A GUC-secure protocol for authenticated anonymous channels

[Extended Abstract] *

Alonso González U. †
asdasd
asdasda
asdasd
alogonza@ing.uchile.cl

Charles Palmer
Palmer Research Laboratories
8600 Datapoint Drive
San Antonio, Texas 78229
cpalmer@prl.com

ABSTRACT

This paper provides a sample of a LaTeX document which conforms to the formatting guidelines for ACM SIG Proceedings. It complements the document *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX*. This source file has been written with the intention of being compiled under LaTeX2$_\epsilon$ and BibTeX.

The developers have tried to include every imaginable sort of "bells and whistles", such as a subtitle, footnotes on title, subtitle and authors, as well as in the text, and every optional component (e.g. Acknowledgments, Additional Authors, Appendices), not to mention examples of equations, theorems, tables and figures.

To make best use of this sample document, run it through LaTeX and BibTeX, and compare this source code with the printed output produced by the dvi file.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Theory

Keywords

ACM proceedings, LaTeX, text tagging

*A full version of this paper is available as *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX* at `www.acm.org/eaddress.htm`

†asdasdasd

1. INTRODUCTION

2. CRYPTOGRAPHIC MODEL

We prove the security guarantees of our protocol in the Generalized Universal Composability (GUC) cryptographical framework. The GUC framework [4] is a generalization of the Universal composability framework [2]. Both frameworks models concurrent protocols, but the GUC framework also models concurrent protocols sharing state between them. Before the GUC framework were proposed an alternative way to model protocols sharing state in the UC framework was the use of JUC theorem, but this is not as general as it only allows protocols sharing state among themselves. Instead the GUC framework models protocols sharing state with unpredictable other protocols.

We first review the UC frameworks as the GUC framework is exactly as UC with some minor, but significant, modifications.

2.1 The Universal Composability framework (UC)

The UC framework is a methodology for a modularized designing and modularized proving the security of *real world* cryptographics protocols. The spirit of UC is to create a protocol and then 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 then real protocol is as secure as the idealized protocol.

Real world protocols are modeled as protocols running in a complete concurrent setting with many other protocols and possibly distributed among many parties. As in the real world the protocol can be monitored and some participants can have "undesired" behavior, attempting with the desired security of the protocol. All the possible undesired behavior executed by one machine, the real adversary $\mathcal{A}$. In an execution of the protocol the adversary can eavesdrop and manipulate all the data sent from one party to another, can corrupt some parties and then execute arbitrary code on them. On the other hand the ideal protocol, called the ideal functionality usually denoted by $\mathcal{F}$, runs in an ideal setting, that is executed by a secure party and the monitoring is restricted. In the ideal world there exists an ideal adversary $\mathcal{S}$ but it cannot eavesdrop and is only allowed to stop or drop exchanged data. The setting where the ideal protocol is executed with the real adversary is known as the *real world*,

and the setting where the ideal protocol is executed with the ideal adversary is known as the *ideal world*. In both worlds concurrent execution with arbitrary other protocols is given by a special machine known as environment. In UC both the real an ideal protocol are analyzed isolated from other possibly concurrent protocols. It is considered that all the externals protocols and the outside calls to the protocol are are executed by the environment. To ensure that the real protocol achieves the desired security it must hold that that for any execution of the protocol in real world and for any adversarial strategy (that is for all environments and all real adversaries) there exists an adversarial strategy with limited resources (an ideal adversary) that has the same effect (the environment is not aware of any difference between the two strategies).

We review formal definitions given in [1]. There Wikström introduces a new machine to the UC framework, named the communication model. The communication model is in charge of all aspects of the communication in UC. This definition comes to simplify the definition of what the adversary can do in an execution, in fact the only difference between the real and the ideal adversary is the communication model. The introduction of a communication model also makes unnecessary the use of session id's as the communication models are local to each protocol instance.

Definition 1. An ITM-graph is a set $V = \{P_1, \ldots, P_t\} \subset$ ITM with a set of links E such that (V, E) is a connected graph, and no P_i is linked to any machine outside V. Let ITMG be the set of ITM-graphs.
During the execution of an ITM-graph, at most one participant is active. An active participant may deactivate itself and activate any of its neighbors, or it may halt, in which case the execution of the ITM-graph halts.

The real communication model models an asynchronous communication network, in which the adversary can read, delete, modify, and insert any message of its choice.

