Universally Composable Security with Global Setup

Ran Canetti[1], Yevgeniy Dodis[2], Rafael Pass[3], and Shabsi Walfish[2]

[1] canetti@csail.mit.edu IBM Research
[2] {dodis,walfish}@cs.nyu.edu New York University
[3] rafael@cs.cornell.edu Cornell University

Abstract. Cryptographic protocols are often designed and analyzed under some *trusted set-up* assumptions, namely in settings where the participants have access to global information that is trusted to have some basic security properties. However, current modeling of security in the presence of such set-up falls short of providing the expected security guarantees. A quintessential example of this phenomenon is the *deniability* concern: there exist natural protocols that meet the strongest known composable security notions, and are still vulnerable to bad interactions with rogue protocols that use the same set-up.

We extend the notion of universally composable (UC) security in a way that re-establishes its original intuitive guarantee even for protocols that use globally available set-up. The new formulation prevents bad interactions even with adaptively chosen protocols that use the same set-up. In particular, it guarantees deniability. While for protocols that use no set-up the proposed requirements are the same as in traditional UC security, for protocols that use global set-up the proposed requirements are significantly stronger. In fact, realizing Zero Knowledge or commitment becomes provably impossible, even in the Common Reference String model. Still, we propose reasonable alternative set-up assumptions and protocols that allow realizing practically any cryptographic task under standard hardness assumptions *even against adaptive corruptions.*

1 Introduction

The trusted party paradigm is a fundamental methodology for defining security of cryptographic protocols. The basic idea (which originates in [24]) is to say that a protocol securely realizes a given computational task if running the protocol amounts to "emulating" an ideal process where all parties secretly hand their inputs to an imaginary "trusted party" who locally computes the desired outputs and hands them back to the parties. One potential advantage of this paradigm is its strong "built in composability" property: The fact that a protocol π emulates a certain trusted party $\mathcal{F}$ can be naturally interpreted as implying that any system that includes calls to protocol π should, in principle, behave the same if the calls to π were replaced by ideal calls to the trusted party $\mathcal{F}$.

Several formalizations of the above intuitive idea exist, e.g. [23, 27, 3, 9, 20, 31, 10, 30]. These formalizations vary in their rigor, expressibility, generality and restrictiveness, as well as security and composability guarantees. However, one point which no existing formalism seems to handle in a fully satisfactory way is the security requirements in the presence of "global trusted setup assumptions", such as a public-key infrastructure (PKI) or a common reference string (CRS), where all parties are assumed to have access to some global information that is trusted to have certain properties. Indeed, as pointed out in [28], the intuitive guarantee that "running π has the same effect as having access to the trusted party" no longer holds.

As a first indication of this fact, consider the "deniability" concern, namely, allowing party A to interact with party B in a way that prevents B from later "convincing" a third party C that the interaction took place. Indeed, if A and B interact via an idealized "trusted party" that communicates only with A and B then deniability is guaranteed in a perfect, idealized way. Thus, intuitively, if A and B interact via a protocol that emulates the trusted party, then deniability should hold just the same. When the protocol in question uses no global setup, this intuition works, in the sense that emulating a trusted party (in most existing formalisms) automatically implies deniability. However, when global setup is used, this is no longer the case: There are protocols that emulate such a trusted party but do *not* guarantee deniability.

For instance, consider the case of Zero-Knowledge protocols, *i.e.* protocols that emulate the trusted party for the "Zero-Knowledge functionality": Zero-Knowledge protocols in the plain model are inherently deniable, but most Zero-Knowledge protocols in the CRS model are completely *un*deniable whenever the reference string is public knowledge (see [28]). Similarly, most authentication protocols (i.e., most protocols that emulate the trusted party that provides ideally authenticated communication) that use public key infrastructure are not deniable, in spite of the fact that ideal authenticated communication via a trusted party is deniable.

One might think that this "lack of deniability" arises only when the composability guarantees provided by the security model are weak. However, even very strong notions of composability do not automatically suffice to ensure deniability in the presence of global setup. For example, consider the Universal Composability (UC) security model of [10], which aims to achieve the following, very strong composability guarantee:

> *A UC-secure protocol π implementing a trusted party $\mathcal{F}$ does not affect any other protocols more than $\mathcal{F}$ does — even when protocols running concurrently with π are maliciously constructed.*

When $\mathcal{F}$ is the Zero-Knowledge functionality, this property would seem to guarantee that deniability will hold even when the protocol π is used in an arbitrary manner. Yet, even UC-secure ZK protocols that use a CRS are *not* deniable whenever the reference string is globally available. This demonstrates that the UC notion, in its present formulation, does *not* protect a secure protocol π from a protocol π' that was maliciously designed to interact badly with π, in the case where π' can use the *same setup* as π.

Deniability is not the only concern that remains un-captured in the present formulation of security in the CRS model. For instance, even UC-secure Zero-

Knowledge proofs in the CRS model may not be "adaptively sound" (see [22]), so perhaps a malicious prover can succeed in proving false statements after seeing the CRS, as demonstrated in [1]. As another example, the protocol in [15] for realizing the single-instance commitment functionality becomes *malleable* as soon as *two* instances use the same reference string (indeed, to avoid this weakness a more involved protocol was developed, where multiple commitments can explicitly use the same reference string in a specific way). Note that here, a UC-secure protocol can even affect the security of another UC-secure protocol if both protocols make reference to the same setup.

This situation is disturbing, especially in light of the fact that *some* form of setup is often *essential* for cryptographic solutions. For instance, most traditional two-party tasks cannot be UC-realized with no setup [15, 10, 16], and authenticated communication is impossible without some sort of setup [12]. Furthermore, providing a *globally available* setup that can be used throughout the system is by far the most realistic and convenient way to provide setup.

A new formalism. This work addresses the question of how to formalize the trusted-party definitional paradigm in a way that preserves its intuitive appeal even for those protocols that use globally available setup. Specifically, our first contribution is to generalize the UC framework to deal with global setup, so as to explicitly guarantee that the original meaning of "emulating a trusted party" is preserved, even when the analyzed protocol is using the *same setup* as other protocols that may be maliciously and adaptively designed to interact badly with it. In particular, the new formalism called simply generalized UC (GUC) security guarantees deniability and non-malleability even in the presence of global setup. Informally,

> *A GUC-Secure protocol π implementing a trusted party $\mathcal{F}$ using some global setup does not affect any other protocols more than $\mathcal{F}$ does — even when protocols running concurrently with π are maliciously constructed, and even when all protocols use the same global setup.*

In a nutshell, the new modeling proceeds as follows. Recall that the UC framework models setup as a "trusted subroutine" of the protocol that uses the setup. This implicitly means that the setup is local to the protocol instance using it, and cannot be safely used by any other protocol instance. That modeling, while mathematically sound, certainly does not capture the real-world phenomenon of setup that is set in advance and publicly known throughout the system. The UC with joint state theorem ("JUC Theorem") of [18] allows several instances of specifically-designed protocols to use the same setup, but it too does not capture the case of public setup that can be used by arbitrary different protocols at the same time.

To adequately capture global setup our new formalism models the setup as an additional (trusted) entity that interacts not only with the parties running the protocol, but also with other parties (or, in other words, with the external environment). This in particular means that the setup entity exists not only as part of the protocol execution, but also in the *ideal process,* where the protocol is replaced by the trusted party. For instance, while in the current UC framework the CRS model is captured as a trusted setup entity that gives the reference string only to the adversary and the parties running the actual protocol instance, here the reference string is globally available, i.e. the trusted setup entity also gives the reference string directly to other parties and the external environment. Technically, the effect of this modeling is that now the simulator (namely, the adversary in the ideal process) cannot choose the reference string or know related trapdoor information.

In a way, proofs of security in the new modeling, even with setup, are reminiscent of the proofs of security without setup, in the sense that the only freedom enjoyed by the simulator is to control the local random choices of the uncorrupted parties. For this reason we often informally say that GUC-secure protocols that use only globally available setup are "fully simulatable". We also remark that this modeling is in line with the "non-programmable CRS model" in [28].

One might thus suspect that achieving GUC-security "collapses" down to UC-security *without any setup* (and its severe limitations). Indeed, as a first result we extend the argument of [15] to show that no two-party protocol can GUC-realize the ideal commitment functionality $\mathcal{F}_{com}$ (namely, emulate the trusted party that runs the code of $\mathcal{F}_{com}$ according to the new notion), *even in the CRS model, or in fact with any global setup that simply provides public information.* On the one hand this result is reassuring, since it means that those deniable and malleable protocols that are secure in the (old) CRS model can no longer be secure according to the new notion. On the other hand, this result brings forth the question of whether there exist protocols for commitment (or other interesting primitives) that meet the new notion under *any* reasonable setup assumption. Indeed, the analyses of all existing UC-secure commitment protocols seem to use in an essential way the fact that the simulator has control over the value of the setup information.

