Universally Composable Security with Local Adversaries

Ran Canetti[⋆] and Margarita Vald[⋆]

Boston University and Tel Aviv University

Abstract. The traditional approach to formalizing ideal-model based definitions of security for multi-party protocols models adversaries (both real and ideal) as centralized entities that control all parties that deviate from the protocol. While this centralized-adversary modeling suffices for capturing basic security properties such as secrecy of local inputs and correctness of outputs against coordinated attacks, it turns out to be inadequate for capturing security properties that involve restricting the sharing of information between separate adversarial entities. Indeed, to capture collusion-freeness and game-theoretic solution concepts, Alwen et al. [Crypto, 2012] propose a new ideal-model based definitional framework that involves a de-centralized adversary.

We propose an alternative framework to that of Alwen et al. We then observe that our framework allows capturing not only collusion-freeness and game-theoretic solution concepts, but also several other properties that involve the restriction of information flow among adversarial entities. These include some natural flavors of anonymity, deniability, timing separation, and information-confinement. We also demonstrate the inability of existing formalisms to capture these properties.

We then prove strong composition properties for the proposed framework, and use these properties to demonstrate the security, within the new framework, of two very different protocols for securely evaluating any function of the parties' inputs.

1 Introduction

Rigorously capturing the security properties of cryptographic protocols has proven to be a tricky endeavor. Over the years, the *trusted party* (or, simulation) paradigm has emerged as a useful and general definitional methodology. The basic idea, first coined in [GM84, GMR85, GMW87], is to say that a protocol "securely realizes" a given computational task if participating in the protocol ”emulates” the process of interacting with an imaginary "trusted party" that securely receives parties' inputs and locally computes their outputs. Intuitively, this paradigm allows expressing and capturing many security properties. Moreover, it has an attractive potential "composability" property: any system using a

⋆ Both authors are supported by the Check Point Institute for Information Security, Marie Curie grant PIRG03-GA-2008-230640, and ISF grant 0603805843.

© Springer-Verlag Berlin Heidelberg 2012

trusted party $\mathcal{F}$ should behave the same way when we replace the trusted party with the realizing protocol.

Over the years, many security definitions were based on this intuitive idea, e.g., [GL90, MR91, Can00, DM00, PW00, Can01, PS04, CDPW07]. First, these definitions formulate an execution model; then they formalize the notion of emulating an ideal task with an "ideal world" attacker, called simulator. The security requirement is based on the inability of an external observer to distinguish an "ideal world" execution from a real one.

These formalisms differ in many ways; however, they have one major thing in common: they all model the attacker as a centralized entity, who can corrupt parties, coordinate their behavior and, intuitively, constitute an "evil coalition" against the protocol being executed. This seems to be an over-simplification of real life situations. Indeed, in real life, parties are often individuals who are not necessary controlled by the same entity or have anything in common. It would seem that letting the malicious parties coordinate their attacks should be a strengthening of the model; however, when this power is also given to the adversary in the ideal model (aka the simulator), the security guarantee can potentially be weakened. Therefore, a natural question to ask is whether it is justified to model the attacker as a centralized entity or does this modeling unduly limit its expressiveness?

Indeed, the existing formalisms do capture basic properties such as privacy of inputs, and correctness of outputs against coordinated attack. However, as has been observed in the past, there exist security concerns that are not naturally captured using the centralized adversary approach. Consider for instance the *collusion- freeness* concern: a protocol is *collusion-free* if even misbehaving protocol participants cannot use the protocol to exchange "disallowed" information without being detected. As pointed out by [ILM05], "centralized simulator" formalisms do not capture the inability of parties to collude. That is, with a centralized adversary, a protocol might allow collusions between corrupted parties even when it realizes an ideal task that is collusion-free.

An additional known limitation of standard security notions is cryptographic implementations of game-theoretic mechanisms. In contrast to cryptography, game theory considers rational players that behave according to their individual goals. In many realistic settings, the incentive structure depends on whom players can collaborate with and the cost of this collaboration. Security with a centralized adversary does not guarantee that the incentive structure with respect to collaboration is preserved when moving from the ideal protocol to the one that realizes it. Consequently, it does not correctly capture the incentive structure and does not suffice for preserving game-theoretic solution concepts that restrict the formation of coalitions.

A natural way to handle those concerns would be to strengthen the model by requiring that the simulation be "local" in some sense; that is, shattering the centralized simulator to many simulators, where each simulator has only some "local" information and is responsible to simulate adversarial behavior in only a "local" sense. However, requiring local simulators while allowing the adversary to be centralized results in an unrealistically strong security requirement that fails to admit many useful schemes that have practical security guarantees. Therefore, the next promising idea would be to restrict also the adversary to be local. This approach indeed appears in the works of [LMS05, ASV08, AKL$^+$09, ILM11, MR11, AKMZ12]. In particular, [AKMZ12] gives general model with a composition theorem and application to game-theory. These works give different and incomparable definitions of collusion-freeness; a common aspect is that they all postulate an adversary/simulator for each *participant*, where a participant represents an entity that is identified via its party identifier and treated as a "single domain" (i.e., it is corrupt as a unit, either wholly or none at all). However, as we demonstrate below, there are a number of security concerns that cannot be naturally captured even by the above formalizations of local simulation.

Our Contributions. We provide an alternative formalization of the local simulators approach in a way that preserves its intuitive appeal and captures reality more tightly. In particular, we establish a general security notion that allows capturing the requirements of arbitrary tasks while preserving the local view of each individual component *and each communication link between components* in the system. This notion enables expressing variety of partitions of the system. Specifically, we refine the UC framework to deal with the locality of information available to clusters of components. The new formalism, called local UC (LUC), assigns a different adversary/simulator to each *ordered pair* of participants. Intuitively, the adversaries/simulators assigned to a pair of parties handle all the communication between the two parties. Informally,

> If π is a *LUC-secure* protocol that implements a trusted party $\mathcal{F}$, then each *individual entity* participating in π affects each other entity in the system no more than it does so in the ideal execution with $\mathcal{F}$.

Note that this is conceptually different from the guarantees provided by the UC framework of [Can01] and the Collusion-Preserving framework of [AKMZ12] (referred as CP). In the UC framework, protocols that implement a trusted party are guaranteed to have similar effect on the external environment as in the ideal execution with $\mathcal{F}$. In the CP framework, the protocol is guaranteed to have the same effect as the trusted party *individually on each entity*. In the LUC framework, it is guaranteed that *each entity affects each other entity* in the same way as in the ideal execution.

We show that this refined granularity allows LUC to capture various security concerns that cannot be captured by previous frameworks. We address some flavors of anonymity, deniability, collusion-freeness, information-confinement, and preservation of incentive structure.

We also extend the UC composition theorem and the dummy adversary theorem to the new framework. We obtain strong composition results that enable "game theoretic composition", i.e., composition that preserves the power of coalitions (whatever they may be). Moreover, our strong composition also preserves deniability and confinement.

