furthermore, the ideal functionality $\mathcal{F}_{\mathsf{auth}}$ is "fully simulatable", meaning that the protocol can be completely simulated without the participation of any party. The conclusion is that the functionality $\mathcal{F}_{\mathsf{auth}}$ is deniable too, because the misinformant can simulate $\mathcal{F}_{\mathsf{auth}}$.

3.1 Definition

We refer to "online deniability" instead of "online authentication deniability", and the definition remains essentially in the same fashion of [12], but for completeness we include it. consider a judge $\mathcal{J}$, an informant $\mathcal{I}$, misinformant $\mathcal{M}$ and a global setup functionality $\bar{\mathcal{G}}$. Instead of just a sender and a receiver running an authentication protocol we consider parties $P_1, \ldots, P_n$ running a distributed protocol π.
In the *real world* the judge $\mathcal{J}$ gives the input of each party participating in π to the informant $\mathcal{I}$. The informant gives the inputs to the parties and witness the execution of the protocol π. More precisely, $\mathcal{I}$ is equal to the UC real world adversary but it also can give input to the parties (in UC this is done by the environment). The parties can communicate with a shared functionality $\bar{\mathcal{G}}$ (if for example the parties need to register in a PKI), and $\mathcal{J}$ and $\mathcal{I}$ have also access to $\bar{\mathcal{G}}$. We denote by $\mathsf{RealDen}^{\bar{\mathcal{G}}}_{\mathcal{J},\mathcal{I},\pi}(k)$ the output of a judge $\mathcal{J}$ executed in the real world with informant $\mathcal{I}$, shared functionality $\bar{\mathcal{G}}$, and protocol π executed with security parameter k.
On the other hand is the *simulated world*, there the misinformant $\mathcal{M}$ try to mimic the behavior of a real execution of π, but with no access to parties $P_1, \ldots, P_n$. We denote by $\mathsf{SimDen}^{\bar{\mathcal{G}}}_{\mathcal{J},\mathcal{M}}(k)$ the output of a judge $\mathcal{J}$ executed in the ideal world with misinformant $\mathcal{M}$, setup functionality $\bar{\mathcal{G}}$, and with security parameter k.

Definition 3. We say that a protocol π is online deniable if for all judge $\mathcal{J}$ and all informant $\mathcal{I}$ there exists a misinformant $\mathcal{M}$ such that

$$\mathsf{RealDen}^{\bar{\mathcal{G}}}_{\mathcal{J},\mathcal{I},\pi} \approx \mathsf{SimDen}^{\bar{\mathcal{G}}}_{\mathcal{J},\mathcal{M}}.$$

Our goal is to express deniability in an equivalent ideal functionality, that is, a protocol π is online deniable if and only if it UC-realizes some ideal functionality $\mathcal{F}_{\mathsf{den}}$.
Before proving that, we note that the deniability real world is just a syntactic transformation of the UC real world experiment. This because any environment and adversary can be simulated with a judge and an informant, and vice versa. The judge only forwards the input of each party, given by a simulated environment, to the informant/misinformant, and the informant gives input to parties and simulates the adversary. On the other hand, each judge and informant can be simulated by an environment and adversary. The environment just sends the input of parties given by the judge and the adversary simulates the informant.

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

$$\mathsf{EXEC}^{\bar{\mathcal{G}}}_{\mathcal{Z},\mathcal{A},\pi} \equiv \mathsf{RealDen}^{\bar{\mathcal{G}}}_{\mathcal{J}^{\mathcal{Z}},\mathcal{I}^{\mathcal{Z}},\pi}$$

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 $\mathcal{J}$ giving his "input information" to the corresponding party. All other information is given to $\mathcal{A}^{\mathcal{I}}$. As the definition of the informant is similar to the the definition of the real adversary, $\mathcal{A}^{\mathcal{I}}$ can simulate perfectly $\mathcal{I}$. 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 start an execution of π following the instructions of $\mathcal{J}^{\mathcal{Z}}$. The simulation of $\mathcal{Z}$ and $\mathcal{A}$ is perfect, and thus the output of $\mathcal{J}^{\mathcal{Z}}$ follows the same distribution that follows the output of $\mathcal{Z}$. □

The functionality $\mathcal{F}_{\mathsf{den}}$, defined in figure 1, gives all the necessary to run an misinformant, and thus a simulator can

simulate the misinformant.
We show an equivalent result to the one in [12].

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 x_i to the adversary $\mathcal{S}$.
2. When $(y_i, \tilde{P}_i)$ is received from the adversary $\mathcal{S}$, send y_i to $\tilde{P}_i$.

Figure 1: The ideal functionality $\mathcal{F}_{\mathsf{den}}$

THEOREM 2. *A $\bar{\mathcal{G}}$-subroutine respecting protocol π is online deniable if and only if it UC-realizes the ideal functionality $\mathcal{F}_{\mathsf{den}}$ in the $\bar{\mathcal{G}}$-hybrid model.*

PROOF. We first prove the first implication, that is: π is online deniable implies π UC-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

$$\left|\Pr\left[\mathsf{RealDen}^{\bar{\mathcal{G}}}_{\mathcal{J}^{\mathcal{Z}},\mathcal{I}^{\mathcal{A}},\pi} = 1\right] - \Pr\left[\mathsf{SimDen}^{\bar{\mathcal{G}}}_{\mathcal{J}^{\mathcal{Z}},\mathcal{M}} = 1\right]\right| = \epsilon.$$

and ϵ is negligible in the security parameter.
Consider the ideal adversary $\mathcal{S}^{\mathcal{M}}$, with oracle access $\mathcal{M}$, defined in figure 2. We note that in both, $\mathsf{SimDen}^{\bar{\mathcal{G}}}_{\mathcal{J}^{\mathcal{Z}},\mathcal{M}}$ and

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 $(x_i, \tilde{P}_i)$ is received from $\mathcal{F}_{\mathsf{den}}$, send $(x_i, \tilde{P}_i)$ to $\mathcal{M}$.
3. When x is received from $\mathcal{Z}$, send x to $\mathcal{M}$.
4. When x is received from $\mathcal{M}$, send x to $\mathcal{Z}$.
5. When $\mathcal{M}$ writes $(\texttt{Output}, y_i, \tilde{P}_i)$, write $(y_i, \tilde{P}_i)$ to $\mathcal{F}_{\mathsf{den}}$.