Definition 2. A real communication model $\mathcal{C}$ is a machine with a link l_{P_i}, to P_i for $i = 1, \ldots, k$, , and a link $l_{\mathcal{A}}$ to a real adversary $\mathcal{A}$. Its program is defined as follows.

1. If m is read on l_s where $s \in \{P_1, \ldots, P_k\}$, then (s, m) is written on $l_{\mathcal{A}}$ and $mathcal A$ is activated.
2. If (r, m) is read on $l_{\mathcal{A}}$, where $r \in \{P_1, \ldots, P_k\}$ and P_r is activated.

For simplicity we avoid to explicitly call the real communication model. When in a real world protocol we write "P_i send m to P_j" we mean "P_i send (P_j, m) to $\mathcal{C}$".
The ideal communication model below captures the fact that the adversary may decide if and when it would like to deliver a message from the ideal functionality to a participant, but it can not read the contents of the communication between participants and the ideal functionality.

Definition 3. An ideal communication model $\mathcal{C}_{\mathcal{I}}$ is a machine with a link l_{P_i}, to P_i for $i = 1, \ldots, k$, and links $l_{\mathcal{F}}$, and $l_{\mathcal{S}}$ to a (shared) ideal functionality $\mathcal{F}$ and an ideal adversary $\mathcal{S}$ respectively. Its program is defined as follows.

1. If a message m is read on l_s, where $s \in \{P_1, \ldots, P_k\}$, then (s, m) is written on $l_{\mathcal{F}}$ and $\mathcal{F}$ is activated.
2. If a message (s, m) written on $l_{\mathcal{F}}$ is returned unaltered, m is written on l_s. If not, any string read from $l_{\mathcal{F}}$ is interpreted as a list $((r_1, m_1), \ldots, (r_t, m_t))$, where $r_i \in \{\mathcal{S}, P_1, \ldots, P_k\}$. For each m_i a random string $\tau_i \in \{0,1\}^n$ is chosen, and (r_i, m_i) is stored under (τ_i). Then $((r_1, |m_1|, \tau_1), \ldots, (r_t, |m_t|, \tau_t))$, where $|m_i|$ is the bit-length of m_i, is written on $l_{\mathcal{S}}$ and $\mathcal{S}$ is activated.
3. Any string read from $l_{\mathcal{S}}$ is interpreted as a pair (b, τ), where $b \in \{0, 1\}$ and τ is an arbitrary string. If $b = 1$ and (r_i, m_i) is stored in the database under the index τ, m_i is written on l_{r_i} and r_i is activated. Otherwise $(\mathcal{S}, \tau)$ is written to $l_{\mathcal{F}}$ and $\mathcal{F}$ is activated.

An adversary can normally corrupt some subset of the participants in a protocol. A *dummy participant* is a machine that given two links writes any message from one of the links to the other. There may be many copies of the dummy participant. Following Canetti we use the $\tilde{}$-notation, e.g. $\tilde{P}$, for dummy participants.
The ideal model below captures the ideal world, i.e. the environment may indirectly interact with ideal functionalities and the ideal adversary $\mathcal{S}$ has some control over how the communication model behaves.

Definition 4. The ideal model is defined to be the map $\mathcal{I} : \text{ITM}^2 \times \text{ITM}^* \to \text{ITMG}$, where $\mathcal{I} : (\mathcal{F}, \mathcal{S}\tilde{P}_1, \ldots, \tilde{P}_k) \mapsto (V, E)$ is given by:

$$V = \{\mathcal{C}_{\mathcal{I}}, \mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k\}$$

$$E = \{(\mathcal{S}, \mathcal{C}_{\mathcal{I}}), (\mathcal{C}_{\mathcal{I}}, \mathcal{F})\} \cup \bigcup_{i=1}^{k} \{(\tilde{P}_i, \mathcal{C}_{\mathcal{I}})\}$$

If $\tilde{\pi} = (\tilde{P}_1, \ldots, \tilde{P}_k)$, we write $\mathcal{I}(\mathcal{S}, \tilde{\pi}^{\mathcal{F}})$ instead of $\mathcal{I}(\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k)$ to ease notation.
The real model is supposed to capture the properties of the real world. The participants may interact over the real communication model.