Next we present two protocols for secure function evaluation with LUC security. The protocols, called the Physical GMW and the Mediated SFE protocols, satisfy the new security definition. The protocols are very different from each other: The Physical GMW protocol, which is strongly inspired by [ILM05], models players sitting in a room equipped with machines and jointly computing a function. The Mediated SFE protocol is a simplified version of [AKL^{+}09]. Like there, we use a semi-trusted mediator. That is, if the mediator is honest then the protocol is LUC secure. It is also UC secure in the standard sense even if the mediator is corrupted. It is interesting to note that although these two protocols have significantly different nature, they are both analyzable within our framework.

1.1 Our Contributions in More Details

The New Formalism. In a nutshell, the new modeling proceeds as follows. Recall that in the UC framework, the adversary is a centralized entity that not only controls the communication in the network, but also coordinates the corrupted parties' behavior. This centralization is also inherited by the simulator. As mentioned above, while this modeling captures privacy and correctness, which are "global" properties of the execution, it certainly does not capture rationality or locality of information. This implicitly means that only the situations where corrupted parties enjoy global view of the system are being fully captured.

A first attempt to bridge this gap might be to follow the formalisms of previous works [ASV08, AKMZ12, MR11] and consider one adversary per party. However, this modeling does not completely capture reality either. Consider for instance the following scenario: one of two parties A and B is having a conversation with a third party C. Later C is instructed to transfer this information to some honest but curious fourth party D without revealing whether the source was A or B. Clearly for the protocol to make intuitive sense, A and B need to assume that C is trusted, or "incorruptible". However, going back to the suggested model we notice that the adversary associated with C participates in all the conversations that C participates in, and can thus correlate the $C - D$ communication with the $A - C$ and $B - C$ communication without corrupting C at all, thereby harming the anonymity in a way that is not intended by the protocol (see more detailed account of this issue in Section 3). In other words, the suggested modeling does not distinguish between the "obviously insecure" protocol that allows C to be corrupted, and the "obviously secure" protocol that uses an incorruptible C. We conclude that having a single adversary per party does not faithfully model honest parties. This motivates us to look for a more refined model.

To adequately capture locality, we extend the UC model as follows: For each party identity (denoted PID) we consider an adversary for any PID it might communicate with. In other words, each pair of PIDs has a pair of adversaries, where each adversary is in a different side of the "potential communication line".

Each local adversary is in charge of a specific communication line and is aware only of the communication via this line.

Another feature of our modeling is that we let the environment directly control the communication, by letting the local adversaries communicate with each other only through the environment. This is an important definitional choice that is different from [AKMZ12]. In particular, this means that the centralized simulator no longer exists and, each local adversary is replaced with a local simulator in the ideal process, where the protocol is replaced by the trusted party. The trusted party may allow different subsets of simulators to communicate by forwarding messages between them. Therefore, the communication interface provided by the trusted party to the simulators represents partition of the system to clusters. The effect of this modeling is that the simulator for an entity can no longer rely on other parties' internal information or communication in which it was not present. This way, a proof of security relies only on each entity's local information, and potentially, represents independence of clusters defined by the trusted party.

To preserve meaningfulness, we allow the local adversaries to communicate across party identities only via the environment or with ideal functionalities. Aside from these modifications in the adversarial interface, the model is identical to the UC model.

Capturing security concerns. We discuss variety of security concerns that are captured by LUC security but not in other security notions:

Collusion-freeness. To provide initial evidence for the expressiveness of LUC, we consider any UC-secure protocol for multi-party computation (e.g. the [CLOS02] protocol). While this protocol UC realizes any ideal functionality (even ones that guarantee collusion freeness) in the presence of malicious adversaries, it allows individually corrupted parties to collude quite freely, even when the environment does not pass any information among parties. Indeed, this protocol does not LUC-realize any ideal functionality that guarantees collusion-freeness. This is so even in the presence of only semi-honest adversaries. The reason for the failure in the LUC model is the inability of the separate simulators to produce consistent views on adversaries' shared information (i.e., scheduling, committed values etc.) We note that this concern is captured by the definitions of [LMS05, ASV08, AKL$^+$09, AKMZ12] as well.

Anonymity. We consider several flavors of anonymity such as existence-anonymity, timing-anonymity, and sender-anonymity. Specifically, we show UC and CP realizations of ideal functionalities that have these anonymity requirements by a protocol that does not have these properties. We'll then show that this realization is not LUC secure. Let us informally present the above flavors of anonymity: The first anonymity concern we present is existence-anonymity. Intuitively, we would like to have a "dropbox" that does not let the recipient know whether a new message was received, and thus hides information regarding the existence of the sender.

Consider the following one-time-dropbox functionality. The dropbox is a virtual box initialized with some random file. People can put files into ones dropbox. In addition, the owner can one time query the dropbox if any new file has been received; if there are any incoming files in the box, they would be delivered to the owner; else the default file would be delivered.

Indeed, whenever receiving a file from this dropbox, there is no certainty regarding the existence of a sender. Correspondingly, any protocol that LUC-realizes the dropbox functionality is guaranteed to provide anonymity regarding the existence of a sender. This is not so for standard UC security.

An additional anonymity concern that we consider is timing-anonymity. Timing-anonymity means hiding the time in which an action took place. For example consider the following email feature: whenever sending an email, the sender can delay the sending of the email by some amount of time (say, randomly chosen from some domain).

Indeed, upon receiving an email, the receiver does not know when this email was sent. This property can be captured via an ideal functionality in a straightforward way. Again, any protocol that LUC-realizes this functionality will provide anonymity regarding the time of sending. This is not so for standard UC security.

An additional anonymity property already mentioned here is sender-anonymity. The common way to achieve this anonymity property in practice is onion routing. In the work of [CL05] the onion-routing problem is defined in the UC framework; however, they only address a potential solution to the sender-anonymity concern rather than the concern itself. In contrast, we formalize the sender-anonymity property. We also show how UC security (and even CP security) fails to capture this property. Specifically, we define an ideal functionality and show a protocol that is clearly non-anonymous but still CP-realizes the functionality according to the definition of [AKMZ12]. This protocol is not LUC secure.

Deniability. It was pointed out in [CDPW07, DKSW09] that UC security does not guarantee deniability due to issues with modeling of the PKI. While these issues were resolved in the context of global setup and deniable authentication in the generalized UC framework, it turns out that the UC formalism does not capture another deniability flavor, called *bi-deniability* (the name is taken from [OPW11]): A protocol is *bi-deniable* if the protocol participants can "deny" before a judge having participated in the protocol by arguing that any "evidence" of their participation in the protocol could have been fabricated without their involvement, *even if there exists an external entity that has an access to parties' log files of the communication.* In the context of authentication, the judge is provided with “evidences” of sender's participation not only by the receiver but also by this external entity. Specifically, the sender can argue that any “evidence” of participation was fabricated by this external entity, even though this external entity cannot communicate with the receiver and only has an access to the communication log files of the sender. This notion is stronger than standard deniability, in which sender's log files are ideally hidden from the judge. To motivate bi-deniability consider a corporation that is obligated to store its communication log files. The log files are collected by an external law enforcement agency. Clearly, it would be desirable to ensure that even if these files are disclosed, the corporation can always deny their authenticity.