Figure 2: The ideal adversary $\mathcal{S}^{\mathcal{M}}$

$\mathsf{EXEC}^{\bar{\mathcal{G}},\mathcal{F}_{\mathsf{den}}}_{\mathcal{Z},\mathcal{S}^{\mathcal{M}}}$, $\mathcal{Z}$ is wired with $\mathcal{M}$ as if $\mathcal{M}$ was an adversary. The only difference is that in $\mathsf{SimDen}^{\bar{\mathcal{G}}}_{\mathcal{J}^{\mathcal{Z}},\mathcal{M}}$ the input to the parties is forwarded from $\mathcal{Z}$ by $\mathcal{M}$. This change must be overlooked by $\mathcal{Z}$ and thus

$$\mathsf{SimDen}^{\bar{\mathcal{G}}}_{\mathcal{J}^{\mathcal{Z}},\mathcal{M}} \equiv \mathsf{EXEC}^{\bar{\mathcal{G}},\mathcal{F}_{\mathsf{den}}}_{\mathcal{Z},\mathcal{S}^{\mathcal{M}}}.$$

Then

$$\left|\Pr\left[\mathsf{EXEC}^{\bar{\mathcal{G}}}_{\mathcal{Z},\mathcal{A},\pi} = 1\right] - \Pr\left[\mathsf{EXEC}^{\bar{\mathcal{G}},\mathcal{F}_{\mathsf{den}}}_{\mathcal{Z},\mathcal{S}^{\mathcal{M}}} = 1\right]\right| = \epsilon.$$

Therefore, π UC-emulates $\mathcal{F}_{\mathsf{den}}$.
The other direction is similar. Suppose that π UC-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 as in the first implication, a misinformant $\mathcal{M}^{\mathcal{S}}$ can simulate $\mathcal{S}$ for $\mathcal{J}$. This seems indistinguishable from $\mathcal{I}$ to $\mathcal{J}$, and therefore π is online deniable. □

Deniability is a concern not important per se, it becomes important when is an additional concern to achieve some task, say UC-realize some functionality $\mathcal{F}$. As an empirical fact, most functionalities tends to be deniable ($\mathcal{F}_{\mathsf{auth}}$ and $\mathcal{F}_{\mathsf{zk}}$ are examples). Indeed, this is a consequence of the fact that most ideal functionalities are subroutine respecting.

LEMMA 2. *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 simulated only with access to the inputs given by $\mathcal{F}_{\mathsf{den}}$.
For ρ the adversary must also use his access to the public interface of $\bar{\mathcal{G}}$. □

By the transitivity of the UC-emulation relation, when we prove that a protocol UC-realizes some deniable ideal functionality we are also proving that the protocol is deniable.

3.2 Bi-Deniability and UC with Local adversaries (LUC)

Canetti and Vald [9, 10] introduced the notion of *Bi-Deniability* with many similarities to the definition of deniability of Dodis et al. [12] and our work.
In the two party case, the notion of Bi-deniability is exactly as deniability with the difference that instead of just one informant and misinformant, two informants and two misinformants are considered. Each pair of informant/misinformant is assigned to one of the two parties participating in the protocol, and can provide input to the party, corrupt the party, and give output to the judge.
A protocol said to be Bi-deniable if for for any pair of informants there exists a pair of misinformants, such that for any judge the execution of the protocol witnessed by the pair of informants is indistinguishable from the execution of the misinformants.
Canetti and Vald showed that a two party protocol is Bi-deniable if and only if it LUC-realizes $\mathcal{F}_{\overline{\mathsf{auth}}}$ [9, Thm. 7], where $\mathcal{F}_{\overline{\mathsf{auth}}}$ is the LUC functionality for authenticated communication and is described in fig. 3.
Next, we will describe a simple protocol that is Bi-deniable but does not LUC-realize $\mathcal{F}_{\overline{\mathsf{auth}}}$. Consider the protocol $\pi = (S, R)$ executed in the $\mathcal{F}_{\overline{\mathsf{auth}}}$-hybrid model, where S runs the code of the dummy party and R returns the message 0 whenever it receives some message m from $\mathcal{F}_{\overline{\mathsf{auth}}}$.

The ideal functionality $\mathcal{F}_{\overline{\mathsf{auth}}}$ running with parties S, R, and adversaries $\mathcal{S}_{(S,R)}, \mathcal{S}_{(R,S)}$ proceeds as follows:

1. Upon receiving an input $(\mathtt{Send}, sid, m)$ from party S, do: If $sid = (S, R, sid')$ for some R, then record m and output $(\mathtt{Send}, sid, m)$ to $\mathcal{S}_{(S,R)}$.
2. Upon receiving "approve" from $\mathcal{S}_{(S,R)}$, if m is recorded provide $(\mathtt{Send}, sid, m)$ to $\mathcal{S}_{(R,S)}$, and after $\mathcal{S}_{(R,S)}$ approves, output $(\mathtt{Send}, sid, m)$ to R and halt. Otherwise, provide $(\mathtt{Send}, sid, \perp)$ to $\mathcal{S}_{(R,S)}$ and halt. (Both adversaries control the channel delay.)
3. Upon receiving $(\mathtt{Corruptsend}, sid, m')$ from $\mathcal{S}_{(S,R)}$, if S is corrupt and m was not yet delivered to $\mathcal{S}_{(R,S)}$, then output $(\mathtt{Send}, sid, m')$ to $\mathcal{S}_{(R,S)}$, and after $\mathcal{S}_{(R,S)}$ approves, output $(\mathtt{Send}, sid, m')$ to R and halt.

Figure 3: The ideal functionality $\mathcal{F}_{\overline{\mathsf{auth}}}$

Note that π is indeed Bi-deniable. The misinformant $\mathcal{M}_S$ forwards the input m given by $\mathcal{J}$ and send the $(\mathtt{signal})$ message to $\mathcal{M}_R$ after $\mathcal{J}$ instruct to send "approve" to $\mathcal{F}_{\overline{\mathsf{auth}}}$. The misinformant $\mathcal{M}_R$ waits for the $(\mathtt{signal})$ message from $\mathcal{M}_S$ and the instruction "approve" from $\mathcal{J}$, and outputs the message 0 to $\mathcal{J}$. Clearly, the view of a judge executed in the Informant-Experiment is exactly the same as when executed in the Misinformant-Experiment.
The protocol π can not LUC-realize $\mathcal{F}_{\overline{\mathsf{auth}}}$, because it violates the *correctness* property that is inherent to any protocol realizing an authentication functionality. This also shows that the equivalence between Bi-deniability and LUC realization of $\mathcal{F}_{\overline{\mathsf{auth}}}$ is not correct, indeed the only implication that holds is that if a protocol LUC-realizes $\mathcal{F}_{\overline{\mathsf{auth}}}$ then the protocol is Bi-deniable.
Nevertheless, we can show that the equivalence holds when the correctness property is removed from $\mathcal{F}_{\overline{\mathsf{auth}}}$. In that case we obtain our deniability function adapted to LUC, $\mathcal{F}_{\overline{\mathsf{den}}}$, which is described in figure 4.