New setup and constructions. Perhaps surprisingly, we answer the realizability question in the affirmative, in a strong sense. Recall that our impossibility result shows that a GUC protocol for the commitment functionality must rely on a setup that provides the parties with some *private* information. We consider two alternative setup models which provide such private information in a *minimal* way, and show how to GUC-realize practically any ideal functionality in any one of the two models.

The first setup model is reminiscent of the "key registration with knowledge (KRK)" setup from [5], where each party registers a public key with some trusted authority in a way that guarantees that the party can access the corresponding secret key. However, in contrast to [5] where the scope of a registered key is only a single protocol instance (or, alternatively, several instances of specifically designed protocols), here the registration is done once per party throughout the lifetime of the system, and the public key can be used in all instances of all the protocols that the party might run. In particular, it is directly accessible by the external environment.

We first observe that one of the [5] protocols for realizing $\mathcal{F}_{com}$ in the KRK model can be shown to satisfy the new notion, even with the global KRK setup, as long as the adversary is limited to *non-adaptive* party corruptions. (As demonstrated in [17], realizing $\mathcal{F}_{com}$ suffices for realizing *any* "well-formed" multi-party functionality.) However, when adaptive party corruptions are allowed, and the adversary can observe the past internal data of corrupted parties, this protocol becomes insecure. In fact, the problem seems inherent, since the adversary is now able to eventually see *all* the secret keys in the system, even those of parties that were uncorrupted when the computation took place.

Still, we devise a new protocol that realizes $\mathcal{F}_{com}$ in the KRK model even in the presence of adaptive party corruptions, and without any need for data erasures. The high level idea is to use the [15] commitment scheme with a new CRS that is chosen by the parties per commitment. The protocol for choosing the CRS will make use of the public keys held by the parties, in a way that allows the overall simulation to go through even when the same public keys are used in multiple instances of the CRS-generation protocol. Interestingly, our construction does not realize a CRS that is "strong" enough for the original analysis to go through. Instead, we provide a "weaker" CRS, and provide a significantly more elaborate analysis. The protocol is similar in spirit to the coin-tossing protocol of [19], in that it allows the generated random string to have different properties depending on which parties are corrupted. Even so, their protocol is not adaptively secure in our model.

Augmented CRS. Next we formulate a new setup assumption, called "augmented CRS (ACRS)" and demonstrate how to GUC-realize $\mathcal{F}_{com}$ in the ACRS model, in the presence of adaptive adversaries. As the name suggests, ACRS is reminiscent of the CRS setup, but is somewhat augmented so as to circumvent the impossibility result for plain CRS. That is, as in the CRS setup, all parties have access to a short reference string that is taken from a pre-determined distribution. In addition, the ACRS setup allows corrupted parties to obtain "personalized" secret keys that are derived from the reference string, their public identities, and some "global secret" that's related to the public string and remains unknown. It is stressed that *only corrupted parties* may obtain their secret keys. This means that the protocol may not include instructions that require knowledge of the secret keys and, therefore, the protocol interface tn the ACRS setup is identical to that of the CRS setup.

The main tool in our protocol for realizing $\mathcal{F}_{com}$ in the ACRS model is a new *identity-based trapdoor commitment (IBTC)* protocol. IBTC protocols are constructed in [2, 32], in the Random Oracle model. In the full version of this paper [13], we provide a construction of IBTC in the standard model (assuming only one-way functions), using the Σ-protocol based commitment technique of Feige [21], where the committer runs the *simulator* of the Σ-protocol.

Realizing the setup assumptions. "Real world implementations" of the ACRS and KRK setups can involve a trusted entity (say, a "post office") that only publicizes the public value. The trusted entity will also agree to provide the secret keys to the corresponding parties upon request, with the understanding that once a party gets hold of its key then it alone is responsible to safeguard it and use it appropriately (much as in the case of standard PKI). In light of the impossibility of a completely non-interactive setup (CRS), this seems to be a minimal "interactiveness" requirement from the trusted entity.

Another unique feature of our commitment protocol is that it guarantees security even if the "global secret" is compromised, as long as this happens *after the commitment phase is completed.* In other words, in order to compromise the overall security, the trusted party has to be *actively malicious during the commitment phase.* This point further reduces the trust in the real-world entity that provides the setup.

Despite the fact that the trusted entity need not be constantly available, and need not remain trustworthy in the long term, it may still seem difficult to provide such an interactive entity in many real-world settings. Although it is impossible to achieve true GUC security with a mere CRS, we observe that the protocols analyzed here do satisfy some notion of security even if the setup entity remains non-interactive (*i.e.* when our ACRS setup functionality is instead collapsed to a standard CRS setup). In fact, although we do not formally prove a separation, protocols proven secure in the ACRS model seem intuitively *more secure* than those of [15, 17] *even when used in the CRS model*! Essentially, in order to simulate information that could be obtained via a real attack on the protocols of [15, 17], knowledge of a "global trapdoor" is required. This knowledge enables the simulator to break the security *of all parties* (including their privacy). On the other hand, simulating the information obtained by real attacks on protocols that are proven secure in the ACRS model merely requires some specific "identity-based trapdoors". These specific trapdoors used by the simulate allow it to break only the security *of corrupt parties who deviate from the protocol*. Of course, when using a CRS setup in "real life" none of these trapdoors are available to anyone, so one cannot actually simulate information obtained by an attacker. Nevertheless, it seems that the actual advantage gained by an attack which *could* have been simulated using the more minimal resources required by protocol simulators in the ACRS model (*i.e.* the ability to violate the security only of corrupt parties, as opposed to all parties) is intuitively smaller.

A New Composition Theorem. We present two formulations of GUC security: one formulation is more general and more "intuitively adequate", while the other is simpler and easier to work with. In particular, while the general notion directly considers a multi-instance system, the simpler formulation (called EUC) is closer to the original UC notion that considers only a single protocol instance in isolation. We then demonstrate that the two formulations are equivalent. As may be expected, the proof of equivalence incorporates much of the argumentation involved in the proof of the universal composition theorem. We also demonstrate that GUC security is preserved under universal composition.

Related work. Relaxed variants of UC security are studied in [30, 8]. These variants allow reproducing the general feasibility results without setup assumptions other than authenticated communication. However, these results provide significantly weaker security properties than UC-security. In particular, they do not guarantee security in the presence of arbitrary other protocols, which is the focus of this work.

Alternatives to the CRS setup are studied in [5]. As mentioned above, the KRK setup used here is based on the one there, and the protocol for GUC-realizing $\mathcal{F}_{com}$ for non-adaptive corruptions is taken from there. Furthermore, [5] informally discuss the deniability properties of their protocol. However, that work does not address the general concern of guaranteeing security in the presence of global setup. In particular, it adopts the original UC modeling of setup as a construct that is internal to each protocol instance.

In a concurrent work, Hofheinz et. al [25] consider a notion of security reminiscent of EUC, with similar motivation to the motivation here. They also formulate a new setup assumption and show how to realize any functionality given that setup. However, their setup assumption is considerably more involved than ours, since it requires the trusted entity to interact with the protocol in an on-line, input-dependent manner. Also, they do not consider adaptive corruptions.

Future work. This work develops the foundations necessary for analyzing security and composability of protocols that use globally available setup. It also re-establishes the feasibility results for general computation in this setting. Still, there are several unexplored research questions here.

One important concern is that of guaranteeing *authenticated communication* in the presence of global PKI setup. As mentioned above, this is another example where the existing notions do not provide the expected security properties (e.g., they do not guarantee deniability, whereas the trusted party solution is expressly deniable). We conjecture that GUC authentication protocols (namely, protocols that GUC-realize ideally authentic communication channels) that use a global PKI setup can be constructed by combining the techniques of [26, 15]. However, we leave full exploration of this problem out of scope for this work.

The notions of key exchange and secure sessions in the presence of global PKI setup need to be re-visited in a similar way. How can universal composability (and, in particular, deniability) be guaranteed for such protocols? Also, how can existing protocols (that are not deniable) be proven secure with globally available setup?

2 Generalized UC Security

In this section we will provide a high-level overview of our new Generalized UC (GUC) framework, as well as a useful simplification of GUC called the Externalized UC (EUC) framework. We begin with a brief review of the concepts behind the original UC framework of [10] (henceforth referred to as "Basic UC") before proceeding to outline our new security frameworks. To keep our discussion at a high level of generality, we will focus on the notion of protocol "emulation", wherein the objective of a protocol π is to emulate another protocol ϕ. Here, typically, π is an implementation (such as the actual "real world" protocol) and ϕ is a specification (where the "ideal functionality" $\mathcal{F}$ that we wish to implement is computed directly by a trusted entity). Throughout our discussion, all entities and protocols we consider are "efficient" (*i.e.* polynomial time bounded Interactive Turing Machines, in the sense detailed in [11]).