In this work, we give a simulation-based definition of bi-deniable authentication and prove its equivalence to LUC secure authentication. Moreover, due to the strong connection between bi-deniability and LUC security, we obtain that bi-deniability is preserved under composition. In addition, we show that UC framework fails to capture this flavor of deniability.

Confinement. Another important concern that seems hard to capture by the standard notions is the *information confinement* property, defined by [Lam73]. A protocol is said to enforce confinement if even misbehaving participants cannot leak secret information that they possess across predefined boundaries. [HKN05] presents a game based definition of confinement. Their definition introduces changes in the basic UC model, but still considers a centralized adversary. We show that the definition of [HKN05] is excessively strong and protocols that clearly enforce confinement fail to admit it. The root of the problem is the centralized adversary that enables information flow to unauthorized entities.

Intuitively, separate adversaries controlling different parts of the network or different groups of parties would indeed capture this requirement more tightly. We present a formal definition of confinement and show that LUC security implies it. Similarly to bi-deniability, we obtain composability with respect to confinement. In addition, we show the inability of UC to capture confinement. More specifically, we show that any UC functionality that enforces confinement is *super-ideal* in some well-defined sense. As before, this is not so for LUC functionalities.

Game-theoretic implications. As pointed out in [ILM05], standard security does not suffice for implementation of general equilibria due to collusion. In order to overcome this problem, new notions of mechanism implementation were defined in [ILM05, ILM08, ILM11]. However, these notions are specific to the problem at hand and are not suitable as general definitions of security. Alwen et al. [AKMZ12] translate their security notion to the game-theoretic setting and define a corresponding model of mediated games. In addition, they show that their security notion achieves preservation of incentive-structure for mediated games.

In this work, we show how protocols modeled in the LUC framework can be viewed as games. Moreover, we show that *any protocol* that LUC-securely realizes some ideal functionality preserves the incentive structure of the realized functionality. More concretely, for any LUC-secure protocol π there exists an efficient mapping between real world strategies and ideal world strategies that can be computed by each player in a local manner and achieves indistinguishable payoffs. This in particular implies that any Nash Equilibrium (NE) in the ideal-world game is mapped to a computational NE in the real-world game and no new equilibria are introduced. A more complete description of the game-theoretic implications provided by LUC security appears in [CV12].

Composition and Dummy Adversary. We demonstrate that LUC-security is preserved under composition. Due to the local nature of the model, this preservation applies not only to basic security concerns under composition, but rather to much more general security concerns such as deniability, confinement, and game-theoretic solution concepts. The obtained game-theoretic composition implies that Nash equilibrium is preserved under concurrent composition.

We also extend the dummy adversary notion to the local UC framework, and show its equivalence to the general LUC-security notion.

An interesting line for future research is to try to cast the LUC framework within the Abstract Cryptography framework [MR11]. In particular, such a work might provide a unified basis for the LUC, CP and UC frameworks.

LUC Secure Protocols

We sketch the two secure function evaluation protocols that we analyze in this work.

The Physical GMW Protocol. The GMW version we use is the protocol from [Gol04]. Still, our construction is strongly inspired by [ILM05]. We cast the protocol in the physical world by considering a set of players sitting in a room and jointly computing a function by evaluation the gates of its circuit. In order to properly compute the function, the players use the following physical machinery: boxes with serial number, and machines for addition, multiplication, duplication, and shuffle of boxed values. In more details, Let $P_1, ..., P_n$ be a set of parties in the room and let f be the function of interest. Next:

1. *Sharing the inputs:* Each player partitions its input to random shares, one share for each player and then, it publically sends those shares, in opaque boxes, to the players.
2. *Circuit emulation:* Proceeding by the order of wires, all players jointly and publically evaluate the circuit gates.
3. *Output reconstruction:* Each player publically hands the Boxes of the output shares to the appropriate parties. Lastly, each party privately opens the boxes and computes its output.

Theorem 1 (Informal statement). *Let f be a PPT function. Then, there exists a protocol that information-theoretically LUC-securely computes f with respect to adaptive adversaries.*

Throughout the process, everyone sees which operations are performed by each player. Still, the actual values inside the boxes remain secret.

In contrast with classic GMW protocol, here the byzantine case is not done by introducing ZK proofs; rather the primitives themselves are robust.

We achieve LUC-security for any number of corrupted parties. While the work of [ILM05] requires at least one honest party for the collusion-freeness to hold, we achieve LUC security (which implies collusion-freeness) even when all the parties are corrupted. In addition, this protocol meets the strong notion of *perfect implementation* defined by [ILM11], and therefore achieves privacy, strategy, and complexity equivalence.

The Mediated-SFE Protocol. We present here a high-level description of the mediated protocol, following [AKL$^+$09].

Let $P_1, ..., P_n$, and mediator $\mathcal{M}$ be a set of parties and let π be a k-round protocol that UC-securely computes function f. (Inspired by [AKMZ12], we think of the protocol as running directly over unauthenticated communication channels.) The protocol π is compiled to a new LUC-secure protocol for computing f with a semi-trusted mediator, where all the communication is done through the mediator. Specifically, for each round of the protocol π does:

1. Each party and $\mathcal{M}$ runs two-party secure computation, which outputs to $\mathcal{M}$ the next round messages of this party in π.
2. $\mathcal{M}$ sends a commitment to the relevant messages to each party P_i.
3. In the last round of π, the mediator $\mathcal{M}$ and each party run secure two-party computation, where each party obtain its output.

Theorem 2 (Informal statement). *Given a (poly-time) function $f = (f_1, ..., f_n)$ and a protocol π that UC-securely computes f. Then there exists a protocol Π that LUC-securely computes f with respect to adaptive adversaries.*

When $\mathcal{M}$ is honest it separates the parties of π and makes them be independent of each other. When $\mathcal{M}$ is corrupted, the independence disappears. Still, we obtain standard UC security.

We strengthen the protocol to be immune to powerful adversaries that control the scheduling, gain information via leakage in the protocol, and are able to adaptively corrupt players. In contrast to [AKMZ12], we do not assume ideally secure channels between parties and the mediator.

Organization. Section 2 presents an overview of the LUC security definition and composition theorem. Section 3 presents the insufficiency of standard notions to capture interesting flavors of anonymity. Section 4 presents bi-deniability and shows its relationship to security notions. Section 5 presents confinement and states its relationship to various security notions.

2 LUC Security Definition and Composition Overview

The model of protocol execution is defined in terms of a system of ITMs as in [Can01]. At first, a set of party IDs is chosen. Then, we consider a pair of adversaries for all potentially communicating ITIs based on the chosen party IDs (the definition of ITI appears in [Can01]). In addition, jointly with the environment, these adversaries have complete control over the communication between ITIs which under their custody, as opposed to the UC framework, where a centralized adversary controls all the communication in the system. Formally, the technical difference from the UC framework is expressed in the control function, summarized below. The underlying computational model remains unchanged.