Definition 5. The real model is defined to be a map $\mathcal{R} : \text{ITM}^* \to \text{ITMG}$, where $\mathcal{R} : (\mathcal{A}, P_1, \ldots, P_k) \mapsto (V, E)$ is given by:

$$V = \{\mathcal{C}, \mathcal{A}, P_1, \ldots, P_k\}$$

$$E = \{(\mathcal{A}, \mathcal{C})\} \cup \bigcup_{i=1}^{k} \{(P_i, \mathcal{C})\}$$

Let $(V,E) = \mathcal{I}(\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k)$. Then we write $\mathcal{Z}(\mathcal{I}(\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k))$ for the ITM-graph (V', E') defined by $V' = V \cup \{\mathcal{Z}\}$, and $E' = E \cup \{(\mathcal{Z}, \mathcal{S})\} \bigcup_{i=1}^{k} \{(\mathcal{Z}, \tilde{P}_i)\}$. We use the corresponding notation in the real model case.
A hybrid model is a mix between a number of ideal and real models, and captures the execution of a real world protocol with access to some ideal functionalities. It is also a tool to modularize security proofs. It may be viewed as if we "glue" a number of ideal and real models onto an original real model.

Definition 6. Suppose that we are given $(V,E) = \mathcal{R}(\mathcal{A}, \pi)$, $\pi = (P_1, \ldots, P_k)$. Let $(V_j, E_j) = \mathcal{I}(\mathcal{S}_j, \tilde{\pi}_j^{\mathcal{F}_j})$, $\tilde{\pi}_j = (\tilde{P}_{j,1}, \ldots, \tilde{P}_{j,k})$ for $j = 1, \ldots, t$, and $(V_j, E_j) = \mathcal{R}(\mathcal{S}_j, \pi_j)$, $\pi_j = (P_{j,1}, \ldots, P_{j,k})$ for $j = t+1, \ldots, s$.
We denote by $\mathcal{H}(\mathcal{A}^{\mathcal{S}_1,\ldots,\mathcal{S}_t}, \pi^{\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_t^{\mathcal{F}_t},\pi_{t+1},\ldots,\pi_s})$ the hybrid model defined as the ITM-graph (V', E'), where

$$V' = V \cup \bigcup_{j=1}^{t} V_j, \text{ and}$$

$$E' = E \cup \bigcup_{j=1}^{t} E_j \cup \bigcup_{i=1}^{k} \left(\{(\mathcal{S}_i, \mathcal{A})\} \cup \bigcup_{j=1}^{t} \{(P_i, \tilde{P}_{j,i})\} \right)$$

Similarly as above we write $\mathcal{Z}(\mathcal{H}(\mathcal{A}^{\mathcal{S}_1,\ldots,\mathcal{S}_t}, \pi^{\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_t^{\mathcal{F}_t},\pi_{t+1},\ldots,\pi_s}))$ to denote the ITM-graph (V'', E'') defined by $V'' = V \cup \{\mathcal{Z}\}$, and $E'' = E' \cup \{\mathcal{Z}, \mathcal{A}\} \cup \cup \bigcup_{i=1}^{k} \{(\mathcal{Z}, P_i)\}$.

The UC-security is given as the inability of the environment to distinguish between the two worlds.

Definition 7. A protocol π is said to UC-realize an ideal functionality $\mathcal{F}$ if and only if for all environment $\mathcal{Z}$, for all real world adversary $\mathcal{A}$ there exists an ideal world adversary $\mathcal{S}$ and a negligible function η such the following holds

$$|\Pr[\mathcal{Z}(\mathcal{H}(\mathcal{A}, \pi)) = 1] - \Pr[\mathcal{Z}(\mathcal{I}(\mathcal{S}, \mathcal{F})) = 1]| \leq \frac{1}{\eta}$$

A protocol UC-realizing an ideal functionality $\mathcal{F}$ can be completely replaced by the ideal functionality, and nobody can realize of it. The last is guaranteed by the *Composition theorem*:

THEOREM 1. *Suppose that $\pi^{\tilde{\mathcal{F}}_1,\ldots,\tilde{\mathcal{F}}_t}$ is a protocol that UC-realize the ideal functionality $\tilde{\pi}^{\mathcal{F}}$. Let ρ^{π} be a subroutine respecting protocol which calls the protocol π. Then execution of the protocol $\rho^{\pi/\mathcal{F}}$ in the $\mathcal{F}-$hybrid model is indistinguishable from the execution of the protocol ρ^{π}*

By $\rho^{\pi/\mathcal{F}}$ we refer to the same code of the protocol ρ^{π}, but all the calls to π are replaced by calls to $\mathcal{F}$.
A subroutine respecting protocol is protocol with no shared state with other protocols and his importance is discussed next.