The ideal functionality $\mathcal{F}_{\overline{\mathsf{den}}}$ running with parties S, R, and adversaries $\mathcal{S}_{(S,R)}, \mathcal{S}_{(R,S)}$ proceeds as follows:

1. Upon receiving an input $(\mathtt{Send}, sid, m)$ from party S, do: If $sid = (S, R, sid')$ for some R, then record m and output $(\mathtt{Send}, sid, m)$ to $\mathcal{S}_{(S,R)}$.
2. Upon receiving "approve" from $\mathcal{S}_{(S,R)}$, if m is recorded provide $(\mathtt{Send}, sid, m)$ to $\mathcal{S}_{(R,S)}$, and after $\mathcal{S}_{(R,S)}$ sends $(\mathtt{Send}, sid, m')$, output $(\mathtt{Send}, sid, m')$ to R and halt. Otherwise, provide $(\mathtt{Send}, sid, \perp)$ to $\mathcal{S}_{(R,S)}$ and halt. (Both adversaries control the channel delay.)
3. Upon receiving $(\mathtt{Corruptsend}, sid, m')$ from $\mathcal{S}_{(S,R)}$, if S is corrupt and m was not yet delivered to $\mathcal{S}_{(R,S)}$, then output $(\mathtt{Send}, sid, m')$ to $\mathcal{S}_{(R,S)}$, and after $\mathcal{S}_{(R,S)}$ sends $(\mathtt{Send}, sid, m'')$, output $(\mathtt{Send}, sid, m'')$ to R and halt.

Figure 4: The ideal functionality $\mathcal{F}_{\overline{\mathsf{den}}}$

THEOREM 3. *Let π be some protocol. Then π LUC-realizes $\mathcal{F}_{\overline{\mathsf{den}}}$ if and only if π is Bi-deniable.*

The proof of Thm. 3 is simply an adaptation of the proof of Thm. 2 to the LUC setting. We note that Thm. 3 can be easily adapted to the multiparty setting.

4. ANONYMOUS CHANNELS

Anonymous channels allow users to exchange messages without revealing their identities. The modern study of anonymous channels was started in [11] with *mix-nets*. In a mix-net protocol the vector of all parties encrypted messages are sent trough a set of *mixers*. Each mixer perform an operation on cyphertexts (usually partial decryption or reencryption) and send a random permutation to the next mixer. Finally the last mixer publish a permutation of the vector of parties messages. Several modifications have been proposed to mix-nets since Chaum's seminal paper, increasing tolerance to dishonest parties, robustness and many other desirable properties.
To realize our protocol we use the universal composable mix net proposed by Wikström in [26]. We remark that we have modified the original construction of [26], by demanding each mixer to check in the bulletin board that there are at least κ valid published cyphertexts before starting to mix. Basically the mix-net proceeds as follows:

1. Each sender P_i waits for mixers public keys and computes the product public key. Then each encrypt his message under the product public key, publish the cyphertext to a bulletin board an prove in zero knowledge that it is a valid cyphertext.
2. Each mix net M_j $j \in 1, \ldots, k$ discards all the published cyphertexts that are not valid, and if the number of valid published cyphertexts is less than κ, wait.
   If the number of valid published cyphertexts is at least κ then, for $l = 1, \ldots, k$ if $l = j$ the mixer partially decrypt the list of cyphertexts obtained from the bulletin board, perform a randomly chosen permutation on the list of cyphertext, publish on the bulletin board and prove in zero knowledge that the published list is a random permutation of the previous list. If $l \neq j$ the mixer must check that the permutation published by the mixer M_l is a valid one. Finally lexicographically sort the final published list and output it.

In [26] is shown that this protocol UC-realize the ideal functionality $\mathcal{F}_{MN}$, a slightly different functionality than the defined in figure 5, in the $\mathcal{F}_{KG} - hybrid$ model. Our modified mix-net UC-realizes the ideal functionality $\mathcal{F}_{MN}^{\kappa}$ in the $\mathcal{F}_{KG} - hybrid$ model.

THEOREM 4. *There exists a protocol that UC-realizes the ideal functionality $\mathcal{F}_{MN}^{\kappa}$ in the $\mathcal{F}_{KG}$-hybrid model with respect to static adversaries that only corrupt half of the mixers and under the DDH assumption in a cyclic group G_q.*

We argue that the equivalent result for theorem 4 in [26] implies theorem 4. Consider a UC real world execution of our modified mix-net protocol with the dummy adversary, and consider a class environments $\mathcal{C}$ where each $\mathcal{Z} \in \mathcal{C}$ corrupt less than $k/2$ mixer running arbitrary code on them,

The Ideal functionality $\mathcal{F}_{MN}^{\kappa}$ running with mixers $M_1, \ldots, M_k$, senders $P_1, \ldots, P_n$, and ideal adversary $\mathcal{S}$

1. Initialize a list $L = \emptyset$, and sets $J_P = \emptyset$ and $J_M = \emptyset$.
2. Suppose $(\texttt{Send}, m_i)$ $m_i \in G_q$ is received from P_i. If $i \notin J_P$, set $J_P \leftarrow J_P \cup \{i\}$, and append m_i to the list L. Then hand $(P_i, \texttt{Sent})$ to $\mathcal{S}$.
3. Suppose `Run` is received from M_j. Set $J_M \leftarrow J_M \cup \{j\}$. If $|J_M| \geq k/2$ and $|J_P| \geq \kappa$ then sort the list L lexicographically to form a list L', and hand $(M_j, \texttt{Output}, L')$ to $\mathcal{S}$ and $(\texttt{Output}, L')$ to M_l, for $l = 1$ to k. Otherwise, hand the list $(M_j, \texttt{Run})$ to $\mathcal{S}$.

Figure 5: The functionality $\mathcal{F}_{MN}^{\kappa}$

and instructs parties to send at least κ valid messages and before honest mixers send run. Clearly for this class of environments both, the construction of [26] an our modified construction, give the same view to each environment in $\mathcal{C}$. Then the simulator from [26] is still useful.
The simulator also works when $\mathcal{Z}$ is outside $\mathcal{C}$, because each environment $\mathcal{Z}' \notin \mathcal{C}$ can be simulated with another environment $\mathcal{Z} \in \mathcal{C}$. The environment $\mathcal{Z}$ postpones all instructions from $\mathcal{Z}'$ to a honest mixer to start before at least κ honest messages have been, but tell to $\mathcal{Z}'$ that it really happen. As long as the "postponed" mixer is honest there is no difference between the simulated $\mathcal{Z}'$ and the real one (because in one case the mixer(s) have received the `Run` input and is stopped, and in the other case is just stopped), and hence the random variable resulting of executing $\mathcal{Z}$ have the same distribution of the one resulting of executing $\mathcal{Z}'$. Then, quantifying above all environments in $\mathcal{C}$ is the same as quantifying above all environments, and as we argued for each environment in $\mathcal{C}$ there exist a simulator such that it can distinguish from the real protocol execution and the ideal functionality execution.

GUC secure mixnet

One might wonder if the mixnet of [26] is also GUC-secure. We note that each of the subprotocols of the protocol given in [26], excepting one, are GUC-secure because of honest majority assumptions. The only one that not is $\mathcal{F}_{KG}$. In [25] is shown that KG can be UC-realized with $\mathcal{F}_{\mathsf{crs}}$, Is this still valid in GUC? Or we mus use another kind of setup assumtion such as the *Augmented Common Reffernce String*. We left this question open, and assume that we have acces to a functionality $\mathcal{F}_{\mathsf{mn}}^{\kappa}$.