Let π be a protocol over a fixed set of parties. The model is parametrized by three ITMs: the protocol π to be executed, an environment $\mathcal{Z}$ and an adversary $\mathcal{A}$.

The initial ITM in the system is the environment $\mathcal{Z}$. The input of the initial ITM $\mathcal{Z}$ represent all the external inputs to the system, including the local inputs of all parties. As a first step, $\mathcal{Z}$ chooses a set $\mathcal{P}$ of party identities (PIDs) and session ID s. The first ITIs to be invoked by $\mathcal{Z}$ is the adversaries. An adversary with identity $id = ((i, j), \perp)$ where $i, j \in \mathcal{P}$, denoted $\mathcal{A}_{(i,j)}$, is invoke for each ordered pair $i, j \in \mathcal{P}$. The adversaries code is set to be $\mathcal{A}$. In addition, as the computation proceeds, $\mathcal{Z}$ can invoke any ITI, by passing inputs to it, subject to the restriction that all these ITIs have session ID s and PID $\in \mathcal{P}$. The code of these ITIs is set to be π. Consequently, all the ITIs invoked by $\mathcal{Z}$, except for the adversaries, are parties in a single instance of π. Other than that, $\mathcal{Z}$ cannot pass inputs to any ITI other than the adversaries or the parties invoked by $\mathcal{Z}$, nor can any ITI other than these pass outputs to $\mathcal{Z}$.

Each adversary $\mathcal{A}_{(i,j)}$ is allowed to send messages to any ITI in the system with PID$= i$ where the sender identity of delivered messages must be PID$=j$. There need not be any correspondence between the messages sent by the parties and the messages delivered by the adversaries. The adversaries may not pass input to any party, nor can it pass output to any party other than $\mathcal{Z}$. It is important to notice that there is no direct communication between the adversaries and all their communication must go through the environment.

Adversaries may also *corrupt* parties. Corruption of a party (ITI) with identity id is modeled via a special `(Corrupt`, *id, p*`)` message delivered by $\mathcal{A}_{(i,j)}$ to that ITI, where p denotes potential additional parameters.

Any ITI other than $\mathcal{Z}$ and the adversaries, are allowed to pass inputs and outputs to any other ITI other than $\mathcal{Z}$ and the adversaries subject to the restriction that the recipient have the same PID as the sender. In addition, they can send messages to the adversaries where adversary's PID$_{(i,j)}$ requires sender's PID i and recipient's PID j. (These messages may indicate an identity of an intended recipient ITI; but the adversaries is not obliged to respect these indications.)

To summarize the above restrictions, for any ordered pair of PIDs (i, j), there is only one possible route for messages from the ITI with PID i to the ITI with PID j: a message m from the ITI with PID i is sent to the adversary $\mathcal{A}_{(i,j)}$, then it can only be outputted to $\mathcal{Z}$, then it is given to $\mathcal{A}_{(j,i)}$, then it is sent to the ITI with PID j.

The response of the party or sub-party to a `(Corrupt)` message is not defined in the general model; rather, it is left to the protocol. Here we specify one corruption model, namely that of Byzantine party corruption. We extend the known definition to fit multiple adversaries. Here, once a party or a sub-party receives a `(Corrupt)` message for the first time, it sends to that adversary its entire current local state. Also, in all future activations, a corrupted ITI merely forwards the incoming information to that adversary and follows instructions of all PID related adversaries.

All the restrictions above are enforced by the control function that is formally presented in [CV12]. Figure 1 presents a graphical depiction of the model.

Let LEXEC$_{\pi,\mathcal{A},\mathcal{Z}}$ denote the output distribution of $\mathcal{Z}$ in the execution above.

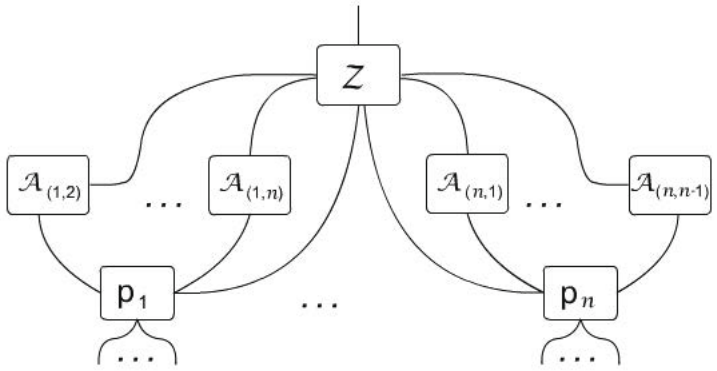

Fig. 1. The model of protocol execution. The environment $\mathcal{Z}$ writes the inputs and reads the subroutine outputs of the main parties running the protocol, while the adversaries, jointly with $\mathcal{Z}$, control the communication. In addition, $\mathcal{Z}$ may interact freely with all adversaries. The parties of π may have subroutines, to which $\mathcal{Z}$ has no direct access.

Now we present the general notion of emulating one protocol via another protocol. Informally, we say that a protocol π emulates protocol ϕ if no environment $\mathcal{Z}$ can tell whether it is participating in an execution of π or ϕ. That is, let A and B be binary distributions, then $A \approx B$ if the statistical distance between A and B is negligible.

Balanced Environments. In order to keep the notion of protocol emulation from being unnecessarily restrictive, we consider only environments where the amount of resources given to each adversary (namely, the length of the adversary's input) is at least some fixed polynomial fraction of the amount of resources given to the protocol. To be concrete, we consider only environments where, at any point in time during the execution, the overall length of the inputs given by $\mathcal{Z}$ to the parties of the main instance of π is at most k times the length of input to each adversary, where k is the security parameter in use. We call such environments Balanced environments.

Definition 1 (LUC-emulation). *Let π and ϕ be PPT protocols. We say that π LUC-emulates ϕ if for any PPT adversary $\mathcal{A}$ there exists an PPT adversary $\mathcal{S}$ such that for any balanced PPT environment $\mathcal{Z}$ we have:* $\text{LEXEC}_{\pi,\mathcal{A},\mathcal{Z}} \approx \text{LEXEC}_{\phi,\mathcal{S},\mathcal{Z}}$

If $\mathcal{F}$ is an ideal functionality we say that π LUC-realizes $\mathcal{F}$.

Hybrid Protocols. As in the UC framework, we define hybrid protocols to be protocols where, in addition to communicating via the adversary in the usual way, the parties also make calls to instances of ideal functionalities. In other words, an $\mathcal{F}$-hybrid protocol π, denoted by $\pi^{\mathcal{F}}$, is a protocol that includes subroutine calls to $\mathcal{F}$.

Theorem 3 (Universal composition, informal statement). *Let π, ρ, ϕ be PPT protocols. If ρ LUC-emulates ϕ then protocol π^ρ LUC-emulates protocol π^ϕ.*

The formal composition theorem and its proof can be found in [CV12].