The Basic UC Framework. At a very high level, the intuition behind security in the basic UC framework is that any adversary $\mathcal{A}$ attacking a protocol π should learn no more information than could have been obtained via the use of a simulator $\mathcal{S}$ attacking protocol ϕ. Furthermore, we would like this guarantee to be maintained even if ϕ were to be used a subroutine of (*i.e.* composed with) arbitrary other protocols that may be running concurrently in the networked environment, and we plan to substitute π for ϕ in all instances. Thus, we may set forth a challenge experiment to distinguish between actual attacks on protocol π, and simulated attacks on protocol ϕ (referring to these protocols as the "challenge protocols"). As part of this challenge scenario, we will allow adversarial attacks to be orchestrated and monitored by a distinguishing environment $\mathcal{Z}$ that is also empowered to control the inputs supplied to the parties running the challenge protocol, as well as to observe the parties' outputs at all stages of the protocol execution. One may imagine that this environment represents all other activity in the system, including the actions of other protocol sessions that may influence inputs to the challenge protocol (and which may, in turn, be influenced by the behavior of the challenge protocol). Ultimately, at the conclusion of the challenge, the environment $\mathcal{Z}$ will be tasked to distinguish between adversarial attacks perpetrated by $\mathcal{A}$ on the challenge protocol π, and attack simulations conducted by $\mathcal{S}$ with protocol ϕ as the challenge protocol instead. If no environment can successfully distinguish these two possible scenarios, then protocol π is said to "UC emulate" the protocol ϕ.

Specifying the precise capabilities of the distinguishing environment $\mathcal{Z}$ is crucial to the meaning of this security notion. We must allow $\mathcal{Z}$ to choose the challenge protocol inputs and observe its outputs (which models the influence of the environment on the users of the protocol, and vice versa). We must also grant $\mathcal{Z}$ the ability to interact with the attacker (which will be either the adversary, or a simulation). As demonstrated in [10], granting precisely these capabilities to $\mathcal{Z}$ (even if we allow it to invoke only a *single session* of the challenge protocol) is sufficient to achieve the strong guarantees of *composition theorem*, which states that any arbitrary instances of the ϕ that may be running in the network can be safely substituted with a protocol π that UC emulates ϕ. Thus, even if we *constrain* the distinguisher $\mathcal{Z}$ to interactions only with the adversary and a *single session* of the challenge protocol (without allowing $\mathcal{Z}$ to invoke other protocols at all), we can already achieve the strong security guarantees we intuitively desired. Notably, although the challenge protocol may invoke subroutines of its own, it was not necessary to grant $\mathcal{Z}$ any capability to interact with such subroutines.

In order to conceptually modularize the design of protocols, the notion of "hybrid models" is often introduced into the basic UC framework. A protocol π is said to be realized "in the $\mathcal{G}$-hybrid model" if π invokes the ideal functionality $\mathcal{G}$ as a subroutine (perhaps multiple times). (As we will soon see below, the notion of hybrid models greatly simplifies the discussion of UC secure protocols that require "setup".) A high-level conceptual view of UC protocol emulation in a hybrid model is shown in Figure 1.

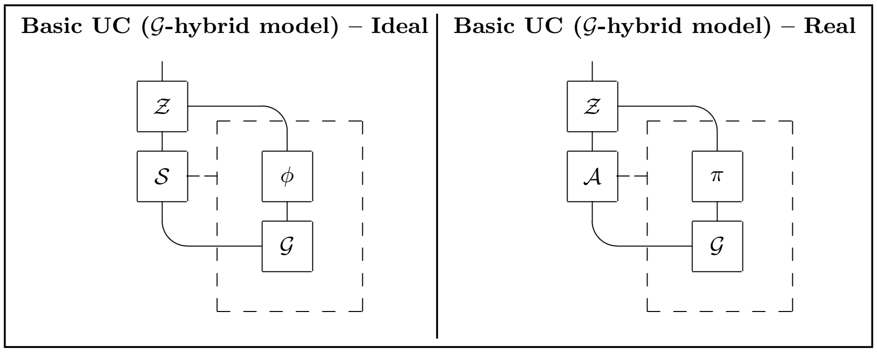

Fig. 1. The Basic UC Experiment in the $\mathcal{G}$-hybrid model. A simulator $\mathcal{S}$ attacks a single session of protocol ϕ running with an ideal subroutine $\mathcal{G}$, whereas an arbitrary "real" adversary $\mathcal{A}$ attacks a session of π running with an ideal subroutine $\mathcal{G}$. The dashed box encloses protocols where $\mathcal{S}$ or $\mathcal{A}$ control the network communications, whereas the solid lines represent a direct Input/Output relationship. (In a typical scenario, ϕ would be the ideal protocol for a desired functionality $\mathcal{F}$, whereas π would be a practical protocol realizing $\mathcal{F}$, with $\mathcal{G}$ modeling some "setup" functionality required by π. Observe that the environment can never interact directly with $\mathcal{G}$, and thus, in this particular scenario, $\mathcal{G}$ is never invoked at all in the ideal world since we are typically interested in the case where ideal protocol for $\mathcal{F}$ does not make use of $\mathcal{G}$.)

Limitations of Basic UC. Buried inside the intuition behind the basic UC framework is the critical notion that the environment $\mathcal{Z}$ is capable of utilizing its input/output interface to the challenge protocol to mimic the behavior of other (arbitrary) protocol sessions that may be running in a computer network. Indeed, as per the result of [10] mentioned in our discussion above, this would seem to be the case when considering challenge protocols that are essentially "self-contained". Such self-contained protocols, which do not make use of any "subroutines" (such as ideal functionalities) belonging to other protocol sessions, are called *subroutine respecting* protocols – and the basic UC framework models these protocols directly. On the other hand, special considerations would arise if the challenge protocol utilizes (or transmits) information that is also shared by other network protocol sessions. An example of such information would be the use of a global setup, such as a public "common reference string" (CRS) that is reused from one protocol session to the next, or a standard Public Key Infrastructure (PKI). Such shared state is not directly modeled by the basic UC framework discussed above. In fact, the composition theorem of [10] only holds when considering instances of subroutine respecting protocols (which *do not* share state information). Unfortunately, it is impossible to produce UC secure realizations of most useful functionalities without resorting to some setup. However, to comply with the requirements of the UC framework, the setup would have to be done on a per-instance basis. This does not faithfully represent the common realization, where the same setup is shared by all instances. Therefore, previous works handled such "shared state" protocol design situations via a special proof technique, known as the JUC Theorem [18].

Yet, even the JUC Theorem does not accurately model truly *global* shared state information. JUC Theorem only allows for the construction of protocols that share state *amongst themselves*. That is, an *a-priori* fixed set of protocols can be proven secure if they share state information *only* with each other. No security guarantee is provided in the event that the shared state information is also used by other protocols which the original protocols were not specifically designed to interact with. Of course, malicious entities may take advantage of this by introducing new protocols that use the shared state information if the shared state is publicly available. In particular, protocols sharing global state (*i.e.* using global setups) which are modeled in this fashion may not resist adaptive chosen protocol attacks, and can suffer from a lack of deniability, as we previously mentioned regarding the protocols of [15], [17], and as is discussed in further detail in Section 3.2.

The Generalized UC Framework. To summarize the preceding discussion, the environment $\mathcal{Z}$ in the basic UC experiment is unable to invoke protocols that share state in any way with the challenge protocol. This limitation is unrealistic in the case of global setup, when protocols share state information with each other (and indeed, it was shown to be impossible to realize UC-secure protocols without resort to such tactics [15, 10, 16]). To overcome this limitation, we propose the Generalized UC (GUC) framework. The GUC challenge experiment is similar to the basic UC experiment, only with an *unconstrained* environment. In particular, we will allow $\mathcal{Z}$ to actually invoke and interact with arbitrary protocols, and even multiple sessions of its challenge protocol (which may be useful to $\mathcal{Z}$ in its efforts to distinguish between the two possible challenge protocols). Some of the protocol sessions invoked by $\mathcal{Z}$ may share state information with challenge protocol sessions, and indeed, they can provide $\mathcal{Z}$ with information about the challenge protocol that it could not have obtained otherwise. The only remaining limitation on $\mathcal{Z}$ is that we prevent it from directly observing or influencing the network communications of the challenge protocol sessions, but this is naturally the job of the adversary (which $\mathcal{Z}$ directs). Thus, the GUC experiment allows a very powerful distinguishing environment capable of truly capturing the behavior of arbitrary protocol interactions in the network, *even if protocols can share state information with arbitrary other protocols.* Of course, protocols that are GUC secure are also composable (this fact follows almost trivially from a greatly simplified version of the composition theorem proof of [11], the simplifications being due to the ability of the unconstrained environment to directly invoke other protocol sessions rather than needing to "simulate" them internally).