5. ANONYMOUS AUTHENTICATED CHANNELS

5.1 The $\mathcal{F}_{\mathsf{aac}}$ ideal functionality

An "anonymous authenticated channel" should allow parties to send authenticated messages to any other party without revealing their identities. We formally define an anonymous authenticated channel through the definition of an ideal functionality called $\mathcal{F}_{\mathsf{aac}}$ (figure 6). The functionality $\mathcal{F}_{\mathsf{aac}}$ just reveals the value of a sent message, but not the identities related to that message. This holds while the receiver of the message is not corrupt, but even in that situation the information revealed by $\mathcal{F}_{\mathsf{aac}}$ is completely simulatable by any one. Then the adversary can not prove to any one else that somebody sent or received a message.

Functionality $\mathcal{F}_{\mathsf{aac}}^{\kappa}$ running with shared functionality $\bar{\mathcal{G}}_{\mathsf{krk}}$ with party $P_1, \ldots, P_n$ and adversary $\mathcal{S}$ proceeds as follows

1. Initialize $\Gamma \leftarrow \emptyset$, $M \leftarrow \emptyset$.
2. Suppose $(\texttt{Send}, m_{i,j}, j)$ is received from $\tilde{P}_i$, do:
   (a) If $\tilde{P}_i$ or $\tilde{P}_j$ aren't registered in $\bar{\mathcal{G}}_{\mathsf{krk}}$ or $i = j$ then send $\perp$ to $\tilde{P}_i$.
   (b) If $\tilde{P}_j$ is corrupted then send $(\texttt{Corruptsend}, \tilde{P}_i, \tilde{P}_j, m_{i,j})$ to $\mathcal{S}$.
   (c) Else, send $(\tilde{P}_i, \texttt{Sent})$ to $\mathcal{S}$ and $(\texttt{Sent}, m_{i,j}, \tilde{P}_j)$ to $\tilde{P}_i$, and let $\Gamma \leftarrow \Gamma \cup \{(m_{i,j}, i, j)\}$ and $M \leftarrow M \cup \{(m_{i,j})\}$.
3. Suppose $(\texttt{Corruptsend}, m_{i,j}, i', j)$ is received from $\tilde{P}_i$, and $\tilde{P}_i$ is corrupted. If $\tilde{P}_{i'}$ is corrupted let $\Gamma \leftarrow \Gamma \cup \{(m_{i,j}, i', j)\}$ and let $M \leftarrow M \cup \{(m_{i',j})\}$.
4. Once that $|M| = \kappa$, for each $j \in \{1, \ldots, n\}$ let the multiset $M_j = \{(m_{i,j}, i)|, (m_i, i, j) \in \Gamma\}$, send $(\texttt{Messages}, M_j)$ to $\tilde{P}_j$, and send $(\texttt{Messages}, M)$ to $\mathcal{C}_{\mathcal{I}}$ to $\mathcal{S}$.

Figure 6: The ideal functionality $\mathcal{F}_{\mathsf{aac}}$

The last holds because functionality $\mathcal{F}_{\mathsf{aac}}$ is online deniable, guaranteed by lemma 3.

LEMMA 3. *The functionality $\mathcal{F}_{\mathsf{aac}}$ is online deniable.*

PROOF. Note that $\mathcal{F}_{\mathsf{aac}}$ 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 $\mathcal{F}_{\mathsf{aac}}$ is online deniable as consequence of Lemma 2. □

5.2 The SIGMIX protocol

A first natural attempt to realize $\mathcal{F}_{\mathsf{aac}}$ is with a protocol that uses anonymous channels to communicate message, and the ideal functionality $\mathcal{F}_{\mathsf{cert}}$ together with a certification authority that registers public keys (in this case $\bar{\mathcal{G}}_{\mathsf{krk}}$) as done in [7]. But this attempt fails as long as the shared functionality $\bar{\mathcal{G}}_{\mathsf{krk}}$ allows all parties to verify the authenticity of a pair (m, σ). Given that $\bar{\mathcal{G}}_{\mathsf{krk}}$ publicizes parties public keys, anonymity is lost by publicly binding the identity of the sender of m with (m, σ).
Providing anonymity and authenticity seems to be contradictory at first sight. But, being more careful, we note that it can be realized if:

1. The messages are signed using a random shared key.
2. Only the receiver can verify if the party P_i is the author of an authenticated message.
3. The receiver can not prove to anybody else if P_i is the author of the received message.

4. The messages are sent from senders to receivers anonymously.

To proceed, we use a modified version of the GUC-secure authentication protocol with respect to static adversaries, from [12]. Noting that they use a deniable signing process that help us to achieve points 1, 2 and 3 of our attempt to realize $\mathcal{F}_{\mathsf{aac}}$. The signing process is done through a signature that depends not only on the content of message an the identity of the sender, instead it additionally depends on the identity of the receiver allowing only the receiver to verify the authenticity of the pair (m, σ). The point 4 is achieved using Wikström's mix net described in section 4.

The SIGMIX protocol runs in the $\mathcal{F}_{\mathsf{mn}}, \bar{\mathcal{G}}_{\mathsf{krk}} - \textit{hybrid}$ model and with static adversaries. The Key Registration with Knowledge $\bar{\mathcal{G}}_{\mathsf{krk}}$ shared functionality from [12] (figure 7) provides a PKI for any protocol running concurrently with the SIGMIX protocol. We fix the key generation algorithm Gen with an algorithm that, using randomness r, sample a random element x from a cyclic group G_q, of order q and generated by g, and return the pair (g^x, x).

It is stressed that any other protocol using $\bar{\mathcal{G}}_{\mathsf{krk}} - \textit{hybrid}$ (that is, a protocol in Φ) might share (sk, pk) pairs with SIGMIX as long as they don't publicize secret keys.

On the other hand, we consider the functionality $\mathcal{F}_{\mathsf{mn}}$ as a traditional UC ideal functionality, meaning that each instance of $\mathcal{F}_{\mathsf{mn}}$ is local to each calling protocol.

To proceed with SIGMIX, each sender P_i signs a message m_i to P_j with a MAC using a shared secret key $k_{i,j}$ between P_i and P_j. Suppose that P_i and P_j have registered pairs of secret key/public keys $(x_i, y_i = g^{x_i})$ and $(x_j, y_j = g^{x_j})$ such that $x_i, x_j \in \mathbb{Z}_q$. Then the shared secret key $k_{i,j}$ can be non-interactively computed by P_i with $k_{i,j} = y_j^{x_i}$ and by P_j with $k_{ij} = y_i^{x_j}$. The signed message $(m_i, \sigma_{i,j} = \mathsf{MAC}_{k_{i,j}(m_i)})$ is sent to P_j using the mix net and finally P_j can check the authenticity recalculating the MAC. The SIGMIX protocol is described in figure 8.:

5.3 Proof of security

Before we prove security of SIGMIX, we will prove the next proposition about the *Multi Decisional Diffie-Hellman* assumption [2].