2.2 The Generalized UC framework (GUC)

A restriction on protocols to be UC-secure is that it must be *subroutine respecting* one. That is, the protocol and all its subroutines must not provide input or output with another protocols. 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. 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 to UC-emulate the ideal functionality. For example consider the case of zero knowledge, is known that is impossible to realize zero knowledge proofs without some setup assumption like the common random string (CRS) when only the minority of the parties is honest [**?**]. In UC the CRS is modeled as an ideal functionality $\mathcal{F}_{CRS}$ that publicizes a random string to all parties, and zero knowledge protocols are proved to UC-realize an ideal zero knowledge functionality in the $\mathcal{F}_{CRS}$. The CRS can be considered a shared variable between all protocols using it. Such ideal functionality (a trusted party) is obvious to be hard to obtain in the real world, and for that reason is unrealistic to be considered local to a single 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 $\mathcal{F}_CRS$ used for realizing a ZK protocol is shown in [8] that it leads to loose of the natural deniability of a ZK protocol. Furthermore in [10] 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 formal modification is done as follows. Consider the ITM-graph (V,E) given by $\mathcal{H}(\mathcal{A}^{\mathcal{S}_1,\ldots,\mathcal{S}_t}, \pi^{\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_r^{\mathcal{F}_r},\tilde{\pi}_{r+1}^{\bar{\mathcal{G}}_{r+1}},\ldots,\tilde{\pi}_t^{\bar{\mathcal{G}}_t},\pi_{t+1},\ldots}$ we write $\mathcal{Z}(V,E)$ to denote the ITM-graph (V'', E'') defined by $V'' = V \cup \{\mathcal{Z}\}$, and $E'' = E' \cup \{\mathcal{Z}, \mathcal{A}\} \cup \bigcup_{i=1}^{k} \bigcup_{j=r+1}^{t} \{(\mathcal{Z}, \tilde{P}_{i,j})\} \cup \bigcup_{i=1}^{k} \{(\mathcal{Z}, P_i)\}$. We remark that the set links $\bigcup_{i=1}^{k} \bigcup_{j=r+1}^{t} \{(\mathcal{Z}, \tilde{P}_{i,j})\}$
Similarly as in UC one can define GUC realization of an ideal functionality with for $\bar{\mathcal{G}}-$subroutine respecting protocols, and is also possible to demonstrate a composition theorem for $\bar{\mathcal{G}}-$subroutine respecting protocols.

2.3 Deniable authentication in the GUC framework

Deniable authentication was first defined, outside the (G)UC framework, by Dwork, Nahor and Sahai in [7]. Several modifications and generalizations have been made since then, and here we consider the definition given by Dodis, Katz, Smith and Walfish in [6], since that is the one who applies to a concurrent and distributed setting
Roughly speaking a protocol is deniable if nobody can prove that a particular session of the protocol is taking place or have ever took place. In [6] 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 mis informant

who do not have 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 on-line 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 mis informant with overwhelming probability. In the full version of [6] is demonstrated that this notion is equivalent to GUC-realize the functionality $\mathcal{F}_{auth}$. They pointed out that in the GUC framework a protocol GUC-realizing a functionality $\mathcal{F}$ is as deniable as $\mathcal{F}$. The ideal functionality $\mathcal{F}_{auth}$ is "fully simulatable", that means that the protocol can be completely simulated without the participation of any party, then the functionality $\mathcal{F}_{auth}$ is deniable. Because a judge can distinguish from the real protocol, the ideal functionality and the simulated ideal functionality.
Then we can define deniable protocols in a very similar way that a deniable authentication protocol is defined, in terms of a judge, an informant and a misinformant. Using the same arguments of [6] we can show that a protocol is deniable if it GUC-realize a fully simulatable functionality.

3. ANONYMOUS CHANNELS

Anonymous channels allow users to exchange messages without revealing their identities. Several protocols have been proposed in the literature for anonymous channels. The modern study of anonymous channels was started in [5] 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 [9]. Basically Wikström's 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. 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 [9] is shown that this protocol UC-realize the ideal functionality $\mathcal{F}_{MN}$, defined in figure 1, in the $\mathcal{F}_{KG}$ − *hybrid* model.

The Ideal functionality $\mathcal{F}_{MN}$ 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 $(P_i, \texttt{Send}, m_i)$ $m_i \in G_q$ is received from $\mathcal{C}_\mathcal{I}$. If $i \notin J_P$, set $J_P \leftarrow J_P \cup \{i\}$, and append m_i to the list L. Then hand $(\mathcal{S}, P_i, \texttt{Send})$ to $\mathcal{C}_\mathcal{I}$.