The Externalized UC Framework. Unfortunately, since the setting of GUC is so complex, it becomes extremely difficult to prove security of protocols in our new GUC framework. Essentially, the distinguishing environment $\mathcal{Z}$ is granted a great deal of freedom in its choice of attacks, and any proof of protocol emulation in the GUC framework must hold even in the presence of other arbitrary protocols running concurrently. To simplify matters, we observe that in practice protocols which are designed to share state do so only in a very limited fashion (such as via a single common reference string, or a PKI, etc.). In particular, we will model shared state information via the use of "shared functionalities", which are simply functionalities that may interact with more than one protocol session (such as the CRS functionality). For clarity, we will distinguish the notation for shared functionalities by adding a bar (*i.e.* we use $\bar{\mathcal{G}}$ to denote a shared functionality). We call a protocol π that *only* shares state information via a single shared functionality $\bar{\mathcal{G}}$ a *$\bar{\mathcal{G}}$-subroutine respecting* protocol. Bearing in mind that it is generally possible to model "reasonable" protocols that share state information as $\bar{\mathcal{G}}$-subroutine respecting protocols, we can make the task of proving GUC security simpler by considering a compromise between the constrained environment of basic UC and the unconstrained environment of GUC. An *$\bar{\mathcal{G}}$-externally constrained* environment is subject to the same constraints as the environment in the basic UC framework, only it is additionally allowed to invoke a single "external" protocol (specifically, the protocol for the shared functionality $\bar{\mathcal{G}}$). Any state information that will be shared by the challenge protocol must be shared via calls to $\bar{\mathcal{G}}$ (*i.e.* challenge protocols are $\bar{\mathcal{G}}$-subroutine respecting), and the environment is specifically allowed to access $\bar{\mathcal{G}}$. Although $\mathcal{Z}$ is once again constrained to invoking a single instance of the challenge protocol, it is now possible for $\mathcal{Z}$ to internally mimic the behavior of multiple sessions of the challenge protocol, or other arbitrary network protocols, by making use of calls to $\bar{\mathcal{G}}$ wherever shared state information is required. Thus, we may avoid the need for JUC Theorem (and the implementation limitations it imposes), by allowing the environment direct access to shared state information (*e.g.* we would allow it to observe the Common Reference String when the shared functionality is the CRS functionality). We call this new security notion Externalized UC (EUC) security, and we say that a $\bar{\mathcal{G}}$-subroutine respecting protocol π $\bar{\mathcal{G}}$-EUC-emulates a protocol ϕ if π emulates ϕ in the basic UC sense with respect to $\bar{\mathcal{G}}$-externally constrained environments. We show that if a protocol π $\bar{\mathcal{G}}$-EUC-emulates ϕ, then it also GUC emulates ϕ (and vice versa, provided that π is $\bar{\mathcal{G}}$-subroutine respecting).

Theorem 1. *Let π be any protocol which invokes no shared functionalities other than (possibly) $\bar{\mathcal{G}}$, and is otherwise subroutine respecting (*i.e. *π is $\bar{\mathcal{G}}$-subroutine respecting). Then protocol π GUC-emulates a protocol ϕ, if and only if protocol π $\bar{\mathcal{G}}$-EUC-emulates ϕ.*

That is, provided that π only shares state information via a single shared functionality $\bar{\mathcal{G}}$, if it merely EUC-emulates ϕ with respect to that functionality, then π is a full GUC-emulation of ϕ! As a special case, we obtain that all basic UC emulations (which may not share *any* state information) are also GUC emulations.

Corollary 1. *Let π be any subroutine respecting protocol. Then protocol π GUC-emulates a protocol ϕ, if and only if π UC-emulates ϕ.*

The corollary follows by letting $\bar{\mathcal{G}}$ be the null functionality, and observing that the $\bar{\mathcal{G}}$-externally constrained environment of the EUC experiment collapses to become the same environment as that of the basic UC experiment when $\bar{\mathcal{G}}$ is the null functionality. Thus, it is sufficient to prove basic UC security for protocols with no shared state, or $\bar{\mathcal{G}}$-EUC security for protocols that share state only via $\bar{\mathcal{G}}$, and we will automatically obtain the full benefits of GUC security.

Figure 2 depicts the differences in the experiments of the UC models we have just described, in the presence of a single shared functionality $\bar{\mathcal{G}}$ (of course, the GUC framework is not inherently limited to special case of only one shared functionality). We further elaborate the technical details of these new models, and provide the proof of Theorem 1, in the full version of the paper [13].

We are now in a position to state a strong new composition theorem, which will directly incorporate the previous result (that proving EUC security is sufficient for GUC security). Let ρ be an arbitrary protocol (not necessarily subroutine respecting!) which invokes ϕ as a sub-protocol. We will write $\rho^{\pi/\phi}$ to denote a modified version of ρ that invokes π instead of ϕ, wherever ρ had previously invoked ϕ. We prove the following general theorem in the full version [13]:

Theorem 2 (Generalized Universal Composition). *Let ρ, π, ϕ be PPT multi-party protocols, and such that both ϕ and π are $\bar{\mathcal{G}}$-subroutine respecting, and π $\bar{\mathcal{G}}$-EUC-emulates ϕ. Then $\rho^{\pi/\phi}$ GUC-emulates protocol ρ.*

We stress that π must merely $\bar{\mathcal{G}}$-EUC-emulate ϕ, but that the resulting composed protocol $\rho^{\pi/\phi}$ fully GUC-emulates ρ, even for a protocol ρ that is not subroutine respecting.

3 Insufficiency of the Global CRS Model

In this section we demonstrate that a global CRS setup is *not* sufficient to GUC-realize even the basic two-party commitment functionality. We then further elaborate the nature of this insufficiency by considering some weaknesses in the security of previously proposed constructions in the CRS model. Finally, we suggest a new "intuitive" security goal, dubbed *full simulatability*, which we would like to achieve by utilizing the GUC-security model (and which was not previously achieved by any protocols in the CRS model).

3.1 Impossibility of GUC-realizing $\mathcal{F}_{com}$ in the $\bar{\mathcal{G}}_{gcrs}$ model

Recall that many interesting functionalities are unrealizable in the UC framework without any setup assumption. For instance, it is easy to see that the ideal authentication functionality, $\mathcal{F}_{auth}$, is unrealizable in the plain model. Furthermore, many two party tasks, such as Commitment, Zero-Knowledge, Coin-Tossing, Oblivious Transfer and others cannot be realized in the UC framework by two-party protocols, even if authenticated communication is provided [15, 16, 10].

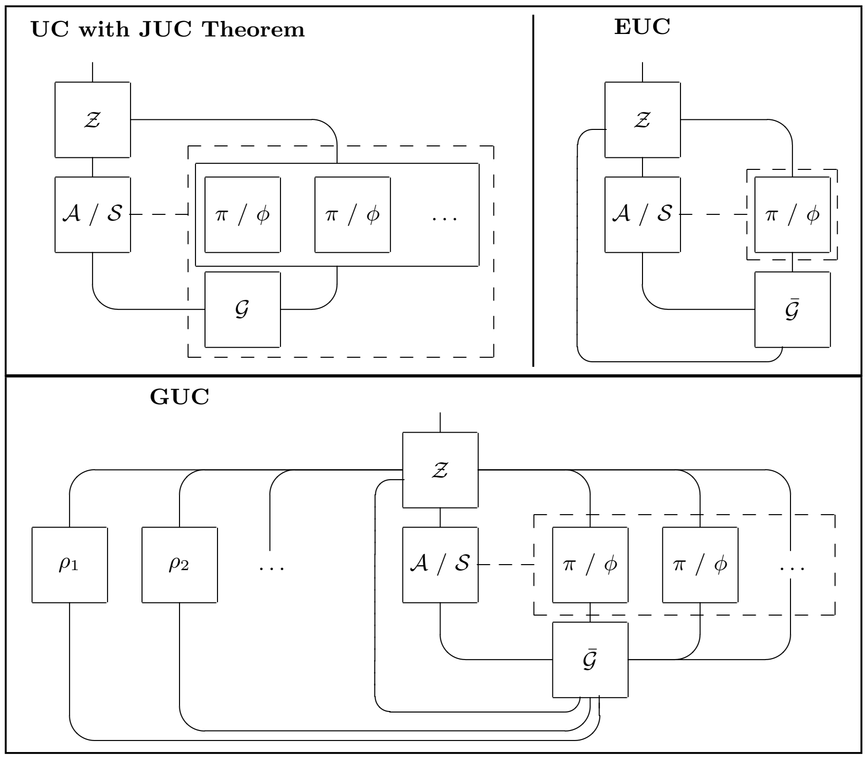

Fig. 2. Comparison of models. Using Basic UC with JUC Theorem to share state, only copies of the challenge protocol (or other protocols which may be jointly designed a priori to share $\mathcal{G}$) are allowed to access the common subroutine $\mathcal{G}$, and $\mathcal{Z}$ may only interact with the "multi-session" version of the challenge protocol. In the EUC paradigm, only a single session of the challenge protocol is running, but the shared functionality $\bar{\mathcal{G}}$ it uses is accessible by $\mathcal{Z}$. Finally, in the GUC setting, we see the full generality of arbitrary protocols $\rho_1, \rho_2, \ldots$ running in the network, alongside multiple copies of the challenge protocol. Observe that both $\mathcal{Z}$, and any other protocols invoked by $\mathcal{Z}$ (such as ρ_1), have direct access to $\bar{\mathcal{G}}$ in the GUC setting. Intuitively, the GUC modeling seems much closer to the actual structure of networked protocol environments.