PROPOSITION 1. *Let G_q be a cyclic group where the DDH assumption holds, then the multiparty DDH (MDDH) assumption also holds, that is:*

$$(\{g^{x_i}\}_{i=1}^n, \{g^{x_i x_j}\}_{i,j=1,i\neq j}^n) \overset{c}{\approx} (\{g^{x_i}\}_{i=1}^n, \{g^{r_{i,j}}\}_{i,j=1,i\neq j}^n).$$

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 MDDH there exists an adversary D' attacking DDH such that

$$\mathrm{Adv}_{D'}^{\mathrm{DDH}}(k) \geq \frac{1}{n} \cdot \mathrm{Adv}_{D}^{\mathrm{MDDH}}(k).$$

This linear bound on the advantges is tighter than the quadratic bound given in [2]. And, to the best of our knowledge, that was the best bound for MDDH.

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 (`register`) from an honest party P_i that has not previously registered, sample $r \overset{R}{\leftarrow} 0,1$ 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 $(PKi, SKi)\mathsf{Gen}^{\lambda}(r)$ 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 $(Pi, \perp)$ to P_j .

Secret Key Retrieval: When receiving a message $(\texttt{retrievesecret}, P_i)$ from a party Pi 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 $(P_i, \perp)$.

Figure 7: The Φ-Key Registration with Knowledge shared functionality.

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

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

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 MDDH. In figure 9 we describe an adversary that chooses a random $\ell \in \{1, \ldots, n\}$ and, if D' breaks MDDH, it distinguish between hybrids $\vec{\chi}_\ell$ and $\vec{\chi}_{\ell+1}$.

Clearly, if $z = xy$ then $\vec{\gamma}_3 = \left(\{g^{\delta_i xy}\}_{i=1}^{\ell}\right)$ and thus $\vec{\chi} = \vec{\chi}_\ell$. Otherwise, if $z \in_R G_q$ then $\vec{\chi} = \vec{\chi}_{\ell+1}$. Then, the advantage of D' is given by

$$\begin{aligned}
\mathrm{Adv}_{D'}^{\mathrm{DDH}}(k) &= \left|\Pr\left[D'(g^x, g^y, g^z) = 1\right] - \Pr\left[D'(g^x, g^y, g^{xy}) = 1\right]\right| \\
&= \left|\sum_{i=1}^{n} \Pr\left[D'(g^x, g^y, g^z) = 1 | \ell = i\right] \Pr\left[\ell = i\right] - \right. \\
&\quad \left. \sum_{i=1}^{n} \Pr\left[D'(g^x, g^y, g^{xy}) = 1 | \ell = i\right] \Pr\left[\ell = i\right]\right| \\
&= \frac{1}{n}\left|\sum_{i=1}^{n} \Pr\left[D(\vec{\chi}_i) = 1\right] - \Pr\left[D(\vec{\chi}_{i+1}) = 1\right]\right| \\
&= \frac{1}{n}\left|\Pr\left[D(\vec{\chi}_1) = 1\right] - \Pr\left[D(\vec{\chi}_n) = 1\right]\right| \\
&= \frac{1}{n}\mathrm{Adv}_{D}^{\mathrm{MDDH}}(k).
\end{aligned}$$

The indistinguishability between $\left(\left(\{g^{x_i}\}_{i=1}^n\right), \left(\{g^{x_i x_j}\}_{i=1,j=i}^{n,n}\right)\right)$ and $\left(\left(\{g^{x_i}\}_{i=1}^n\right), \left(\{g^{r_{i,j}}\}_{i=1,j=i}^{n,n}\right)\right)$ is straightforward.

The protocol SIGMIX$^{\kappa}$ running with parties $P_1, \ldots, P_n$ in the $\mathcal{F}_{MN}^{\kappa}, \bar{\mathcal{G}}_{KRK} - hybrid$ proceeds as follows:

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

1. Wait for input $(\texttt{Send}, P_j, m_{i,j})$.
2. If $i = j$ return $\perp$.
3. Let x_i the secret key of P_i registered on $\bar{\mathcal{G}}_{KRK}$. If P_i is not registered return $\perp$.
4. Hand $(\texttt{Retrieve}, P_j)$ to $\bar{\mathcal{G}}_{KRK}$ and let y_j the answer.
5. If the answer was $\perp$ then return $\perp$. Else compute $k_{i,j} \leftarrow y_j^{x_i}$ and then compute $\sigma_{i,j} = \mathrm{MAC}_{k_{i,j}}(m_{i,j})$.
6. Hand $(\texttt{Send}, m_{i,j}||\sigma_{i,j})$ to $\mathcal{F}_{MN}$.
7. Return $(\texttt{Sent}, m_{i,j}, P_j)$

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

1. Wait for an input $(\texttt{Output}, L)$ from $\mathcal{F}_{MN}$.
2. Let $y_1, \ldots, y_n$ the public keys of all parties participating in the protocol. For each $i \in \{1, \ldots, n\}$ compute shared secret $k_{i,j} \leftarrow y_i^{x_j}$
3. Let the multiset $M_j \leftarrow \emptyset$. For each $(m, \sigma) \in L$ and for each k_{ij}, if $\sigma = \mathrm{MAC}_{k_{ij}}(m)$ then $M_j \leftarrow M_j \uplus \{(m, i)\}$.
4. Return $(\texttt{Messages}, M_j)$.

Mixer M_i**:** Each mixer send $(\texttt{Run})$ to $\mathcal{F}_{MN}$ at the beginning of the protocol execution.

Figure 8: The protocol SIGMIX

□

In order to simplify the analysis we introduce a class of adversaries that do not play *replay attacks*. Replay attacks are attacks where past executions of the protocol can be used again, but in the same way that they were originally used. For example in our case, suppose that sender S sent an authenticated message m to the receiver R. A replay attack occurs when the adversary $\mathcal{A}$ can successfully send the message m to R as it comes from S, but nothing more than that. We note that SIGMIX is secure only with respect to adversaries that do not play reply attacks, but it can be easily adapted to tolerate that kind of adversaries. Indeed, assume that SIGMIX securely realizes $\mathcal{F}_{\mathsf{aac}}$ with respect to adversaries that do not play replay attacks, in order to make useless the past authenticated messages (of th form $(m, \mathsf{MAC}_y(m))$, where y is the shared key) we add state to senders and receivers. Therefore, to sign the i_R-th message to R the sender S computes $(m, \mathsf{MAC}_y(m||i_R))$. In order to verify the i_S-th message (m, σ) from S the receiver checks that $\mathsf{MAC}(m||i_S) = \sigma$. Note state of the senders and receiver are the indexes of received messages for each receiver/sender. The security of SIGMIX is guaranteed by Theorem 5.