3 Anonymity

The timing, existence and sender anonymity were informally presented in the introduction. Recall that in the introduction, these concerns are presented via devices such as dropbox, email future, and trusted coordinator; but in fact these are cryptographic channels guaranteeing anonymity in the subject matter. We present ideal functionalities, which are the formalizations of these channels, and realization by non-trivial protocols that do not provide anonymity. The functionalities are defined in LUC, and the corresponding UC (and CP) functionalities are defined by replacing the multiple adversarial interfaces with an equivalent single adversarial interface. We remark that in absence of any formal definition, we can only show that these protocols do not satisfy our intuitive perception of anonymity. Here, we present the inability of the UC and CP to capture the intuitive idea of anonymity.

3.1 Existence-Anonymity

Here our goal is to model a sender-receiver channel, denoted by existence-anonymous channel that has a strong anonymity guarantee regarding the existence of a sender. The existence-channel always allows the receiver to retrieve a message. However, in absence of a sender this message will be some randomly chosen message. The LUC existence-anonymous channel $\mathcal{F}_{\overline{\text{EA}}}$ is formally presented in Figure 2.

Functionality $\mathcal{F}^{D}_{\overline{\text{EA}}}$

Functionality $\mathcal{F}^{D}_{\overline{\text{EA}}}$ runs with parties S, R, adversaries $\mathcal{S}_{(S,R)}$, $\mathcal{S}_{(R,S)}$, and parametrized on message distribution D. It proceeds as follows:

- Upon receiving an input (Send, *sid*, m) from party S do: verify that $sid = (S, R, sid')$, else ignore the input. Next, record m, and send output (Send, *sid*, m) to $\mathcal{S}_{(S,R)}$. Ignore any subsequent (Send, ...) inputs. Once $\mathcal{S}_{(S,R)}$ allows to forward the message, mark m as approved.
- Upon receiving an input (Output, *sid*) from party R do:
  1. if there is an approved message m then set OUT $= m$; else set OUT $\leftarrow D$.
  2. send output (OUT, *sid*) to $\mathcal{S}_{(R,S)}$; Once $\mathcal{S}_{(R,S)}$ allows to forward the message output (OUT, *sid*) to R and halt.

Fig. 2. The existence-anonymous channel functionality $\mathcal{F}_{\overline{\text{EA}}}$

Functionality $\mathcal{F}_{\overline{\text{EB}}}$

Functionality $\mathcal{F}_{\overline{\text{EB}}}$ runs with parties S, R, and adversaries $\mathcal{S}_{(S,R)}$, $\mathcal{S}_{(R,S)}$. Initialize OUT $= \perp$ and proceed as follows:

- Upon receiving an input (Send, *sid*, m) from party S do: verify that $sid = (S, R, sid')$, else ignore the input. Next, record m, and send output (Send, *sid*, m) to $\mathcal{S}_{(S,R)}$. Ignore any subsequent (Send, ...) inputs. Once $\mathcal{S}_{(S,R)}$ allows to forward the message, mark m as approved.
- Upon receiving an input (Output, *sid*) from party R and there is an approved message m do:
  1. set OUT $= m$, and output (OUT, *sid*) to $\mathcal{S}_{(R,S)}$; Once $\mathcal{S}_{(R,S)}$ allows to forward the message output (OUT, *sid*) to R and halt.

Fig. 3. The basic existence-channel functionality $\mathcal{F}_{\overline{\text{EB}}}$

The underlying communication model is a channel called $\mathcal{F}_{\overline{\text{EB}}}$ that is similar to authentication channel with a difference in the message delivery. More specifically, a message is delivered to the recipient upon recipient's request and only if there exists a message sent to him. It is important to note that $\mathcal{F}_{\overline{\text{EB}}}$ does not provide existence-anonymity since the recipient is guaranteed that any message received was sent by the sender. The LUC channel $\mathcal{F}_{\overline{\text{EB}}}$ is formally presented in Figure 3.

Claim. The functionality $\mathcal{F}_{\overline{\text{EB}}}$ UC-realizes $\mathcal{F}_{\overline{\text{EA}}}$ and does not LUC-realize $\mathcal{F}_{\overline{\text{EA}}}$.

3.2 Timing-Anonymity

Here, our goal is to define a channel that guarantees to the sender that no receiver, upon receiving a message from him, can tell when this message was sent. As mentioned in the introduction, we define a timing-anonymous channel, denoted by $\mathcal{F}_{\overline{\text{TA}}}$, that randomly delay a message in a way that the amount of the delay is unknown to the receiver. In particular, the message is delivered only after a certain delay. The LUC channel $\mathcal{F}_{\overline{\text{TA}}}$ is formally presented in Figure 4.

Now we formally define the underlying model. In order to capture time, we introduce a clock functionality $\mathcal{F}_{\text{clock}}$ that is observable by all participants. This clock is not directly observed by the environment, instead, it is indirectly advanced by the environment, by instructing the sender to advance the clock; this captures a setting in which the receiver's future actions are not affected by the amount of the delay. The formal description of $\mathcal{F}_{\text{clock}}$ presented in Figure 5. The second component is the authentication functionality $\mathcal{F}_{\text{auth}}$. The difference between the LUC and the UC authentication functionality is that the LUC functionality, denoted by $\mathcal{F}_{\overline{\text{auth}}}$, operates not only when a message is sent. That is, it allows the adversary associated with the sender to approve delivery even when

Functionality $\mathcal{F}_{\overline{\text{TA}}}$

Functionality $\mathcal{F}_{\overline{\text{TA}}}$ runs with parties S, R, and adversaries $\mathcal{S}_{(S,R)}$, $\mathcal{S}_{(R,S)}$. Let $\mathcal{T}$ be some finite set of natural numbers. The functionality proceeds as follows:

- Upon receiving an input (Send, *sid*, m) from party S do: verify that $sid = (S, R, sid')$, else ignore the input. Next, choose uniformly at random $N \leftarrow \mathcal{T}$, set $k = N$, and record (k, m). Ignore any subsequent (Send, ...) inputs.
- Upon receiving an input (Advance, *sid*) from party S and $k > 0$ do:
  1. update $k = k - 1$ and output (Advance, *sid*) to adversary $\mathcal{S}_{(S,R)}$.
  2. if $k = 0$ output m to $\mathcal{S}_{(S,R)}$. Once $\mathcal{S}_{(S,R)}$ allows to forward the message output m to $\mathcal{S}_{(R,S)}$. Once also $\mathcal{S}_{(R,S)}$ allows to forward m output it to R.

1. Upon receiving (Corruptsend, *sid*, m') from $\mathcal{S}_{(S,R)}$, if S is corrupt and m has not been delivered to $\mathcal{S}_{(R,S)}$, then change the recorded message to m'.

