Universal Composition with Joint State

Ran Canetti and Tal Rabin

IBM T.J. Watson Research Center. E-mail: {canetti,talr}@watson.ibm.com.

Abstract. Cryptographic systems often involve running multiple concurrent instances of some protocol, where the instances have some amount of joint state and randomness. (Examples include systems where multiple protocol instances use the same public-key infrastructure, or the same common reference string.) Rather than attempting to analyze the entire system as a single unit, we would like to be able to analyze each such protocol instance as stand-alone, and then use a general composition theorem to deduce the security of the entire system. However, no known composition theorem applies in this setting, since they all assume that the composed protocol instances have disjoint internal states, and that the internal random choices in the various executions are independent.
We propose a new composition operation that can handle the case where different components have some amount of joint state and randomness, and demonstrate sufficient conditions for when the new operation preserves security. The new operation, which is called *universal composition with joint state* (and is based on the recently proposed universal composition operation), turns out to be very useful in a number of quite different scenarios such as those mentioned above.

Keywords: cryptographic protocols, protocol composition, security analysis

1 Introduction

Cryptographic systems often involve multiple concurrent instances of a large variety of different protocols, where different instances are being run by different sets of mutually suspicious parties; furthermore, they are often deployed in ever-changing and unpredictable environments that may involve additional unknown protocols. Directly analyzing the security of such a system as a single unit is often prohibitively complex. Instead, we would like to be able to de-compose a complex cryptographic system to simpler components, prove the security of each component in isolation, and then deduce the security of the re-composed system.

This "de-compositional approach" is indeed very attractive. However, its soundness hinges on our ability to deduce the security of the entire system from the security of the components. Here *secure composition theorems* come in handy. Roughly speaking, such theorems assert that if a protocol is secure when considered in isolation, then it remains secure even when multiple instances of this protocol are run in the same system, or (in some cases) even when the protocol is used as a component of an arbitrary system.

A number of *composition operations* (i.e., ways to put together protocols in order to get a composite protocol) have been studied in the context of preservation of security. These include *sequential, parallel,* and *concurrent* composition, when the composed instances are run either by the same sets of parties, or by different sets of parties (e.g., [GK96,Bea91,DDN00,GO94,DNS98,Gol02]). They also include the more general operations of *modular* and *universal* composition, where protocols call other protocols as subroutines [MR91,Can00,DM00,PSW00,Can01]. However, all known composition theorems have the following limitation. They all assume, at least as far as the honest parties are concerned, that the local state of each one of the composed protocol instances is disjoint from the local states of all the other protocol instances run by the party. Furthermore, for each protocol instance, the honest parties are required to use a "fresh" random input that is independent of all the other random inputs. Thus, none of the known composition theorems is applicable when trying to de-compose a system into simpler components, while allowing the components to have some amount of joint state.

In contrast, many cryptographic systems consist of multiple concurrent instances of some (relatively simple) protocol, where all the instances have some limited amount of joint state and joint randomness. Prevalent examples include key-exchange and secure communication protocols, where multiple protocol instances use the same instance of a public-key signature or encryption scheme. Another set of examples include protocols in the common reference string model, where multiple instances use the same short reference string. Indeed, when attempting to analyze such systems, there was so far no alternative but to directly analyze the entire multi-instance system as a single unit.

We formulate a new composition operation for cryptographic protocols, that is applicable even in the case where multiple protocol instances have some amount of joint state. We also demonstrate sufficient conditions for when this operation preserves security. Our new operation, Universal Composition with joint state (JUC, pronounced "juicy"), is formulated within the Universal Composition (UC) framework [Can01], and extends its powers. The new operation drastically simplifies the analysis of systems where multiple instances have joint state, by allowing us to apply the de-composition methodology described above even to such systems. In fact, recent works which originally analyzed their security examining the entire protocol as a single unit have updated and simplified their proofs by utilizing the techniques presented in this paper [CK02,CLOS02].

1.1 The New Composition Theorem

We provide a very informal overview of the new composition theorem and its usage. Our system consists of a "high-level protocol", π, that uses multiple instances of a sub-protocol ρ, where the various instances of ρ have some joint state.