THEOREM 5. *Suppose that MAC is UF-CMA secure and that DDH holds in G_q, then SIGMIX GUC-realizes the ideal functionality $\mathcal{F}_{\mathsf{aac}}$ in the $\bar{\mathcal{G}}$-hybrid model with respect to 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 and a forger $\mathcal{D}_{\mathsf{MAC}}$ such that for all k large enough

$$n \cdot \mathrm{Adv}^{\mathrm{DDH}}_{\mathcal{D}_{\mathsf{DDH}}}(k) + n(n-1) \cdot \mathrm{Adv}^{\mathrm{UF-CMA}}_{\mathcal{D}_{\mathsf{MAC}},\mathsf{MAC}}(k) \geq$$

$$\left| \begin{array}{l} \Pr\left[\mathsf{EXEC}^{\bar{\mathcal{G}}_{\mathsf{krk}},\mathcal{F}_{\mathsf{mn}}}_{\mathcal{Z},\mathcal{A},\mathsf{SIGMIX}}(k) = 1\right] - \\ \Pr\left[\mathsf{EXEC}^{\bar{\mathcal{G}}_{\mathsf{krk}},\mathcal{F}_{\mathsf{aac}}}_{\mathcal{Z},\mathcal{S},\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{aac}}}}(k) = 1\right] \end{array} \right|$$

PROOF. The proof proceeds through the indistinguishability of three games: $\mathsf{Game}_{real}, \mathsf{Game}_{rand}$ and Game_{ideal}. Let $\mathcal{Z}$ be an environment and $\mathcal{A}$ an adversary.
Game_{real} consist of an execution of SIGMIX with environment $\mathcal{Z}$ and adversary $\mathcal{A}$ in the real world. Game_{rand} is the same as Game_{real} except that the shared keys between parties are chosen randomly. Game_{ideal} consist of an execution of $\mathcal{F}_{\mathsf{aac}}$ with environment $\mathcal{Z}$ and the simulator $\mathcal{S}$ (defined in figure 11) in the ideal world.
We let the output of each game be the output of the environment, that is:

$$\mathsf{Game}_{real} \stackrel{\mathrm{def}}{=} \mathsf{EXEC}^{\bar{\mathcal{G}}_{\mathsf{krk}},\mathcal{F}_{\mathsf{mn}}}_{\mathcal{Z},\mathcal{A},\mathsf{SIGMIX}}(k),$$

$$\mathsf{Game}_{rand} \stackrel{\mathrm{def}}{=} \mathsf{EXEC}^{\bar{\mathcal{G}}_{\mathsf{krk}},\mathcal{F}_{\mathsf{mn}}}_{\mathcal{Z},\mathcal{A},\mathsf{SIGMIX}_{rand}}(k),$$

$$\mathsf{Game}_{ideal} \stackrel{\mathrm{def}}{=} \mathsf{EXEC}^{\bar{\mathcal{G}}_{\mathsf{krk}},\mathcal{F}_{\mathsf{aac}}}_{\mathcal{Z},\mathcal{S},\mathsf{IDEAL}_{\mathcal{F}}}(k).$$

SIGMIX_{rand} is almost equal to SIGMIX, except that for each party P_i computing the shared key with $k_{i,j} = (g^{x_j})^{x_i}$ another party P_j we replace $k_{i,j}$ with a random $r_{i,j} \in G_q$. The same value is replaced with any other computation of $k_{i,j}$ and $k_{j,i}$ (done by P_j). The same is done with each concurrent execution of SIGMIX.

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

1. $\ell \overset{R}{\leftarrow} \{1, \ldots, n\}$.
2. Compute public keys for parties in $\{P_1, \ldots, P_\ell\}$:
   (a) $\delta_1 \leftarrow 1$.
   (b) $\delta_i \overset{R}{\leftarrow} G_q$ 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 $x_i \overset{R}{\leftarrow} G_q$ for $i = \ell + 2$ to n.
5. Let γ_1 bet the set of all public keys, $\vec{\gamma}_1 \leftarrow (\{g^{x_i}\}_{i=1}^{n})$.
6. Compute the shared keys for parties in $\{P_1, \ldots, P_\ell\}$:
   (a) $r_{i,j} \overset{R}{\leftarrow} G_q$ for $i = 1$ to ℓ and $j = i$ to ℓ.
   (b) $\vec{\gamma}_2 \leftarrow \left(\{g^{r_{i,j}}\}_{i=1,j=i}^{\ell,\ell}\right)$.
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) $\vec{\gamma}_3 \leftarrow \left(\{g^{r_{i,\ell+1}}\}_{i=1}^{\ell}\right)$.
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) $\vec{\gamma}_4 \leftarrow \left(\{g^{x_i x_j}\}_{i=1,j=\ell+2}^{\ell,n}\right)$.
   (c) $\vec{\gamma}_5 \leftarrow \left(\{g^{x_i x_j}\}_{i=\ell+2,j=i}^{n,n}\right)$.
9. $\vec{\chi} \leftarrow (\vec{\gamma}_1, \vec{\gamma}_2, \vec{\gamma}_3, \vec{\gamma}_4, \vec{\gamma}_5)$.
10. Run $D(\vec{\chi})$ and output whatever it outputs.

Figure 9: Adversary D' attacking DDH

Suppose that

$$\left|\Pr\left[\mathsf{Game}_{real}=1\right]-\Pr\left[\mathsf{Game}_{rand}=1\right]\right|=\epsilon,$$

then there exists a MDDH distinguisher $\mathcal{D}_{\mathsf{MDDH}}$ such that $\mathrm{Adv}^{\mathrm{MDDH}}_{\mathcal{D}_{\mathsf{MDDH}}}(k)=\epsilon$. Indeed, consider the MDDH distinguisher defined in figure 10. Clearly, if $r_{i,j}=x_ix_j$ the output of $\mathcal{D}_{\mathsf{MDDH}}$ follows the same distribution of Game_{real}, and if $r_{i,j}\in_R\mathbb{Z}_q$ the output of $\mathcal{D}_{\mathsf{MDDH}}$ follows the same distribution of Game_{rand}. Then it must be that

$$\mathrm{Adv}^{\mathrm{MDDH}}_{\mathcal{D}_{\mathsf{MDDH}}}(k)=\epsilon.$$

By proposition 1 there exist an adversary $\mathcal{D}_{\mathsf{DDH}}$ such that

The adversary $\mathcal{D}_{\mathsf{MDDH}}$ on input $(\{g^{x_i}\}_{i=1}^n,\{g^{r_{i,j}}\}_{i=1,j>i}^n)$ attacking MDDH over G_q proceeds as follows:

1. Initialize $M_i\leftarrow\emptyset$ for all $i\in\{1,\ldots,n\}$.
2. Simulate an execution of Game_{real}.
3. When a the party $\tilde{P}_i$ $i\in\{1,\ldots,n\}\setminus I_{\mathcal{A}}$ registers on $\bar{\mathcal{G}}_{KRK}$ set the registered public key to g^{x_i}.
4. If $\mathcal{Z}$ sends $(\mathtt{Send},m,j)$ to P_i, replace P_i's computed shared key with $k_{i,j}=g^{r_{i,j}}$.
5. When the simulation halts return whatever $\mathcal{Z}$ returns.