3. Suppose $(M_j, \texttt{Run})$ is received from $\mathcal{C}_\mathcal{I}$. Set $J_M \leftarrow J_M \cup \{j\}$. If $|J_M| \geq k/2$, then sort the list L lexicographically to form a list L', and hand $((\mathcal{S}, M_j, \texttt{Output}, L'), \{M_l, \texttt{Output}, L'\}_{l=1}^{k})$ to to $\mathcal{C}_\mathcal{I}$. Otherwise, hand $\mathcal{C}_\mathcal{I}$ the list $(\mathcal{S}, M_j, \texttt{Run})$

Figure 1: The functionality $\mathcal{F}_{MN}$

Functionality $\mathcal{F}_{AAC}$ running with party $P_1, \ldots, P_N$ and adversary $\mathcal{S}$ proceeds as follows

1. Initialize $\Gamma \leftarrow \emptyset$, $M \leftarrow \emptyset$ and $I = \emptyset$

2. If $(\tilde{P}_i, \texttt{Register})$ is received from $\mathcal{C}_\mathcal{I}$ then set $I \leftarrow I \cup \{i\}$

3. Suppose $(\tilde{P}_i, \texttt{Send}, m_i, j)$ is received from $\mathcal{C}_\mathcal{I}$, do:
   If $i \notin I$ or $j \notin I$ then send $(\tilde{P}_i, \perp)$ to $\mathcal{C}_\mathcal{I}$.
   If P_i or P_j are corrupted then send $(\mathcal{S}, \texttt{Sign}, m_i, i, j)$ $m_i \in G_q$ to $\mathcal{C}_\mathcal{I}$, wait until receiving $(\mathcal{S}, \texttt{Signature}, m_i, i, j, \sigma_{i,j})$ $\sigma_{i,j} \in G_q$ from $\mathcal{C}_\mathcal{I}$ and let $\Gamma \leftarrow \Gamma \cup \{(m_i, i, j, \sigma_{i,j})\}$.
   If P_i nor P_j are corrupted then choose $r \overset{R}{\leftarrow} G_q$, let $\sigma_{i,j} \leftarrow \text{MAC}_r(m)$, send $(\mathcal{S}, P_i, \texttt{Send})$ to $\mathcal{C}_\mathcal{I}$, and let $\Gamma \leftarrow \Gamma \cup \{(m_i, i, j, \sigma_{i,j})\}$ and let $M \leftarrow M \cup \{(m_i, \sigma_{i,j})\}$

4. For each $j \in \{1, \ldots, N\}$ let the multiset $M_j = \{(m_i, i) | , (m_i, i, j, \sigma_{i,j}) \in \Gamma\}$, send $(\tilde{P}_j, \texttt{Messages}, M_j)$ to $\mathcal{C}_\mathcal{I}$ and send $(\mathcal{S}, \texttt{Messages}, M)$ to $\mathcal{C}_\mathcal{I}$.

Figure 2: The ideal functionality $\mathcal{F}_{AAC}$

4. ANONYMOUS AUTHENTICATED CHANNELS

4.1 The $\mathcal{F}_{AAC}$ 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 $\mathcal{F}_{AAC}$ (figure 2).

4.2 The SIGMIX protocol

Providing anonymity and authenticity seems to be contradictory at first sight, but we note that it can be realized if:

1. The messages are signed.

2. Only the addressee can prove if the party P_i is the author of an authenticated message.

3. The addressee can't prove to anybody else if P_i is the author message.

4. The sending of messages is done in an anonymous way.

A first natural attempt to realize $\mathcal{F}_{AAC}$ notion is by simply combining an anonymous channel and a protocol GUC-realizing the ideal functionality $\mathcal{F}_{CERT}$ of [3]. But this attempt fails as the ideal functionality $\mathcal{F}_{CERT}$ allows all parties to verify the authenticity of a pair (m, σ). This brings loose of anonymity by publicly binding the identity of the sender of m with (m, σ). Furthermore each instance of $\mathcal{F}_{CERT}$ is restricted to only one signer and *.
Instead we use a modified version of the GUC-secure authentication protocol with respect to static adversaries, from [6]. Noting that they use a deniable signing process that help us to achieve point 2 of our attempt to realize $\mathcal{F}_{AAC}$. 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 used the Wikström's mix net described in section 3.
The SIGMIX protocol runs in the $\mathcal{F}_{MN}, \bar{\mathcal{G}}_{KRK} - hybrid$ model with static adversaries. The Key registration with knowledge $\bar{\mathcal{G}}_{KRK}$ shared functionality from [6] (figure **??**) provides a PKI for any protocol running concurrently with the SIGMIX protocol. It is stressed that any other protocol using $\bar{\mathcal{G}}_{KRK} - hybrid$ might share (sk, pk) pairs with SIGMIX as long as they don't publicize secret keys. On the other hand we consider the functionality $\mathcal{F}_{MN}$ as a traditional UC ideal functionality, meaning that each instance of $\mathcal{F}_{MN}$ 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_{ij} between P_i and P_j. Let G_q a cyclic group of order q where the Decisional Diffie-Hellman assumption holds, and let g a generator for G_q. 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 $x_i, x_j \in G_q$. Then the shared secret key k_{ij} can be non-interactively computed by P_i with $k_{ij} = y_j^{x_i}$ and by P_j with $k_{ij} = y_i^{x_j}$. The signed message $(m_i, \sigma_{ij} = MAC_{y_{ij}(m_i)})$ 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 3

The security of SIGMIX is guaranteed by lemma 1

LEMMA 1. *The protocol SIGMIX GUC-emulates the ideal functionality $\mathcal{F}_{AAC}$ in the $\mathcal{F}_{MN}, \bar{\mathcal{G}}_{KRK} - hybrid$ model with respect to static adversaries.*

PROOF. *(Lemma 1)* The proofs proceed as follows: For each real adversary $\mathcal{A}$ we construct an ideal adversary $\mathcal{S}^{\mathcal{A}}$ who attacks $\mathcal{F}_{AAC}$, if there exists an adversary $\mathcal{A}$ and an environment $\mathcal{Z}$ that distinguish from an execution of $\mathcal{H}(\mathcal{S}, \tilde{\pi}^{\mathcal{F}_{AAC}}, \tilde{\rho}^{\bar{\mathcal{G}}_{KRK}})$ from an execution of $\mathcal{H}(\mathcal{A}, \text{SIGMIX}, \tilde{\rho}^{\bar{\mathcal{G}}_{KRK}}, \tilde{\phi}^{\mathcal{F}_{MN}})$, then the Diffie-Hellman assumption or the unforgeability of the MAC can be broken.
Let be $I_{\mathcal{A}} \subseteq \{1, \ldots, N\}$ the set indexes of parties corrupted by $\mathcal{A}$. The ideal adversary $\mathcal{S}^{\mathcal{A}}$ is described in figure 4, and it simulates the execution of SIGMIX only with access to $\mathcal{F}_{AAC}$. As the values of honest sent messages (the sender and the receiver are honest) remains unknown to $\mathcal{S}^{\mathcal{A}}$ until all

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

Registration. Each party P_i registers as follows.

1. Wait for input (`Register`)
2. Hand (`Retrieve`, P_i) to $\bar{\mathcal{G}}_{KRK}$ and let (x_i, y_i) the answer.
3. If the answer was $(P_i, \perp)$ then let $x_i \xleftarrow{R} G_q$ and $y_i \leftarrow g^{x_i}$. Hand (`register`, x_i) to $\bar{\mathcal{G}}_{KRK}$.

Sender P_i. Each sender P_i proceeds as follows.

1. Wait for input (`Send`, P_j, m_i).
2. Hand (`Retrieve`, P_j) to $\bar{\mathcal{G}}_{KRK}$ and let y_j the answer.
3. If the answer was $\perp$ then return $\perp$. Else compute $k_{ij} \leftarrow y_j^{x_i}$ and then compute $\sigma_i = \text{MAC}_{k_{ij}}(m_i)$.
4. Hand (`Send`, $m_i || \sigma_i$) to $\mathcal{F}_{MN}$.
5. Return (`Sent`, P_j, m_i)

Receiver P_j. Each receiver P_j proceeds as follows.

1. Wait for an input (`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 $k_{ij} \leftarrow y_i^{x_j}$
3. Let the multiset $M_j \leftarrow \emptyset$. For each $(m_l, \sigma_l) \in L$ and for each k_{ij}, if $\sigma_l = \text{MAC}_{k_{ij}}(m_l)$ then $M_j \leftarrow M_j \uplus \{(m_l, l)\}$.
4. Return (`Messages`, M_j).