To be able to use the JUC theorem, we need to have in hand a protocol, $\hat{\rho}$, where a single instance of $\hat{\rho}$ has essentially the same functionality as multiple independent copies of ρ. We then proceed as follows. We first analyze the overall

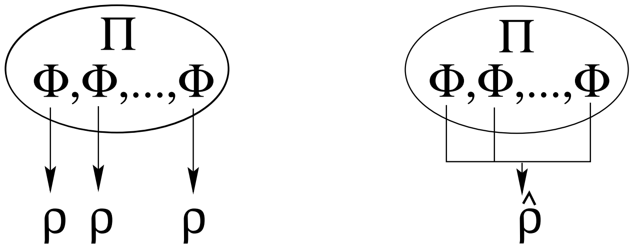

Fig. 1. Universal Composition with Joint State: Protocol π is analyzed under the assumption that the copies of ρ are independent (left). This is in spite of the fact that all copies of ρ are replaced by a single copy of $\hat{\rho}$ (right).

protocol π under the assumption that the copies of ρ are independent. (This can be done using known composition theorems, such as the UC theorem.) We then replace all instances of ρ within π with a single instance of $\hat{\rho}$. The JUC Theorem essentially states that protocol π behaves the same regardless of whether it is using multiple independent copies of ρ, or alternatively a single copy of $\hat{\rho}$.

The JUC theorem proves to be instrumental in de-composing complex systems. Using the terminology of the previous paragraph, it allows us to decompose our system into a "π part" plus a "$\hat{\rho}$ part," analyze each part in isolation, and then deduce the security of the re-composed system. The important thing to notice is that the "π part" treats all the copies of ρ as if they were independent copies without any joint state. This proves to be very useful in cases where the "π part" by itself consists of multiple copies of some other, simpler protocol ϕ, where different copies of ϕ call different copies of ρ. We can now analyze each copy of ϕ as stand-alone, then compose all copies of ϕ into a single protocol π (say, using the UC Theorem), and then use the JUC Theorem to compose π with all the copies of ρ as described above, in spite of the fact that the copies of ρ (and consequently the copies of ϕ) have joint state (see Figure 1).

As said above, in order for the JUC Theorem to hold, protocol $\hat{\rho}$ must exhibit the same behavior as multiple "independent" instances of ρ. (A formalization of this intuitive requirement appears within.) Clearly, the protocol $\hat{\rho}$ that simply runs multiple independent copies of ρ would guarantee this "independence". However, such a protocol $\hat{\rho}$ would not be very interesting. The power of the JUC theorem is in the cases where protocol $\hat{\rho}$ is more efficient, and in particular makes meaningful use of joint state between copies of ρ. The rest of the introduction is dedicated to providing examples for the use of the JUC theorem.

1.2 Application to Protocols Using Digital Signatures

We exemplify the use of the JUC Theorem for de-composing systems where multiple protocol instances use the same instance of a signature scheme. Specifically, we concentrate on the prevalent example of key-exchange and secure channel protocols authenticated via digital signatures. Here, multiple parties run multiple instance of the key-exchange protocol in order to exchange multiple keys, while using the same public-key infrastructure which is the joint state. The exchanged keys are then used to establish secure communication channels. Previous analytical works of key-exchange and secure channel protocols treats the entire multi-party, multi-execution system as a single unit (see, e.g., [BR93,Sho99,CK01]).

Here we show how, using the JUC Theorem, one can de-compose the system to individual sessions, analyze each session independently of all others, and deduce the security of the entire multi-session system. We proceed as follows. Using the terminology of the previous subsection, the key-exchange protocol (for exchanging a single key) is denoted ϕ, and the multi-instance composition of ϕ is denoted π. We are given a protocol ρ which satisfies the security requirements from a digital signature scheme. The protocol π uses multiple calls to ρ, where different calls are made by different copies of ϕ within ρ. (We assume that each of these independent calls is assigned a unique identifier i. This assumption is central in our solution. See discussion about the identifiers in Section 2).

In order to be able to apply the JUC Theorem, we show how to construct a protocol $\hat{\rho}$ that behaves, within a single instance, like multiple independent copies of ρ. In fact, this can be done with essentially the complexity of a single instance of ρ. Protocol $\hat{\rho}$ runs a single copy of ρ. When the ith instance of ϕ invokes its instance of ρ to generate a signature on message m, protocol $\hat{\rho}$ uses its single instance of ρ to sign the message (i, m). Similarly, whenever asked to verify whether s is a signature of instance i on a message m, $\hat{\rho}$ uses its single instance of ρ to verify whether s is a signature on (i, m). We show that if ρ is a secure signature protocol then $\hat{\rho}$ satisfies the conditions required by the JUC Theorem (i.e., $\hat{\rho}$ behaves essentially like multiple independent copies of a signature scheme). This allows us to deduce the security of the entire, multi-session key exchange protocol even though we only analyzed the security of the single session protocol ϕ and the signature protocol $\hat{\rho}$.

The same methodology applies also to the treatment of secure session protocols. That is, it is possible to analyze the security of a single session protocol as stand alone, and then use the JUC theorem to deduce the security of the composite multi-session system — in spite of the fact that all sessions use the same instance of the signature scheme. Indeed, the updated version of [CK02] on key-exchange and secure session protocols has modified its presentation and analysis to utilize the JUC Theorem as proven here.