Figure 10: The MDDH distinguisher $\mathcal{D}_{\mathsf{MDDH}}$

$$\mathrm{Adv}^{\mathrm{DDH}}_{\mathcal{D}_{\mathsf{DDH}}}(k)\geq\frac{1}{n}\mathrm{Adv}^{\mathrm{MDDH}}_{\mathcal{D}_{\mathsf{MDDH}}}(k).$$

Let $I_{\mathcal{A}}\subseteq\{1,\ldots,n\}$ be the set indexes of parties corrupted by $\mathcal{A}$ and let $I^M_{\mathcal{A}}\subseteq\{1,\ldots,k\}$ the set of indexes of mixers corrupted by $\mathcal{A}$. The ideal adversary $\mathcal{S}^{\mathcal{A}}$ is described in figure 11, and it simulates the execution of SIGMIX only with access to $\mathcal{F}_{AAC}$.
As the values of honest messages (the sender and the receiver is honest) remains unknown to $\mathcal{S}^{\mathcal{A}}$ until all messages are sent, $\mathcal{S}^{\mathcal{A}}$ cheats the simulated $\mathcal{A}$ making $\mathcal{F}_{MN}$ tells $\mathcal{A}$ that the message was sent. When the set of honest sent messages M is revealed to $\mathcal{S}^{\mathcal{A}}$, it silently makes the simulated parties send the messages to $\mathcal{F}_{MN}$. This seems to $\mathcal{A}$ indistinguishable from an execution where an hypothetical adversary $\mathcal{S}'^{\mathcal{A}}$ guesses the messages sent from $\mathcal{Z}$ to each honest party, as the strings seen by $\mathcal{A}$ are the same in both experiments.

Suppose that

$$\left|\Pr\left[\mathsf{Game}_{rand}=1\right]-\Pr\left[\mathsf{Game}_{ideal}=1\right]\right|=\delta. \qquad (1)$$

As the view of the simulated $\mathcal{A}$ in Game_{ideal} is the same view of the view of $\mathcal{A}$ in Game_{rand}, then the only possible difference on the view of $\mathcal{Z}$ is due to a difference in the outputs of the protocol. Specifically this difference is possible because in the line 2 of the simulation of corrupted parties in $\mathcal{S}$, some messages $m||\sigma$ could be dropped. The set of all dropped messages contains all possible forgery. But if $\mathcal{A}$ does not forges then the the output of $\mathcal{Z}$ in Game_{rand} must

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

At the beginning $\mathcal{S}$ corrupts parties $\tilde{P}_i$ $i\in I_{\mathcal{A}}$ and and mixers M_i $i\in I^{\mathcal{M}}_{\mathcal{A}}$, then simulates an execution of Game_{rand} with environment $\mathcal{Z}'$, where $\mathcal{Z}'$ is machine controlled by $\mathcal{S}$. $\bar{\mathcal{G}}_{KRK}$ and $\mathcal{F}_{MN}$ are honestly simulated with some minor modifications.

Simulation of links $(\mathcal{Z}',\mathcal{A})$ with $(\mathcal{Z},\mathcal{S})$:
If m is received from $\mathcal{Z}$ then make $\mathcal{Z}'$ send m to $\mathcal{A}$. If m is sent from $\mathcal{A}$ to $\mathcal{Z}'$ then send m to $\mathcal{Z}$

Simulation of corrupted parties $\tilde{P}_i$ $i\in I_{\mathcal{A}}$:

1. If P_i $i\in I_{\mathcal{A}}$ send $m||\sigma$ to $\mathcal{F}_{MN}$ and $\sigma=\mathrm{MAC}_{y_j^{x_{i'}}}(m)$ 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 $(\mathtt{Corruptsend},m,i',j)$ to $\tilde{P}_l$.
2. If $\sigma\neq\mathrm{MAC}_{y_j^{x_{i'}}}(m)$ for all registered public keys y_j $j\in\{1,\ldots,n\}$ and all registered secret keys $x_{i'}$ $i'\in I_{\mathcal{A}}$ do nothing.

Simulation of honest parties P_i $i\in I_{\mathcal{A}}$:

1. If $(\tilde{P}_i,\mathtt{Sent})$ is received from $\mathcal{F}_{MN}$ then make the simulated $\mathcal{F}_{MN}$ sends $(\tilde{P}_i,\mathtt{Sent})$ to $\mathcal{A}$.
2. If $(\mathtt{Corruptsend},\tilde{P}_i,\tilde{P}_j,m_{i,j})$ is received from $\mathcal{F}_{AAC}$ make $\mathcal{Z}'$ sends to $\tilde{P}_i$ the message $(\mathtt{Send},m,j)$. When P_i ask for his secret key the simulated $\bar{\mathcal{G}}_{KRK}$ must answer P_j secret key ($\mathcal{S}^{\mathcal{A}}$ knows it because P_j is corrupted), and when P_i ask for P_j's public key the simulated $\bar{\mathcal{G}}_{KRK}$ must answer P_i's public key.
3. Once $(\mathtt{Messages},M)$ is received from $\mathcal{F}_{AAC}$, modify the list L of $\mathcal{F}_{MN}$ to be equal to M.

Figure 11: The ideal adversary $\mathcal{S}$

be the same in Game_{ideal}. Let $\mathcal{A}$ forges be the event wen $\mathcal{A}$ forges at least one signature, then

$$\Pr\left[\mathsf{Game}_{rand}=1|\overline{\mathcal{A}\text{ forges}}\right]=\Pr\left[\mathsf{Game}_{ideal}=1|\overline{\mathcal{A}\text{ forges}}\right].$$

With this in mind we express $\Pr[\mathsf{Game}_{rand}=1]$ and $\Pr[\mathsf{Game}_{ideal}=1]$ in terms of the event $\mathcal{A}$ forges.

$$\begin{aligned}\Pr\left[\mathsf{Game}_{rand}=1\right]=&\\ &\Pr\left[\mathsf{Game}_{rand}=1|\mathcal{A}\text{ forges}\right]\Pr\left[\mathcal{A}\text{ forges}\right]+\\ &\Pr\left[\mathsf{Game}_{rand}=1|\overline{\mathcal{A}\text{ forges}}\right]\Pr\left[\overline{\mathcal{A}\text{ forges}}\right],\\ \Pr\left[\mathsf{Game}_{ideal}=1\right]=&\\ &\Pr\left[\mathsf{Game}_{ideal}=1|\mathcal{A}\text{ forges}\right]\Pr\left[\mathcal{A}\text{ forges}\right]+\\ &\Pr\left[\mathsf{Game}_{ideal}=1|\overline{\mathcal{A}\text{ forges}}\right]\Pr\left[\overline{\mathcal{A}\text{ forges}}\right].\end{aligned}$$