Fig. 4. The timing-anonymous channel functionality $\mathcal{F}_{\overline{\text{TA}}}$

Functionality $\mathcal{F}_{\text{clock}}$

Functionality $\mathcal{F}_{\text{clock}}$ runs with parties S, R, and adversaries $\mathcal{S}_{(S,R)}$, $\mathcal{S}_{(R,S)}$. Initialize $T = 0$. Next:

- Upon receiving an input (Advance, *sid*) from party S do: in the first activation verify that $sid = (S, R, sid')$, else ignore the input. Next, set $T = T + 1$.
- Upon receiving an input (time, *sid*) from some party, output (time, *sid*, T).

Fig. 5. The clock functionality $\mathcal{F}_{\text{clock}}$

no message was sent; in this case $\mathcal{F}_{\overline{\text{auth}}}$ outputs $\perp$ to receiver's adversary and halts. We note that $\mathcal{F}_{\overline{\text{auth}}}$ seems as a natural relaxation of the UC authentication functionality. The LUC authentication functionality $\mathcal{F}_{\overline{\text{auth}}}$ is formally presented in Figure 6.

Claim. There exists a protocol π_{TA} that UC-realizes $\mathcal{F}_{\overline{\text{TA}}}$ and does not LUC-realize $\mathcal{F}_{\overline{\text{TA}}}$.

A protocol π_{TA} that UC-realizes $\mathcal{F}_{\overline{\text{TA}}}$ is:
Let $\mathcal{T}$ be some finite set of natural numbers.

1. INPUT: Having received input (Send, *sid*, m), S chooses uniformly at random $N \leftarrow \mathcal{T}$, set $k = N$, and records (k, m).
2. ADVANCE: Having received input $(\mathsf{Advance}, sid)$, S forward it to $\mathcal{F}_{\text{clock}}$ and updates $k = k - 1$. Once $k = 0$ send m to $\mathcal{F}_{\overline{\text{auth}}}$ and halt.
3. OUTPUT: Having received (Send, *sid*, m) from $\mathcal{F}_{\overline{\text{auth}}}$, the receiver R outputs m.

Functionality $\mathcal{F}_{\overline{\text{auth}}}$

Functionality $\mathcal{F}_{\overline{\text{auth}}}$ runs with parties S, R, and adversaries $\mathcal{S}_{(S,R)}$, $\mathcal{S}_{(R,S)}$. It proceeds as follows:

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

Fig. 6. The message authentication functionality $\mathcal{F}_{\overline{\text{auth}}}$

We note that π_{TA} does not provide timing anonymity since all participants in the protocol observe the clock. In particular, upon receiving a message, the receiver can retrieve the time by sending (time,*sid*) to $\mathcal{F}_{\text{clock}}$, and thus knows exactly when the message was sent.

3.3 Sender-Anonymity

The sender anonymity property is presented in the introduction via a trusted mediator that masks the identity of the sender. This mediator is similar to the two-anonymous channels of [NMO08]; however, their formalism is not applicable in our setting. The channel enables two senders and a *honest but curious* receiver to communicate anonymously in the following sense: both senders may send a message to the receiver but only one message is delivered. This sender-anonymous channel, denoted by $\mathcal{F}_{\overline{\text{SA}}}$, does not disclose the identity of the actual sender. The LUC formulation of $\mathcal{F}_{\overline{\text{SA}}}$ is presented in Figure 7.

Functionality $\mathcal{F}_{\overline{\text{SA}}}$

Functionality $\mathcal{F}_{\overline{\text{SA}}}$ running with parties S_1, S_2, R, and adversaries $\mathcal{S}_{(S_1,R)}$, $\mathcal{S}_{(S_2,R)}$, $\mathcal{S}_{(R,S_1)}$, $\mathcal{S}_{(R,S_2)}$. At first activation verify that $sid = (S_1, S_2, R, sid')$, else halt. Next, proceed as follows:

– Upon receiving an input (Send, *sid*, m_i) from party S_i do: record m_i and send output (Send, *sid*, m_i) to $\mathcal{S}_{(S_i,R)}$. Ignore any subsequent (Send,...) inputs from S_i. Once $\mathcal{S}_{(S_i,R)}$ allows to forward the message, output (Send, *sid*, m_i) to $\mathcal{S}_{(R,S_i)}$. Once approved, output (Send, *sid*, m_i) to R and halt.

Fig. 7. The sender-anonymous channel functionality $\mathcal{F}_{\overline{\text{SA}}}$

Functionality $\mathcal{F}_{\overline{\mathrm{S1}}}$

Functionality $\mathcal{F}_{\overline{\mathrm{S1}}}$ running with parties S_1, S_2, R, and adversaries $\mathcal{S}_{(S_1,R)}$, $\mathcal{S}_{(S_2,R)}$, $\mathcal{S}_{(R,S_1)}$, $\mathcal{S}_{(R,S_2)}$. Initialize variable BLOCK = 0. At first activation verify that $sid = (S_1, S_2, R, sid')$, else halt. Next, proceed as follows:

- Upon receiving an input (Send, *sid*, m_1) from party S_1 do: record m_1, and send output (Send, *sid*, m_1) to $\mathcal{S}_{(S_1,R)}$. Ignore any subsequent (Send, ...) inputs from S_1. Once $\mathcal{S}_{(S_1,R)}$ allows to forward the message output (Send, *sid*, m_1) to $\mathcal{S}_{(R,S_1)}$. Once approved, if BLOCK = 0 output (Send, *sid*, m_1) to R and halt; else halt.
- Upon receiving an input (Send, *sid*, m_2) from party S_2 do: record m_2, and send output (Send, *sid*, m_2) to $\mathcal{S}_{(S_2,R)}$. Ignore any subsequent (Send, ...) inputs from S_2. Once $\mathcal{S}_{(S_2,R)}$ allows to forward the message output (Send, *sid*, m_2) to $\mathcal{S}_{(R,S_2)}$. Once approved set BLOCK = 1.

Fig. 8. The basic sender-channel functionality $\mathcal{F}_{\overline{\mathrm{S1}}}$

The underlying communication channel, denoted by $\mathcal{F}_{\overline{\mathrm{S1}}}$, is a two-sender one receiver channel that delivers only messages sent by the first sender S1. We note that $\mathcal{F}_{\overline{\mathrm{S1}}}$ does not provide sender-anonymity. The formal description of $\mathcal{F}_{\overline{\mathrm{S1}}}$ is presented in Figure 8.

Claim. The functionality $\mathcal{F}_{\overline{\mathrm{S1}}}$ UC-realizes $\mathcal{F}_{\overline{\mathrm{SA}}}$ and does not LUC-realize $\mathcal{F}_{\overline{\mathrm{SA}}}$.

Sender-Anonymity in the CP Framework. We note that in the context of sender-anonymity, the CP model suffers from the same weakness as the UC model. That is, the above non sender-anonymous protocol is a CP-realization of the sender-anonymous channel $\mathcal{F}_{\overline{\mathrm{SA}}}$.