1.3 Application to Protocols in the Common Reference String Model

A similar phenomenon happens in the case of protocols in the common reference string (CRS) model, where all parties have access to a reference string taken from a predefined distribution. We often have protocols where multiple instances use the same short reference string. These instances may be run either by the same set of parties or by different sets. So far, the only known way to analyze such multi-instance systems was to directly analyze them as a single unit. Using universal composition with joint state, we show how one can de-compose a multi-instance system to individual instances, analyze each instance in isolation, and then deduce security of the entire multi-instance system — in spite of the fact that all the protocol instances use the same short reference string. We demonstrate several alternative ways to go about this de-composition.

The first and most general way to de-compose multi-instance systems in the CRS model proceeds as follows. We first recall that the CRS model can be captured as the model that provide the parties with access to an "idealized protocol" ρ that returns the same string to all parties, where the string is chosen from a predefined distribution. Let ϕ be a protocol that uses the CRS (i.e., ϕ invokes protocol ρ) and let π be some protocol that runs multiple instances of protocol ϕ. In order to prove the security of π, we proceed in four steps: (a) We prove the security of a single instance of ϕ using a single instance of ρ, in a stand-alone setting where no other protocol executions exist. (b) Using known composition theorems (e.g., the universal composition theorem of [Can01]), we deduce that the multi-instance protocol π is also secure. Here, however, protocol π uses multiple independent instances of ρ, which corresponds to having multiple independent copies of the reference string. (c) We construct a protocol, $\hat{\rho}$, that mimics the behavior of multiple independent copies of ρ, using only a single copy of ρ. In other words, $\hat{\rho}$ generates multiple "independent" copies of the reference string, given only a single copy of the string. (d) Using the JUC Theorem, we deduce that the entire composed protocol (consisting of the multi-instance protocol π, where all calls to all copies of ρ are replaced with calls to a single instance of protocol $\hat{\rho}$) is secure. We stress that here all the copies of ϕ within π use the same copy of protocol ρ, namely only a single instance of the reference string.

In order to complete the de-composition process, we need to come up with a protocol $\hat{\rho}$ that realizes multiple instances of ρ, given only a single instance of ρ. Our construction of protocol $\hat{\rho}$ is essentially the three-message coin-tossing-into-the-well protocol of Blum [Blu82], where the commitments are taken to be universally composable commitments e.g., those of [CF01,CLOS02,DN02].

However, while the above de-composition method is quite general, it is not completely satisfactory because of the need to run the additional interactive protocol $\hat{\rho}$. In particular, in the composed protocol each copy of ϕ is interactive — even if the original construction of ϕ is non-interactive. We would like to be able to carry out the de-composition paradigm without paying the price in communication rounds.

At a first glance, it may appear that the way to avoid adding rounds is to come up with better constructions of protocol $\hat{\rho}$, which would be non-interactive. However, we show that this is not possible. That is, we show that *any* protocol which realizes multiple "independent" copies of the CRS, given only a single copy of the CRS, must be interactive. Essentially, each party must send at least one message per each new instance of the reference string. Furthermore, this message must be essentially at least as long as the generated string.

Given the impossibility of a non-interactive solution for the general problem of generating multiple CRSs given a single short CRS, we turn to other, less general ways to de-compose multi-instance systems in the CRS model. Specifically, we describe how our methodology can be applied to Zero-Knowledge (ZK) protocols and commitment protocols in the CRS model. Let us first sketch how this works for ZK protocols. Here we let protocol ρ be a single-instance ZK protocol. That is, protocol ρ carries out a single ZK proof. Assume we have a protocol π that consists of multiple copies of some protocol ϕ, where each instance of ϕ uses (perhaps multiple) copies of ρ. We can now use the JUC theorem to replace all instances of ρ with a single instance of protocol $\hat{\rho}$ that realizes multiple ZK proofs within a single instance. Luckily, such protocols $\hat{\rho}$ exist, and use only a single short instance of the CRS for all instances of the ZK proof [CF01,DCO$^+$01]. In particular, the protocol of [DCO$^+$01] is non-interactive.

In the case of commitment protocols we follow the same steps, with the exception that protocol ρ is a commitment protocol for a single commitment-decommitment process. The "composite protocol" $\hat{\rho}$ now provides the functionality of multiple commitments and decommitments, while using only a single short instance of the CRS. Such protocols exist, e.g., those of [CF01,CLOS02,DN02].

We remark that our formalization and results for the CRS model play a central role in the updated proofs for the general construction in [CLOS02]. Earlier versions of the paper analyzed these constructions directly as multi-instance protocols and were considerably more cumbersome.