As a recourse, the common reference string (CRS) model was used to re-assert the general feasibility results of [24] in the UC framework. That is, it was shown that any "well-formed" ideal functionality can be realized in the CRS model [15, 17]. However, the formulation of the CRS model in these works postulates a setting where the reference string is given *only to the participants in the actual protocol execution.* That is, the reference string is chosen by an ideal functionality, $\mathcal{G}_{crs}$, that is dedicated to a given protocol execution. $\mathcal{G}_{crs}$ gives the reference string only to the adversary and the participants in that execution.

Intuitively, this formulation means that, while the reference string need not be kept secret to guarantee security, it cannot be safely used by other protocol executions. In other words, no security guarantees are given with respect to executions that use a reference string that was obtained from another execution rather than from a dedicated instance of $\mathcal{G}_{crs}$. (The UC with joint state theorem of [18] allows multiple executions of certain protocols to use the same instance of the CRS, but it requires all instances that use the CRS to be carefully designed to satisfy some special properties.)

In contrast, we are interested in modeling a setting where the same CRS is globally available to all parties and all protocol executions. This means that a protocol π that uses the CRS must take into account the fact that the same CRS may be used by *arbitrary* other protocols, even protocols that were specifically designed to interact badly with π. Using the GUC security model defined in Section 2, we define this weaker setup assumption as a *shared* ideal functionality that provides the value of the CRS not only to the parties of a given protocol execution, but rather to all parties, and even directly to the environment machine. In particular, this global CRS functionality, $\bar{\mathcal{G}}_{gcrs}$, exists in the system both as part of the protocol execution and as part of the ideal process. Functionality $\bar{\mathcal{G}}_{gcrs}$ is presented in Figure 3.

Functionality $\bar{\mathcal{G}}_{gcrs}$

Parameterized by a distribution D, $\bar{\mathcal{G}}_{gcrs}$ proceeds as follows, when activated by any party:

1. If no value has been previously recorded, choose a value $d \xleftarrow{R} D$, and record the value d.
2. Return the value d to the activating party.

Fig. 3. The Global Common Reference String functionality. The difference from the Common Reference String functionality $\mathcal{G}_{crs}$ of [10, 15] is that $\mathcal{G}_{crs}$ provides the reference string only to the parties that take part in the actual protocol execution. In particular, the environment does not have direct access to the reference string.

We demonstrate that $\bar{\mathcal{G}}_{gcrs}$ is insufficient for reproducing the general feasibility results that are known to hold in the $\mathcal{G}_{crs}$ model. To exemplify this fact, we show that no two-party protocol that uses $\bar{\mathcal{G}}_{gcrs}$ as its only setup assumption GUC-realizes the ideal commitment functionality, $\mathcal{F}_{com}$ (presented in Figure 4). The proof, which we provide in the full version of this work [13], follows essentially the same steps as the [15] proof of impossibility of realizing $\mathcal{F}_{com}$ in the plain model. The reason that these steps can be carried out even in the presence of $\bar{\mathcal{G}}_{gcrs}$ is, essentially, that the simulator obtains the reference string from an external entity ($\bar{\mathcal{G}}_{gcrs}$), rather than generating the reference string by itself. We conjecture that most other impossibility results for UC security in the plain model can be extended in the same way to hold for GUC security in the presence of $\bar{\mathcal{G}}_{gcrs}$.

Functionality $\mathcal{F}_{com}$

Commit Phase: Upon receiving a message $(\texttt{commit}, sid, C, V, b)$ from party C, where $b \in \{0,1\}$, record the value b and send $(\texttt{receipt}, sid, C, V)$ to V and the adversary. Ignore any future `commit` messages.

Reveal Phase: Upon receiving a message $(\texttt{reveal}, sid)$ from C: If a value b was previously recorded, then send the message $(\texttt{reveal}, sid, b)$ to V and the adversary and halt. Otherwise, ignore.

Fig. 4. The Commitment Functionality $\mathcal{F}_{com}$ (see [15])

Theorem 3. *There exists no bilateral, terminating protocol π that GUC-realizes $\mathcal{F}_{com}$ and uses only the shared functionality $\bar{\mathcal{G}}_{gcrs}$. This holds even if the communication is ideally authentic.*

In fact, it can be shown that the above impossibility result extends beyond the mere availability of $\bar{\mathcal{G}}_{gcrs}$ to any circumstance where the shared functionality will only provide information globally (or, yet more generally, the impossibility holds whenever all the shared information available to protocol participants can also be obtained by the environment). For instance, this impossibility will hold even in the (public) random oracle model, which is already so strong that it cannot truly be realized without the use of a fully interactive trusted party. Another interpretation of this result is that no completely *non-interactive* global setup can suffice for realizing $\mathcal{F}_{com}$. The next section studies the problem of realizing $\mathcal{F}_{com}$ using setup assumptions with minimal interaction requirements.

3.2 Deniability and Full Simulatability

To demonstrate that the problems with using a global CRS to realize $\mathcal{F}_{com}$, in the fashion of [17], are more than skin deep technicalities that arise only in the GUC framework we now consider the issue of deniability. Intuitively, a protocol is said to be "deniable" if it is possible for protocol participants to deny their participation in a protocol session by arguing that any "evidence" of their participation (as obtained by other, potentially corrupt protocol participants) could have been fabricated.

Recalling the intuition outlined in the introduction, we would like realized protocols to guarantee the same security as the ideal functionalities they realize, meaning that the adversary will learn nothing more from attacking the protocol than could be learned from attacking its corresponding ideal functionality. Protocols realized with such a guarantee are inherently *deniable*, since a protocol participant can accurately argue that any information sent during the protocol session could have been obtained by an adversary using only the output from the ideal functionality[4] in an attack simulation conducted entirely without his or her actual participation.

[4] Of course, if the output of the ideal functionality "incriminates" a user by revealing some of his secrets, the resulting protocol does not meet our intuitive understanding of the word "deniable". Still, the protocol itself may be said to be "as deniable" as the functionality it realizes.

For instance, if we consider the ideal functionality for Zero Knowledge (ZK), we expect that any secure realization of that functionality should reveal no information to the adversary beyond the output of the ideal functionality (which contains only a single bit). In particular, that output can easily be generated entirely without the help of the prover, and thus the prover should be able to deny his participation in the protocol, since it reveals no information that could not have been obtained independently. However, we already know from the result of [28] that it is impossible to achieve such deniability for ZK in the CRS model. Indeed, we may see that the UC simulator for ZK functionality in [17] chooses a fresh CRS, and generates the simulated protocol transcripts with respect to that, instead of the published real-world CRS. Thus, if a protocol transcript makes use of the real-world CRS, it could not have been obtained via simulation (so a successful prover is indeed incriminated by the transcript).

When there is no deniability, the adversary is truly able to obtain some valuable information by observing protocol interactions that would not by revealed by the ideal functionality. Thus we have found a practical example of security loss that directly results from the relaxations of UC security inherent in the CRS technique of [17]. We can now clearly see that the impossibility of realizing $\mathcal{F}_{com}$ via the CRS model in the GUC setting is due to a meaningful strengthening of security guarantees (since deniability is guaranteed in the GUC setting, and that guarantee is not achieved by protocols realized in the CRS model).

On an intuitive level, it might be helpful to consider the issue of deniability in light of the "real world" resources required in order to run the GUC simulator to simulate a given protocol session. If the resources required to simulate a protocol session are readily available, then the protocol is *plausibly deniable* (since it is plausible that information obtained from the protocol was the result of a simulation). If the resources are difficult or impossible to obtain, then the protocol is not plausibly deniable. We wish to employ simulation techniques that require only minimal resources to conduct a simulation, increasing the plausibility of denials (as well as decreasing the value of any information that an adversary might obtain by attacking a secure protocol). Thus, we use the term *fully simulatable* to refer to any plausibly deniable protocol realized in the GUC framework.