4 Bi-deniability

Here, we formalize a notion of bi-deniable authentication and show that UC security does not capture this flavor of deniability. In fact, this is true also for GUC. Moreover, we define bi-deniability separately and show equivalence between bi-deniable authentication and LUC secure authentication.

4.1 Bi-deniable Authentication

Bi-deniability aims to capture the ability of a participant in a two party protocol to deny participation in a protocol execution even if its communication had been externally exposed. The actual definition has some similarities to the definition presented in [DKSW09] (see details in the full version [CV12]).

The relevant entities are the following: we have a sender S who is potentially communicating with a receiver R, a judge $\mathcal{J}$ who will eventually rule whether or not the transmission was attempted, two informants $\mathcal{I}_S$, $\mathcal{I}_R$ who witness the communication (represented as log files owned by $\mathcal{I}_S$, $\mathcal{I}_R$) between S and R and are trying to convince the judge, and two misinformants $\mathcal{M}_S$, $\mathcal{M}_R$ who did not witness any communication but still want to convince the judge that one occurred.

The idea of the bi-deniability definition is that no party should be accused of participating in a protocol, if any evidence presented to the judge (by the informants) based on witnessing the protocol execution can be also presented (by the misinformants) without any communication whatsoever. This idea is formalized via indistinguishability of experiments as follows: Let π be some two-party protocol.

Informant-Experiment. The inputs to parties are given by the judge, and any output produced by the parties is given to the judge. S and R run π in the presence of the informants $\mathcal{I}_S$, $\mathcal{I}_R$. The informants report to $\mathcal{J}$ regarding any observed communication and execute all $\mathcal{J}$'s instruction. The output distribution of the judge $\mathcal{J}$ in this basic informant-experiment is denoted by $\mathrm{EXP}_{\pi,\mathcal{I}_S,\mathcal{I}_R,\mathcal{J}}$.

Misinformant-Experiment. The inputs to parties are also given to the misinformants. S and R do not communicate except with $\mathcal{M}_S$, $\mathcal{M}_R$. The misinformant $\mathcal{M}_S$ can send a single `(signal)` message[1] to $\mathcal{M}_R$; in addition, they can freely communicate with $\mathcal{J}$. Any message received by the parties from their misinformant is outputted to the judge. The output distribution of the judge $\mathcal{J}$ in this basic misinformant-experiment is denoted by $\mathrm{EXP}_{\mathcal{M}_S,\mathcal{M}_R,\mathcal{J}}$.

Definition 2 (Bi-deniability). *Let π be some PPT protocol and let the informants $\mathcal{I}_S, \mathcal{I}_R$ be as defined above. We say that π is bi-deniable if there exist PPT misinformants $\mathcal{M}_R$ and $\mathcal{M}_S$ such that for any PPT judge $\mathcal{J}$ we have:* $\mathrm{EXP}_{\pi,\mathcal{I}_S,\mathcal{I}_R,\mathcal{J}} \approx \mathrm{EXP}_{\mathcal{M}_S,\mathcal{M}_R,\mathcal{J}}$.

Theorem 4. *Let $\mathcal{F}_{\overline{\text{auth}}}$ be the LUC authentication functionality, and let $\mathcal{F}_{\text{auth}}$ be the UC authentication functionality of [Can01]. Then:*

1. $\mathcal{F}_{\overline{\text{auth}}}$ is bi-deniable.

2. Let π be some protocol. Then π LUC-realizes $\mathcal{F}_{\overline{\text{auth}}}$ if and only if π is bi-deniable.

3. $\mathcal{F}_{\text{auth}}$ is not bi-deniable.

5 Confinement

Recall that a protocol is said to enforce confinement if it prevents leakage of secret information to unauthorized processes in the network. This guarantee should hold even if all parties are faulty. In this section we present a definition of confinement and show that any LUC secure realization enforces confinement as long as the realized task does. In contrast to the other concerns, we will not show that UC security does not imply confinement but rather argue that

[1] Discussion regarding (signal) message appears in the full version [CV12].

any definition based on a centralized adversary does not enable proper separation between protocols that enforces confinement and the one that do not. In addition, we show that any UC functionality that enforces confinement is "super-ideal", in a well-defined sense.

5.1 Confinement with a Centralized Adversary

In the work of [HKN05] a definition of confinement is presented. Their definition considers the UC execution model with the following modifications: the UC environment is split into two environments $\mathcal{E}_{\mathcal{H}}$ and $\mathcal{E}_{\mathcal{L}}$, where $\mathcal{E}_{\mathcal{H}}$ interacts with the high-level processes and $\mathcal{E}_{\mathcal{L}}$ with the low-level processes. All processes have an I/O interface with the appropriate environment according to their classification. In addition, the high-level environment $\mathcal{E}_{\mathcal{H}}$ cannot give inputs either to the adversary or to the low-level environment $\mathcal{E}_{\mathcal{L}}$. [HKN05] define confinement as the following game: a random bit b is chosen by $\mathcal{E}_{\mathcal{H}}$, the parties run the protocol π, and eventually $\mathcal{E}_{\mathcal{L}}$ outputs its guess for b. We say that π enforce confinement for partition $\mathcal{H} : \mathcal{L}$ of the parties in π, if for any environments $\mathcal{E}_{\mathcal{H}}$, $\mathcal{E}_{\mathcal{L}}$ and adversary as above, $\mathcal{E}_{\mathcal{L}}$ succeeds in the confinement game with probability $\approx \frac{1}{2}$.

This definition enforces very strong requirements on the examined protocols[2], and as a consequence, many protocols that "obviously enforce confinement" do not satisfy this definition. We remark that this weakness is not unique to the [HKN05] definition, and any definition based on centralized adversary is subject to this weakness.

5.2 Definition of Confinement

Our definition follows the idea of [HKN05]. Like there, we consider split environments. More precisely, our definition consists of the same entities as in [HKN05] where the centralized adversary is split to multiple adversaries, an adversary for each pair of potentially communicating parties. We denote by $\mathcal{A}_{(i,j)}$ the adversary with identity $((i, j), \perp)$ and code $\mathcal{A}$.

The executed experiment for partition $\mathcal{H} : \mathcal{L}$ of the participating parties in π is the following: a random bit b is chosen by $\mathcal{E}_{\mathcal{H}}$, the parties run π, while the adversaries, jointly with $\mathcal{E}_{\mathcal{H}}$ and $\mathcal{E}_{\mathcal{L}}$ control the communication. The environments $\mathcal{E}_{\mathcal{H}}$ and $\mathcal{E}_{\mathcal{L}}$ write inputs and read the subroutine outputs of parties according to their classification. $\mathcal{E}_{\mathcal{L}}$ can also give inputs to $\mathcal{E}_{\mathcal{H}}$. In addition, $\mathcal{E}_{\mathcal{L}}$ can interact freely with all adversaries associated with $\mathcal{L}$, and all the adversaries associated with $\mathcal{H}$ can give outputs to $\mathcal{E}_{\mathcal{H}}$. The adversaries can communicate with the appropriate party and corrupt it. Eventually $\mathcal{E}_{\mathcal{L}}$ outputs its guess for b.