Then

$$\begin{aligned}&\left|\Pr\left[\mathsf{Game}_{rand}=1\right]-\Pr\left[\mathsf{Game}_{ideal}=1\right]\right|\\ &\quad=\quad \Pr\left[\mathcal{A}\text{ forges}\right]\cdot\\ &\qquad\left|\Pr\left[\mathsf{Game}_{rand}=1|\mathcal{A}\text{ forges}\right]-\Pr\left[\mathsf{Game}_{ideal}=1|\mathcal{A}\text{ forges}\right]\right|\\ &\quad\leq\quad \Pr\left[\mathcal{A}\text{ forges}\right]\end{aligned}$$

Combining this with equation 1 gives

$$\Pr\left[\mathcal{A}\text{ forges}\right]\geq\delta.$$

In figure 12 we construct a forger $\mathcal{D}_{\mathsf{MAC}}$ whose advantage is polynomially related with $\Pr\left[\mathcal{A}\text{ forges}\right]$. Indeed,

$$\begin{aligned}\mathrm{Adv}^{\mathrm{UF-CMA}}_{\mathcal{D}_{\mathsf{MAC}},\mathsf{MAC}}(k)\quad&\geq\quad \Pr\left[\mathcal{D}_{\mathsf{MAC}}\text{ forges}|\mathcal{A}\text{ forges}\right]\Pr\left[\mathcal{A}\text{ forges}\right]\\ &=\quad \frac{\Pr\left[\mathcal{A}\text{ forges}\right]}{n(n-1)}\cdot\\ &\qquad\sum_{\substack{i,j=1\\ j\neq i}}^{n}\Pr\left[\mathcal{D}_{\mathsf{MAC}}\text{ forges}|\right.\\ &\qquad\qquad\left.\mathcal{A}\text{ forges}\wedge(i^*,j^*)=(i,j)\right]\end{aligned}$$

The forger $\mathcal{D}_{\mathrm{MAC}}$ attacking unforgeability of MAC with oracle access to MAC_k, $k\in_R G_q$:

1. Simulate an execution of Game_{rand}.
2. Choose $i^*,j^*\in_R\{1,\ldots,n\}\backslash I_{\mathcal{A}}$, $i^*\neq j^*$ and let $\hat{M}_{j^*}\leftarrow\emptyset$.
3. Whenever party P_{i^*} calls MAC with key $g^{r_{i^*,j^*}}$ and message m replace the result with $\mathrm{MAC}_k(m)$ and let $\hat{M}_{j^*}\leftarrow\hat{M}_{j^*}\uplus\{m,\mathrm{MAC}_k(m)\}$.
4. When $\mathcal{F}_{MN}$ publish the list L drop all messages $m||\sigma$ such that $\sigma=\mathrm{MAC}_{g^{r_{i,j}}}(m)$, $i\neq i^*$ or $j\neq j^*$, and drop all messages in $\hat{M}_{j^*}$. With the other messages form a list M'_{j^*}.
5. Verify each $m||\sigma$ in M'_{j^*}. When a valid one is founded return it.
6. Else, abort.

Figure 12: Forger $\mathcal{D}_{\mathsf{MAC}}$ attacking MAC

Given that in the event "$\mathcal{A}$ forges" $\mathcal{A}$ forges at least one signature for some P_i and P_j (it can not be signature seen previously by $\mathcal{A}$ because it does not play replay attacks), then at least one term in the sum must be exactly 1. Then

$$\begin{aligned}\mathrm{Adv}^{\mathrm{UF-CMA}}_{\mathcal{D}_{\mathsf{MAC}},\mathsf{MAC}}(k)\quad&\geq\quad\frac{\Pr\left[\mathcal{A}\text{ forges}\right]}{n(n-1)}\\ &\geq\quad\frac{\delta}{n(n-1)}.\end{aligned}$$

Combining the results

$$\begin{aligned}n\cdot\mathrm{Adv}^{\mathrm{DDH}}_{\mathcal{D}_{\mathsf{DDH}}}(k)+&\\ n(n-1)\cdot\mathrm{Adv}^{\mathrm{UF-CMA}}_{\mathcal{D}_{\mathsf{MAC}},\mathsf{MAC}}(k)\quad&\geq\quad\epsilon+\delta\\ &\geq\quad\left|\begin{matrix}\Pr\left[\mathsf{Game}_{real}=1\right]-\\ \Pr\left[\mathsf{Game}_{ideal}=1\right]\end{matrix}\right|.\end{aligned}$$

By unforgeability of MAC and the assumption that DDH holds in G_q, both advantages must be negligible. Then the advantage of $\mathcal{Z}$ distinguishing SIGMIX from $\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{aac}}}$ must be negligible too. □

6. REFERENCES

[1] 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.

[2] 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.

[3] Canetti, Dwork, Naor, and Ostrovsky. Deniable encryption. In *CRYPTO: Proceedings of Crypto*, 1997.

[4] Canetti, Kushilevitz, and Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *JCRYPTOL: Journal of Cryptology*, 19, 2006.

[5] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000.

[6] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[7] R. Canetti. Universally composable signature, certification, and authentication. In *CSFW*, page 219. IEEE Computer Society, 2004.

[8] 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.

[9] R. Canetti and M. Vald. Universally composable security with local adversaries. Cryptology ePrint Archive, Report 2012/117, 2012. http://eprint.iacr.org/.

[10] R. Canetti and M. Vald. Universally composable security with local adversaries. In *Proceedings of the 8th international conference on Security and Cryptography for Networks*, pages 281–301. Springer-Verlag, 2012.

[11] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[12] 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.

[13] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In *Advances in Cryptology-EUROCRYPT 2004*, pages 609–626. Springer, 2004.

[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] Dwork, Naor, and Sahai. Concurrent zero-knowledge. *JACM: Journal of the ACM*, 51, 2004.

[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] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography from anonymity. Cryptology ePrint Archive, Report 2006/084, 2006.

[18] 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.

[19] Y. Lindell. Anonymous authentication. *Journal of Privacy and Confidentiality*, 2(2):4, 2007.

[20] M. Naor. Deniable ring authentication. In M. Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 481–498. Springer, 2002.

[21] Pass. On deniability in the common reference string and random oracle model. In *CRYPTO: Proceedings of Crypto*, 2003.

[22] 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.

[23] 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.

[24] 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.

[25] Wikstrom. Universally composable DKG with linear number of exponentiations. In *International Conference on Security in Communication Networks, SCN, LNCS*, volume 4, 2004.

[26] Wikstrom. A universally composable mix-net. In *Theory of Cryptography Conference (TCC), LNCS*, volume 1, 2004.

[27] A. C. Yao and Y. Zhao. Deniable internet key exchange. In *Applied Cryptography and Network Security*, pages 329–348. Springer, 2010.

[28] A. C.-C. Yao, F. F. Yao, and Y. Zhao. A note on universal composable zero-knowledge in the common reference string model. *Theor. Comput. Sci*, 410(11):1099–1108, 2009.