Figure 3: The protocol SIGMIX

messages are sent, $\mathcal{S}^{\mathcal{A}}$ cheats the simulated $\mathcal{A}$ making $\mathcal{F}_{MN}$ tells $\mathcal{A}$ that the message was sent. Finally, when the set of honest sent messages M is revealed to $\mathcal{S}$, it silently makes the simulated parties send the messages to $\mathcal{F}_{MN}$. This seems to $\mathcal{A}$ indistinguishable from an execution where an unrealistic 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.

We define Real $= \mathcal{Z}(\mathcal{H}(\mathcal{A}, \text{SIGMIX}, \tilde{\rho}^{\bar{\mathcal{G}}_{KRK}}, \tilde{\phi}^{\mathcal{F}_{MN}}))$ and Ideal $= \mathcal{Z}(\mathcal{H}(\mathcal{S}^{\mathcal{A}}, \tilde{\pi}^{\mathcal{F}_{AAC}}, \tilde{\rho}^{\bar{\mathcal{G}}_{KRK}}))$. Suppose there exists an environment $\mathcal{Z}$, an adversary $\mathcal{A}$ and a polynomial p such that:

$$|\Pr[\text{Real} = 1] - \Pr[\text{Ideal} = 1]| \geq \frac{1}{p} \tag{1}$$

Then we define in figure 5 the adversary $\mathcal{D}_{\text{DDH}}$.

Note that as in the DDH experiment $\alpha \in_R G_q$ and $\beta \in_R G_q$ then it is undistinguishable to the environment the modification made by $\mathcal{D}_{DDH}$. Then

$$\Pr[\text{Real} = 1] = \Pr[\mathcal{D}_{DDH} = 1 | \gamma = \alpha\beta] \tag{2}$$

Now, the only possible difference between executing $\mathcal{D}_{DDH}$ when $\gamma \in_R G_q$ and executing $\mathcal{Z}(\mathcal{H}(\mathcal{S}^{\mathcal{A}}, \tilde{\pi}^{\mathcal{F}_{AAC}}, \tilde{\rho}^{\bar{\mathcal{G}}_{KRK}}))$ is the output of honest parties. But if there exists an honest party with index j such that $M_j^{\text{Real}} \neq M_j^{\text{Ideal}}$, then there exists a pair $(m, l) \in M_j^{\text{Real}}$. And it must hold that $(m, l) \notin M_j^{\text{Ideal}}$ because the same messages are sent in Real and Ideal, except for those dropped by $\mathcal{S}^{\mathcal{A}}$ in step 1 of the simulation of corrupted parties. The previous means that the adversary created a signature $\sigma \neq \text{MAC}_{y_i^{x_l}}(m)$ for all $l \in I_{\mathcal{A}}$, and as it was accepted by P_j it must be that $\sigma = \text{MAC}_{g^{\gamma r_i r_j}}(m)$ for some $i \notin I_{\mathcal{A}}$. The latter contradicts unforgeability of MAC, then:

$$\Pr[\text{Ideal} = 1] \approx \Pr[\mathcal{D}_{DDH} = 1 | \gamma \in_R G_q] \tag{3}$$

Replacing equation 2 and 3 in 1 contradicts the assumption that DDH holds in G_q. □

Ideal adversary $\mathcal{S}^{\mathcal{A}}$ running with parties $\tilde{P}_1, \ldots, \tilde{P}_N$ and shared functionality $\bar{\mathcal{G}}_{KRK}$ proceeds as follows:

At the beginning $\mathcal{S}^{\mathcal{A}}$ corrupts parties $\tilde{P}_i$ $i \in I_{\mathcal{A}}$ and executes a simulation of $\mathcal{Z}'(\mathcal{H}(\mathcal{A}, \text{SIGMIX}, \tilde{\pi}^{\bar{\mathcal{G}}_{KRK}}, \tilde{\rho}^{\mathcal{F}_{MN}}))$, where $\mathcal{Z}'$ is machine controlled by $\mathcal{S}^{\mathcal{A}}$, and $\bar{\mathcal{G}}_{KRK}$ and $\mathcal{F}_{MN}$ are honestly simulated with some minor modifications.

Simulation of link $(\mathcal{Z}', \mathcal{A})$ 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 $\sigma = \text{MAC}_{y_j^{x_l}}(m)$ for some registered public key y_j $j \in \{1, \ldots, N\}$ and some registered secret key x_l $l \in I_{\mathcal{A}}$, then send $(\texttt{Send}, m, j)$ to $\tilde{P}_l$ wait for $(\texttt{Sign}, m, l, j)$ from $\mathcal{C}_{\mathcal{I}}$ and send $(0, \texttt{Signature}, m, l, j, \sigma)$. If $\sigma \neq \text{MAC}_{y_j^{x_l}}(m)$ for all registered public keys y_j $j \in \{1, \ldots, N\}$ and all registered secret keys x_l $l \in I_{\mathcal{A}}$ do nothing.
2. If P_i sends $(\texttt{Register}, x_i)$ to the simulated $\bar{\mathcal{G}}_{KRK}$, then send $(\texttt{Register}, x_i)$ to $\tilde{P}_i$.