Let $\mathrm{CEXP}^{\mathcal{H}:\mathcal{L}}_{\pi,\mathcal{A},\mathcal{E}_{\mathcal{H}},\mathcal{E}_{\mathcal{L}}}$ denote the success indicator of $\mathcal{E}_{\mathcal{L}}$ in the above experiment.

Definition 3 (Confinement). *Let π be a PPT protocol and let $\mathcal{H} : \mathcal{L}$ be some partition of the parties in π. We say that π enforces $(\mathcal{H} : \mathcal{L})$-confinement if for*

[2] Shown in the full version [CV12].

any PPT adversary $\mathcal{A}$ and for any balanced PPT environments $\mathcal{E}_{\mathcal{H}}$ and $\mathcal{E}_{\mathcal{L}}$ we have: $\mathrm{CEXP}^{\mathcal{H}:\mathcal{L}}_{\pi,\mathcal{A},\mathcal{E}_{\mathcal{H}},\mathcal{E}_{\mathcal{L}}} \approx U_1$*, where U_1 is the uniform distribution over* $\{0,1\}$.

Theorem 5. *Let π, ϕ be protocols such that π LUC emulates ϕ. Then π enforce $(\mathcal{H} : \mathcal{L})$-confinement for all partitions $\mathcal{H} : \mathcal{L}$ of the parties in π for which ϕ enforce $(\mathcal{H} : \mathcal{L})$-confinement.*

5.3 Confinement with Respect to Super-Ideal Functionalities

Here, we show that any UC functionality that enforces confinement must be "super-ideal". That is, such functionalities do not provide the adversary with any information, even when a party is corrupted. We call such functionalities super-ideal since such functionalities essentially mandate communication channels which offer absolute physical security that hides even whether communication took place at all.

Definition 4 (Super-ideal, informal statement). *Let $\mathcal{F}$ be a n-party functionality and let $\mathcal{H} : \mathcal{L}$ be some partition of the parties. Then $\mathcal{F}$ is super-ideal with respect to a set of identities $\mathcal{H}$ if for any adversary associated with a party P_i for $i \in \mathcal{H}$ and for any two possible inputs the following holds: the adversary cannot tell, even with the assistance of the adversarial interface of $\mathcal{F}$, which one of the inputs was used by party P_i.*

Claim. Let $\mathcal{F}$ be a UC functionality and let $\mathcal{H} : \mathcal{L}$ be some partition for which $\mathcal{F}$ enforces $(\mathcal{H} : \mathcal{L})$-confinement. Then, $\mathcal{F}$ is super-ideal with respect to all parties in $\mathcal{H}$.

Acknowledgments. We thank Noam Livne, Vassilis Zikas, Ilia Gorelik, and Daniel Shahaf for helpful discussions.

References

[AKL$^+$09] Alwen, J., Katz, J., Lindell, Y., Persiano, G., Shelat, A., Visconti, I.: Collusion-Free Multiparty Computation in the Mediated Model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 524–540. Springer, Heidelberg (2009)

[AKMZ12] Alwen, J., Katz, J., Maurer, U., Zikas, V.: Collusion-Preserving Computation. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 124–143. Springer, Heidelberg (2012), `http://eprint.iacr.org/2011/433.pdf`

[ASV08] Alwen, J., Shelat, A., Visconti, I.: Collusion-Free Protocols in the Mediated Model. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 497–514. Springer, Heidelberg (2008)

[CL05] Camenisch, J., Lysyanskaya, A.: A Formal Treatment of Onion Routing. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 169–187. Springer, Heidelberg (2005)

[Can00] Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptology 13(1), 143–202 (2000)

[Can01] Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: 42nd FOCS (2001); revised version (2005), `eprint.iacr.org/2000/067`

[CDPW07] Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally Composable Security with Global Setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)

[CLOS02] Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th STOC (2002)

[CV12] Canetti, R., Vald, M.: Universally Composable Security With Local Adversaries. IACR Eprint (2012)

[DM00] Dodis, Y., Micali, S.: Parallel Reducibility for Information-Theoretically Secure Computation. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 74–92. Springer, Heidelberg (2000)

[DKSW09] Dodis, Y., Katz, J., Smith, A., Walfish, S.: Composability and On-Line Deniability of Authentication. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 146–162. Springer, Heidelberg (2009)

[GL90] Goldwasser, S., Levin, L.: Fair Computation of General Functions in Presence of Immoral Majority. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 77–93. Springer, Heidelberg (1991)

[GM84] Goldwasser, S., Micali, S.: Probabilistic encryption. JCSS 28(2), 270–299 (1984)

[GMR85] Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. SIAM J. Comput. 18, 186–208 (1989); (also in STOC 1985, pp. 291-304)

[GMW87] Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game. In: 19th Symposium on Theory of Computing (STOC), pp. 218–229. ACM (1987)

[Gol04] Goldreich, O.: Foundations of Cryptography, vol. 2: Basic Applications. Cambridge University Press, Cambridge (2004)

[HKN05] Halevi, S., Karger, P.A., Naor, D.: Enforcing confinement in distributed storage and a cryptographic model for access control. Cryptology Eprint Archive Report 2005/169 (2005)

[ILM05] Izmalkov, S., Lepinski, M., Micali, S.: Rational Secure Computation and Ideal Mechanism Design. In: FOCS 2005: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, pp. 585–595. IEEE Computer Society, Washington, DC (2005)

[ILM08] Izmalkov, S., Lepinski, M., Micali, S.: Verifiably Secure Devices. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 273–301. Springer, Heidelberg (2008)

[ILM11] Izmalkov, S., Lepinski, M., Micali, S.: Perfect implementation. Games and Economic Behavior 71(1), 121–140 (2011)

[Lam73] Lampson, B.W.: A note on the confinement problem. Communications of the ACM 16(10), 613–615 (1973)

[LMS05] Lepinksi, M., Micali, S., Shelat, A.: Collusion-Free Protocols. In: STOC 2005: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, pp. 543–552. ACM, New York (2005)

[MR11] Maurer, U., Renner, R.: Abstract cryptography. In: Innovations in Computer Science. Tsinghua University Press (2011)

[MR91] Micali, S., Rogaway, P.: Secure Computation. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 392–404. Springer, Heidelberg (1992)

[NMO08] Nagao, W., Manabe, Y., Okamoto, T.: Relationship of Three Cryptographic Channels in the UC Framework. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 268–282. Springer, Heidelberg (2008)

[OPW11] O'Neill, A., Peikert, C., Waters, B.: Bi-Deniable Public-Key Encryption. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 525–542. Springer, Heidelberg (2011)

[PS04] Prabhakaran, M., Sahai, A.: New notions of security: achieving universal composability without trusted setup. In: 36th STOC, pp. 242–251 (2004)

[PW00] Pfitzmann, B., Waidner, M.: Composition and integrity preservation of secure reactive systems. In: 7th ACM Conf. on Computer and Communication Security, pp. 245–254 (2000)