From this vantage point, we observe that the resource required to conduct the protocol simulations in [17] is a "trapdoor" for the CRS. In particular, the CRS must be "rigged" with such a trapdoor *a priori*. Such rigging is certainly not plausible when there is a trusted party choosing the CRS, and this is in fact the root of the deniability problem for the CRS model. Furthermore, knowledge of this trapdoor implies the ability to completely violate security of any protocol constructed using the techniques of [17], and thus there would be no security against any entity capable of simulating protocols. Similarly, in the "imaginary angel" model of [30], the simulator requires access to super-polynomial time functionalities that are certainly not plausibly available in the real world (and thus, the deniability problem arises there as well). Indeed, if the "imaginary angels" of [30] were to somehow be made practical in the real world, all security would be lost.

We comment that, although we do not make any attempt to formalize a "general" notion of deniability here, the guarantee we seek to provide is that protocols are "as deniable" in the real world as they would have been in the ideal world (past works did not satisfy even this basic requirement). In fact, as we will see, our particular realization of fully simulatable security will guarantee that even "on line" (interactive) deniability is preserved, since the simulator can very practically be run in real time. Indeed, as long as an honest party P never deviates from the protocol, it is not possible for other (even corrupt) protocol participants to conclusively demonstrate P's participation in the protocol session to a third party, *even while the protocol is ongoing*!

4 Fully Simulatable General Computation

We now turn our attention to the problem of *constructing* fully simulatable GUC-secure protocols. That is, we would like it to be possible for a real-world adversary to simulate the effects of any attack on a protocol, without actually attacking the protocol (instead utilizing only the information that would be revealed by an ideally secure realization). The result of Section 3 implies that we cannot do this in the CRS model (if we correctly model a globally available CRS). Thus, we must consider alternative models if we hope to achieve our goal.

To that end, we would like to find reasonable alternative global setup assumptions that allow for realizing interesting tasks. That is, we are looking for shared functionalities $\bar{\mathcal{G}}$ (as defined in Section 2), so that on the one hand $\bar{\mathcal{G}}$ will be implementable in reality with reasonable trust assumptions, and on the other hand we will have protocols that GUC-realize interesting functionalities and still use no setup (*i.e.*, no ideal functionalities) other than $\bar{\mathcal{G}}$. We say that such GUC-secure protocols are "fully simulatable" since the GUC-simulator for attacking the ideal protocol can, in a very practical sense, be run directly by the adversary. This allows the adversary to simulate *the same information* that can be gotten by attacking any session of the real protocol, without actually performing any attack.

We first observe that if the system is equipped with a "fully interactive trusted party" that realizes, say, $\mathcal{F}_{mcom}$, the multi-session variant of $\mathcal{F}_{com}$, by interacting separately and privately with each session, then we can directly use the protocol of [17] to GUC-realize any "well-formed" functionality. However, we would like to find more reasonable global setup assumptions, and in particular assumptions that require less interaction from the trusted entity. (Indeed, this realization requires the trusted party to perform strictly more work than it would by directly computing the desired functionalities, *i.e.* the trivial realization of ideal model functionalities). Although it is clear that we can achieve fully simulatable protocols by using highly interactive trusted parties to compute functionalities, it seems to be a more difficult problem to realize GUC-secure protocols using an "offline" shared functionality. Indeed, by our earlier impossiblity results, *some* degree of interaction would seem to be essential, so we begin by considering the idea of limiting the interaction to a "registration phase".

4.1 The KRK Model

We observe that the "key registration with knowledge (KRK)" setup of [5], can be modified to serve as a shared functionality, allowing us to realize any "well-formed" ideal functionality against non-adaptive ("static") adversaries using the techniques of that work. Although the setup phase is interactive (parties must register their public keys with registration authorities), it is possible to show (with some minor modifications) that the protocol of [5] can allow the trusted party to remain "offline" for all subsequent protocol activity.

Functionality $\mathcal{G}_{krk}$

$\mathcal{G}_{krk}$ proceeds as follows, given a (deterministic) key generation function Gen (with security parameter λ), running with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$:

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

Retrieval: When receiving a message $(\texttt{retrieve}, \mathsf{sid}, P_i)$ from party P_j (where $j \neq i$), if there is a previously recorded tuple of the form (P_i, PK_i, SK_i), then return $(\mathsf{sid}, P_i, PK_i)$ to P_j. Otherwise return $(\mathsf{sid}, P_i, \perp)$ to P_j. When receiving a message $(\texttt{retrieve}, \mathsf{sid}, P_i)$ from party P_i, if there is a previously recorded tuple of the form (P_i, PK_i, SK_i), then return $(\mathsf{sid}, P_i, PK_i, SK_i)$ to P_i. Otherwise, return $(\mathsf{sid}, P_i, \perp)$ to P_i.

Fig. 5. The Knowledge-based Key Registration Functionality (similar to that of [5]). Note that each instance of $\mathcal{G}_{krk}$ can only be invoked by the parties of a single protocol session (*i.e.* with a fixed sid). After converting this ideal functionality to a shared functionality, $\bar{\mathcal{G}}_{krk}$, and restricting retrieval of private keys to *corrupt parties only*, it is possible GUC-realize any functionality using only a single public key per-party.

Recall that the KRK setup of [5] is an ideal functionality $\mathcal{G}_{krk}$ (shown in Figure 5), that chooses a private and public key pair for each registered party and lets all parties know the value of the public key. In the natural version of the KRK setup, parties are also allowed to retrieve their own secret keys. Since $\mathcal{G}_{krk}$ is not a shared functionality, each instance of a protocol will have its own instance of $\mathcal{G}_{krk}$, which does not lend itself to easy implementation. To fix this, we re-formulate $\mathcal{G}_{krk}$ as a shared functionality, $\bar{\mathcal{G}}_{krk}$, that chooses public keys only *once* per party – used by all instances. (This modeling makes $\bar{\mathcal{G}}_{krk}$ significantly easier to implement in a real system, since only one key is required for each party.) Furthermore, we add a simple modeling restriction: we only allow parties to learn their own secret keys if they are corrupt.[5] Using this new $\bar{\mathcal{G}}_{krk}$ setup, the protocol of [5] works (with minor modifications) even in the GUC-security model, provided that (a) party corruptions are non-adaptive, and

[5] This modeling restriction is discussed in further detail in Section 4.2.

(b) all parties with the same PID (party identity) are corrupted together – we call such corruption pattern $\mathsf{PID\text{-}wise}$.

Theorem 4. *The [5] protocol GUC-realizes $\mathcal{F}_{zk}$, even when given access only to $\bar{\mathcal{G}}_{krk}$, as long as the party corruptions are non-adaptive and PID-wise.*

The proof of this theorem is by a natural extension of the proof in [5] to the EUC framework (which is, of course, equivalent to GUC), but surprisingly, we can achieve a much stronger goal than non-adaptive GUC security with interactive setup.

4.2 The Augmented CRS Model

Although it may seem that *at least* an interactive "registration phase" is required in order to avoid our earlier impossibility result, we show that something even *less interactive* will suffice. We propose a further simplification of $\bar{\mathcal{G}}_{krk}$, denoted $\bar{\mathcal{G}}_{acrs}$, and a protocol that GUC-realizes $\mathcal{F}_{com}$ (and thus any well-formed functionality) having access only to $\bar{\mathcal{G}}_{acrs}$. Unlike $\bar{\mathcal{G}}_{krk}$, the $\bar{\mathcal{G}}_{acrs}$ shared functionality does not *require* any interaction (much like $\mathcal{G}_{crs}$), but merely offers a one-time use interactive "key retrieval" service to those who choose to use it. Therefore, we refer to this new setup assumption as the *Augmented CRS* (ACRS) model. In particular, protocols realized in the ACRS model will not actually make use of the key retrieval service, since the model only allows corrupt parties to retrieve their keys. Thus, we are assured that honest parties need never communicate interactively with $\bar{\mathcal{G}}_{acrs}$.

Somewhat counter-intuitively, it is even crucial that uncorrupted parties in ACRS model never "bother" to obtain their secret keys from the trusted authority (since even an honest party may inadvertently execute a rogue protocol, which might expose the secret key). Similarly, it is crucial that corrupted parties have access to their secret keys, since otherwise they would be unable to conduct attack simulations. (On a side note, security is still guaranteed to honest parties who obtain their keys and use them to conduct attack simulations *provided that* they only use their keys for simulation purposes.) To enforce the protocol design criteria that honest parties should not require access to their secret keys, we directly define the $\bar{\mathcal{G}}_{acrs}$ functionality so that it refuses to supply secret keys to honest parties. (Of course, a direct realization of $\bar{\mathcal{G}}_{acrs}$ by a trusted party cannot actually determine which parties are honest, yet intuitively this modeling should still suffice. In fact, it is not problematic even if the real-world trusted party gives keys to honest parties, as long as they are careful to protect their own security by keeping their keys secret.)