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

1. Let $M'_j = \emptyset$ for $j = 1, \ldots, k$, $l_s \leftarrow 0$ and $l_r \leftarrow 0$.
2. If $(\tilde{P}_i, \texttt{Send})$ is received from $\mathcal{C}_{\mathcal{I}}$ then make $\mathcal{Z}'$ sends $(\texttt{Register})$ to P_i and make $\mathcal{F}_{MN}$ sends $(\mathcal{S}_{\mathcal{F}_{MN}}, \tilde{P}_i, \texttt{Send})$ to its own copy of $\mathcal{C}_{\mathcal{I}}$.
3. If $(\texttt{Sign}, m, i, j)$ is received from $\mathcal{C}_{\mathcal{I}}$ recover x_j and y_i from $\bar{\mathcal{G}}_{KRK}$ (not the simulated one), let $\sigma_{i,j} = \text{MAC}_{y_i^{x_j}}(m)$, make $\mathcal{Z}'$ sends $(\texttt{Register})$ to P_i and then sends $(\texttt{Send}, m, j)$ to P_i. Finally send $(0, \texttt{Signature}, m, i, j, \sigma_{i,j})$ to $\mathcal{C}_{\mathcal{I}}$
4. If $(\texttt{Messages}, M)$ is received from $\mathcal{C}_{\mathcal{I}}$ then for each $(m, \sigma) \in M$ choose $i, j \xleftarrow{R} \{1, \ldots, N\} \setminus I_{\mathcal{A}}$, make $\mathcal{Z}'$ sends $(\texttt{Send}, m, j)$ to P_i and drop the message $(\mathcal{S}, P_i, \texttt{Send})$ sent from $\mathcal{F}_{MN}$ to his own copy of $\mathcal{C}_{\mathcal{I}}$ (similarly $\mathcal{S}^{\mathcal{A}}$ can just append M to the list L of $\mathcal{F}_{MN}$).

Figure 4: The ideal adversary $\mathcal{S}^{\mathcal{A}}$

5. REFERENCES

[1] Paper completo wikstrom.

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

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

[4] 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.

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

[6] Y. Dodis, J. Katz, A. Smith, and S. Walfish. Composability and on-line deniability of authentication. In O. Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2009.

The adversary $\mathcal{D}_{DDH}(g^{\alpha}, g^{\beta}, g^{\gamma})$ attacking DDH works as follows:

1. Simulate $\mathcal{Z}(\mathcal{H}(\mathcal{A}, \text{SIGMIX}, \tilde{\rho}^{\bar{\mathcal{G}}_{KRK}}, \tilde{\phi}^{\mathcal{F}_{MN}}))$.
2. Choose $r_i, r_j \xleftarrow{R} G_q$ for $i, j \in \{1, \ldots, N\} \setminus I_{\mathcal{A}}$, when $\tilde{P}_i$ registers on $\bar{\mathcal{G}}_{KRK}$ set the registered public key to $g^{\alpha r_i}$ and when $\tilde{P}_j$ registers set the public key to $g^{\beta r_j}$.
3. When $\mathcal{Z}$ sends $(\texttt{Send}, m, j)$ to P_i then replace the signature computed by P_i with $\text{MAC}_{g^{\gamma r_i r_j}}(m)$ for $j \notin I_{\mathcal{A}}$ and $\text{MAC}_{g^{\gamma r_i x_j}}(m)$ for $j \in I_{\mathcal{A}}$.
4. When P_i $i \notin I_{\mathcal{A}}$ receives $(\texttt{Output}, L)$ from $\mathcal{F}_{MN}$ then compute each signature with $\text{MAC}_{g^{\gamma r_i r_j}}(m)$ for $j \notin I_{\mathcal{A}}$ and with $\text{MAC}_{g^{\gamma r_i x_j}}(m)$ for $j \in I_{\mathcal{A}}$.
5. When the simulation halts return the output of $\mathcal{Z}$

Figure 5: The DDH adversary $\mathcal{D}_{\text{DDH}}$

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

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

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

[10] 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.