More formally, our new shared functionality $\bar{\mathcal{G}}_{acrs}$ is parameterized by two functions, Setup and $\mathsf{Extract}$. It first chooses a random secret value MSK and a public value $PK \leftarrow \mathsf{Setup}(MSK)$, and publicizes PK (as a CRS). Next, whenever a *corrupted* party P asks for its secret key, $\bar{\mathcal{G}}_{acrs}$ returns the value $SK_P \leftarrow \mathsf{Extract}(PK; P; MSK)$. The functionality is presented in Figure 6.

Functionality $\bar{\mathcal{G}}_{acrs}^{\mathsf{Setup},\mathsf{Extract}}$

Initialization Phase: At the first activation, compute a Common Reference String $(PK) \leftarrow \mathsf{Setup}(MSK)$ for a randomly chosen λ-bit value MSK, and record the pair (PK, MSK).

Providing the public value: Whenever activated by a party requesting the CRS, return PK to the requesting party and the adversary.

Dormant Phase: Upon receipt of a message $(\texttt{retrieve}, \mathsf{sid}, P)$ from a *corrupt* party P, return the value $SK_P \leftarrow \mathsf{Extract}(PK; P; MSK)$ to P. (Receipt of this message from honest parties is ignored.)

Fig. 6. The Identity-Based Augmented CRS Shared Functionality

Comparing $\bar{\mathcal{G}}_{krk}$ and $\bar{\mathcal{G}}_{acrs}$. The main difference between $\bar{\mathcal{G}}_{acrs}$ and $\bar{\mathcal{G}}_{krk}$ (the global variant of $\mathcal{G}_{krk}$) is that in $\bar{\mathcal{G}}_{acrs}$ there is a single public value, whereas in $\bar{\mathcal{G}}_{krk}$ an extra public value must be given per party identity. Using a paradigm analogous to the identity-based encryption of [6], we avoid the use of per-party public keys and replace them with a single *short* "master public key" (and indeed our constructions use short public keys that depend only on the security parameter). This property, combined with the fact that the parties who follow their protocols never obtain their secret keys, makes $\bar{\mathcal{G}}_{acrs}$ very close in spirit to a global CRS setup as in $\bar{\mathcal{G}}_{gcrs}$. In fact, in light of the far-reaching impossibility result for $\bar{\mathcal{G}}_{gcrs}$, $\bar{\mathcal{G}}_{acrs}$ can be regarded as a "minimum interaction" global setup.

We note that, as pointed out in [5], $\bar{\mathcal{G}}_{krk}$ can be naturally implemented by multiple "registration authorities", where no single authority needs to be fully trusted by all. (However, we once again stress that $\bar{\mathcal{G}}_{krk}$ requires *all* parties, *even those who honestly follow their protocols*, to interactively register with a *some* authority and obtain a public key.) Similarly, multiple instances of $\bar{\mathcal{G}}_{acrs}$ may be run by different trusted authorities. Unlike $\bar{\mathcal{G}}_{krk}$, however, parties may participate in protocols while placing their trust in an arbitrary trusted authority, without ever having registered with *any* authority. This is extremely useful for settings where PKIs are not desirable or easy to implement, and where no single "global" authority is available (see *e.g.* [4]).[6]

In the full version of this work [13], we prove the following result:

Theorem 5. *There exists a protocol that GUC-realizes $\mathcal{F}_{com}$ given access to $\bar{\mathcal{G}}_{acrs}$. Party corruptions can be adaptive (and in the non-erasure model), as long as they are PID-wise.*

[6] In fact, the protocol we will describe in Section 5 can also support a "graceful failure" approach similar to that outlined in [5], in the scenario where protocol participants do not mutually trust any single authority. That is, by using suitable "graceful" tools (in the case of our protocol, a "graceful" IBTC) , we can ensure full GUC security if trustworthy authorities are used by all parties, and ordinary stand-alone security for party P in the case where only party P's authority is trustworthy (even if party P's own authority is made completely unavailable after publishing its reference string, and/or is later corrupted subsequent to the completion of the protocol).

Finally, we note that a GUC secure realization of $\mathcal{F}_{com}$ is indeed sufficient to GUC-realize any "well-formed" *multi-party* functionality. This may be accomplished by first using $\mathcal{F}_{com}$ to realize $\mathcal{F}_{zk}$ (as in [17]), and then using $\mathcal{F}_{zk}$ to realize the "one-to-many" Zero-Knowledge functionality, $\mathcal{F}_{zk}^{1:M}$ (via the technique of [29]). The protocol compiler from [17] can then be used to yield a UC-secure realization of any well-formed multi-party functionality in the $\mathcal{F}_{zk}^{1:M}$-hybrid model, without using any shared state (thus it is also a GUC-secure realization by Corollary 1).

5 GUC-Realizing $\mathcal{F}_{com}$ using the $\bar{\mathcal{G}}_{acrs}$ Global Setup

We now describe the construction of a protocol satisfying the conditions of Theorem 5, above. When combined with the compiler from [17], such a *fully simulatable* realization of $\mathcal{F}_{com}$ yields a fully simulatable realization of any well-formed two-party or multi-party functionality. Furthermore, we show that, in addition to requiring only the more minimal $\bar{\mathcal{G}}_{acrs}$ setup, our protocol achieves significantly stronger properties than the fully simulatable protocol from [5] realized in the $\bar{\mathcal{G}}_{krk}$ model. (Of course, our protocol can also be trivially modified for use in the $\bar{\mathcal{G}}_{krk}$ model, where it will enjoy the same strengthened security guarantees.)

Firstly, our protocol realizing $\mathcal{F}_{com}$ remains secure even in the presence of adaptive corruptions (whereas the protocol of [5] does not). Intuitively, adaptive security seems to be difficult to attain in either the $\bar{\mathcal{G}}_{krk}$ or $\bar{\mathcal{G}}_{acrs}$ models, since an adaptive adversary is eventually able to learn nearly all secrets in the system (save only for the random coins of the trusted party), yet the simulator must make use of these secrets. Our protocol essentially skirts this difficulty by using some additional interactivity. Remarkably, the same technique also enables it to maintain the security of past executions even when the trusted party implementing $\bar{\mathcal{G}}_{acrs}$ is later corrupted (revealing the random coins used to generate the CRS, leaving the overall system with no secrets at all)! That is, our protocol guarantees that past transcripts of protocol interactions can *never* be used to compromise the security or deniability of honest parties *even if the trusted party is later corrupted*. Security is only lost when the trusted party acts maliciously *prior to*, or *during* protocol execution. This kind of "forward security" with respect to the trusted party further minimizes the trust assumptions required to realize $\bar{\mathcal{G}}_{acrs}$ in the real-world. For instance, an adversary cannot later coerce the trusted party into breaking the security of an honest party after the completion of the protocol. Such forward security cannot be achieved using the protocol of [5] since knowledge of the secret key allows "extraction" from past commitments, breaking privacy. Similarly, the protocol of [17] also loses privacy of past transcripts if the trusted party implementing the CRS setup later reveals a trapdoor.

5.1 High-level description of the protocol

Our protocol for realizing $\mathcal{F}_{com}$ in the $\bar{\mathcal{G}}_{acrs}$ shared hybrid model, which we call Protocol UAIBC (for UC Adaptive Identity-Based Commitment), relies on two new techniques. First, we construct an *identity-based* trapdoor commitment (IBTC) which enjoys adaptive security. Then we provide a general transformation from any IBTC into a protocol that securely implements $\mathcal{F}_{com}$.

Constructing IBTC. In the setting of IBTC a single "master-key" is made public. Additionally, all parties can obtain a private-key that is associated to their party identifier. (Note that this setting corresponds exactly to the interface of $\bar{\mathcal{G}}_{acrs}$.) Intuitively, an IBTC is a commitment scheme with the additional property that a committer who *knows* the receiver's secret-key can *equivocate* commitments (*i.e.*, it can open up commitments to any value, breaking the binding property). Furthermore, an adversary that obtains the secret-keys of multiple parties still should not be able to violate the binding property of commitments sent to parties for which it has not obtained the secret-key.

Constructions of IBTCs were previously known in the Random Oracle Model [2, 32]. Here we provide a conceptually simple approach to constructing an adaptively secure IBTC from any one-way function, in the standard model. Our approach relies on the use of Σ-protocols [14], in an approach based on that of [21] (and perhaps surprisingly can result in a very practical protocol). On a very high-level (and very oversimplified) the general idea is as follows: 1) let the master-key be a public-key for a signature scheme, 2) let the secret-key for a party be a signature on its party identifier, and 3) construct a commitment scheme where the reveal phase consists of a "proof" that *either* the revealed value is consistent with the value committed to, *or* the committer knows a signature on the receiver's party identifier (this "proof" must also "hide" which of these two statements actually holds). We mention that the actual instantiation of this idea is somewhat more involved, in order to guarantee adaptive security, and we provide the full details of our construction in [13].

From IBTC to GUC Commitments. Recall that a protocol for realizing $\mathcal{F}_{com}$ must intuitively satisfy two properties (in addition to the traditional binding and hiding properties of any commitment scheme): 1) it must be equivocable, and 2) it must be extractable. We show how to transform any "equivocable" commitment scheme (such as an IBTC) into a protocol for securely realizing $\mathcal{F}_{com}$ (for single bit commitments). Previously similar types of transformations have appeared in the literature (e.g., [17], [7]). Unfortunately all such transformations either require some additional *non-global* setup (and are thus not applicable in out setting), or only work in the case of static security. We now turn our focus to the protocol UAIBC, which GUC-realizes the $\mathcal{F}_{com}$ functionality via a novel transformation of an IBTC from a mere equivocable commitment (in the standard model), to an equivocable *and* extractable commitment secure against adaptive corruptions in the GUC-security model. We remark that our transformation technique can be employed by substituting *any* merely equivocable commitment scheme (such as standard public key based trapdoor commitments) in place of the IBTC in our protocol, and will yield a scheme that is both equivocable and extractable, a general approach that may prove useful in many other contexts.

On a high-level, protocol UAIBC proceeds as follows. The committer P_i and receiver P_j first perform a coin-tossing to generate a public-key K for a dense crypto-system. This coin-tossing requires the receiver to use an IBTC, and has the property that if the committer is corrupted, the outcome of the coin-tossing can be set to any value. After a completed coin-tossing, the committer commits to a single bit b using an IBTC (let c denote this commitment), and additionally sends an auxiliary string e: e is either a random string in case $b = 1$, and an encryption to the decommitment information of c if $b = 0$. (We here require that the encryption scheme used has *pseudo-random ciphertexts*.) In the reveal phase, the committer is required to provide correct decommitment information for c, and additionally reveal the value encrypted in e in case $b = 0$. We graphically illustrate the operation of this protocol in Figure 7. In the full version of this work [13], we prove that UAIBC GUC-realizes the $\mathcal{F}_{com}$ ideal functionality in a *fully simulatable* manner (even for adaptive adversaries in the non-erasure setting), and in addition features the aforementioned "forward security" property.

Step #	$P_i(b)$		P_j
Commit			
(1)		$\xrightarrow{\texttt{commit}, sid, P_i, P_j}$	
(2)		$\xleftarrow{c_k}$	$k_1 \leftarrow_r \{0,1\}^\lambda$ $(c_k, d_k) = \mathsf{Com}(P_i; k_1)$
(3)	$k_2 \leftarrow_r \{0,1\}^\lambda$	$\xrightarrow{k_2}$	
(4)		$\xleftarrow{d_k}$	$K = k_1 \oplus k_2$
(5)	$k_1' = \mathsf{Open}(P_i; c_k, d_k)$ $K = k_1' \oplus k_2$ $(c, d) = \mathsf{Com}(P_j; b)$ if $b = 0$ $\quad e = E_K(r; d)$ if $b = 1$ $\quad e =$ random	$\xrightarrow{c,\ e}$	
Reveal			
(1&2)	if $b = 0$ if $b = 1$	$\xrightarrow{\texttt{b=0},\ d,\ r}$ $\xrightarrow{\texttt{b=1},\ d}$	if $b = 0$ $\ E_K(r; d) \stackrel{?}{=} e$ $\mathsf{Open}(P_j; c, d) \stackrel{?}{=} b$

Fig. 7. Operation of Protocol UAIBC, with party P_i committing bit b to party P_j. Note that Com and Open are operations for an IBTC (the first input is the identity of the recipient), and E_K is a Dense OT-PRC secure encryption using key K (the first input is the random coins fed to the encryption operation, and the second is the plaintext). Steps 2 to 4 of the **Commit** phase are essentially a coin-tossing protocol, whereas the subsequent steps are similar to the protocol of [17].

6 Acknowledgments

The authors would like to thank Philip MacKenzie, Johan Håstad, and Silvio Micali for insightful observations, as well as various anonymous referees for helpful comments and suggestions.

References

1. M. Abe, and S. Fehr. Perfect NIZK with Adaptive Soundness. In *Proc. of TCC*, 2007.
2. G. Ateniese and B. de Medeiros. Identity-based Chameleon Hash and Applications. *Proc. of Financial Cryptography*, 2004. Available at `http://eprint.iacr.org/2003/167/`.
3. D. Beaver. Secure Multi-party Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority. in *J. Cryptology*, vol 4., pp. 75–122, 1991.
4. B. Barak, R. Canetti, Y. Lindell, R. Pass and T. Rabin. Secure Computation Without Authentication. In *CRYPTO 2005*, Springer-Verlag (LNCS 3621), pages 361-377, 2005.
5. B. Barak, R. Canetti, J. Nielsen and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *Proc. of FOCS*, 2004.
6. D. Boneh, and M. Franklin. Identity Based Encryption from the Weil Pairing. In *Proc. of Crypto*, 2001.
7. B. Barak and Y. Lindell. Strict Polynomial-time Simulation and Extraction. In *SIAM J. Comput.*, 33(4), pp. 783-818, 2004.
8. B. Barak and A. Sahai, How To Play Almost Any Mental Game Over the Net - Concurrent Composition via Super-Polynomial Simulation. In *Proc. of FOCS*, 2005.
9. R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology,* Vol. 13, No. 1, winter 2000.
10. R. Canetti. Universally Composable Security: A New paradigm for Cryptographic Protocols. In *Proc. of FOCS*, pages 136–145, 2001.
11. R. Canetti. Universally Composable Security: A New paradigm for Cryptographic Protocols. In *Cryptology ePrint Archive*, Report 2000/067, **revised edition from Dec. 2005**. Available at: http://eprint.iacr.org/2000/067
12. R. Canetti. Universally Composable Signature, Certification, and Authentication. In *Proc. of CSFW*, p. 219, 2004.
13. R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally Composable Security with Global Setup. In *Cryptology ePrint Archive*, Report 2006/432. Available at: http://eprint.iacr.org/2006/432
14. R. Cramer, I. Damgard, B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *Proc. of CRYPTO*, pp. 174–187, 1994.
15. R. Canetti and M. Fischlin. Universally Composable Commitments. In *Proc. of Crypto*, pages 19–40, 2001.
16. R. Canetti, E. Kushilevitz and Y. Lindell. On the Limitations of Universally Composable Two-Party Computation Without Set-Up Assumptions. In *Proc. of Eurocrypt*, Springer-Verlag (LNCS 2656), pp. 68–86, 2003.
17. R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *Proc. of STOC*, pp. 494–503, 2002.

18. R. Canetti and T. Rabin. Universal Composition with Joint State. In *Proc. of Crypto 2003*, Springer-Verlag, pp. 265-281, 2003.
19. I. Damgard and J. Nielsen. Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In *Proc. of Crypto*, Springer-Verlag, pp. 581–596, 2002.
20. Y. Dodis and S. Micali. Parallel Reducibility for Information-Theoretically Secure Computation. In *Proc. of Crypto*, Springer-Verlag (LNCS 1880), pp. 74–92, 2000.
21. U. Feige. Alternative Models for Zero Knowledge Interactive Proofs. Ph.D. thesis, Weizmann Institute of Science, Rehovot, Israel, 1990.
22. U. Feige, D. Lapidot, and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String. In *Proc. of FOCS*, 1990.
23. S. Goldwasser, and L. Levin. Fair Computation of General Functions in Presence of Immoral Majority. *CRYPTO '90, LNCS 537,* 1990.
24. O. Goldreich, S. Micali, and A. Wigderson. How to Solve any Protocol Problem. In *Proc.of STOC*, 1987.
25. D. Hofheinz, J. Muller-Quade, and D. Unruh. Universally Composable Zero-Knowledge Arguments and Commitments from Signature Cards. In *Proc. of the 5th Central European Conference on Cryptology MoraviaCrypt 2005*, June 2005.
26. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated Verifier Proofs and their Applications. In *Proc. of Eurocrypt*, Springer-Verlag, 1996.
27. S. Micali and P. Rogaway. Secure Computation. unpublished manuscript, 1992. Preliminary version in *CRYPTO '91, LNCS 576*, 1991.
28. R. Pass. On Deniabililty in the Common Reference String and Random Oracle Model. In *Proc. of Crypto*, LNCS 2729, pp. 216–337, 2003.
29. R. Pass. Bounded-Concurrent Secure Multi-Party Computation with a Dishonest Majority. In *Proc. of STOC*, pp. 232–241, 2004.
30. M. Prabhakaran and A. Sahai. New Notions of Security: Achieving Universal Composability without Trusted Setup. In *Proc. of STOC*, 2004.
31. B. Pfitzmann and M. Waidner. Composition and Integrity Preservation of Secure Reactive Systems. In *Proc. of ACM CCS*, pages 245–254, 2000.
32. F. Zhang, R. Safavi-Naini and W. Susilo. ID-Based Chameleon Hashes from Bilinear Pairings. Available at `http://eprint.iacr.org/2003/208/`.