Organization. Section 2 reviews the notion of UC security and the UC theorem of [Can01]. Section 3 presents and proves the JUC Theorem. For lack of space, the paper contains only the application to protocols in the CRS model in Section 4. The application to protocols that use signature schemes appears in [CR03].

2 Review of the Universal Composition Theorem

We provide a brief review of the universally composable security framework [Can01]. The framework allows for defining the security properties of cryptographic tasks so that the security of protocols is maintained under a general composition operation with an unbounded number of instances of arbitrary protocols running concurrently in the system. This composition operation is called universal composition. Similarly, definitions of security in this framework are called universally composable (UC).

As in other general definitions (e.g., [GL90,MR91,Bea91,Can00,PSW00]), the security requirements of a given task (i.e., the functionality expected from a protocol that carries out the task) are captured via a set of instructions for a "trusted party" that obtains the inputs of the participants and provides them with the desired outputs (in one or more iterations). Informally, a protocol securely carries out a given task if running the protocol with a realistic adversary amounts to "emulating" an ideal process where the parties hand their inputs to a trusted party with the appropriate functionality and obtain their outputs from it, without any other interaction. We call the algorithm run by the trusted party an ideal functionality.

In order to allow proving the universal composition theorem, the notion of emulation in this framework is considerably stronger than previous ones. Traditionally, the model of computation includes the parties running the protocol and an adversary, $\mathcal{A}$, that controls the communication channels and potentially corrupts parties. "Emulating an ideal process" means that for any adversary $\mathcal{A}$ there should exist an "ideal process adversary" (or, simulator) $\mathcal{S}$ that causes the *outputs* of the parties in the ideal process to have similar distribution to the outputs of the parties in an execution of the protocol. In the UC framework the requirement on $\mathcal{S}$ is more stringent. Specifically, an additional entity, called the environment $\mathcal{Z}$, is introduced. The environment generates the inputs to all parties, reads all outputs, and in addition interacts with the adversary in an arbitrary way throughout the computation. A protocol is said to securely realize a given ideal functionality $\mathcal{F}$ if for any "real-life" adversary $\mathcal{A}$ that interacts with the protocol and the environment there exists an "ideal-process adversary" $\mathcal{S}$, such that *no environment* $\mathcal{Z}$ can tell whether it is interacting with $\mathcal{A}$ and parties running the protocol, or with $\mathcal{S}$ and parties that interact with $\mathcal{F}$ in the ideal process. In a sense, here $\mathcal{Z}$ serves as an "interactive distinguisher" between a run of the protocol and the ideal process with access to $\mathcal{F}$.

The following *universal composition theorem* is proven in [Can01]. Consider a protocol π that operates in the $\mathcal{F}$-hybrid model, where parties can communicate as usual, and in addition have ideal access to an unbounded number of *copies* of an ideal functionality $\mathcal{F}$. Let ρ be a protocol that securely realizes $\mathcal{F}$ as sketched above, and let π^ρ be identical to π with the exception that the interaction with *each copy* of $\mathcal{F}$ is replaced with an interaction with a *separate instance* of ρ. Then, π and π^ρ have essentially the same input/output behavior. In particular, if π securely realizes some ideal functionality $\mathcal{I}$ in the $\mathcal{F}$-hybrid model then π^ρ securely realizes $\mathcal{I}$ in the standard model (i.e. without ideal functionality).

On the Session Identifiers (SID's). Let us highlight one detail regarding the hybrid model that will become important in subsequent sections. Each copy of the ideal functionality $\mathcal{F}$ is assumed to have a unique identifier, called the session identifier (SID) of that copy. Each message sent to $\mathcal{F}$ in the hybrid model should contain a SID, and is then forwarded to the corresponding copy of $\mathcal{F}$. (If there is no such copy then a new one is invoked and given this SID.) Similarly, each message from a copy of $\mathcal{F}$ to a party contains the SID of that copy. The SIDs are determined by the protocol running in the hybrid model. Notice that this formalization allows each copy of $\mathcal{F}$, and each instance of the protocol that later replaces this copy, to know its own SID. It also guarantees that no two copies of $\mathcal{F}$ can ever have the same SID. This is essential for the composition theorem to hold. (See discussion in the Introduction.) Let $\mathcal{F}(sid)$denote the copy of functionality $\mathcal{F}$ with SID $=(sid)$.

We stress that the model does not specified how the parties learn, or "agree" on the SID. While we cannot always assume that a set of uncoordinated parties can agree on an SID, there are many interesting and important settings where this assumption is reasonable. As an example consider a network in which two-party protocols are being executed by a set of uncoordinated parties. In this case a unique SID can be easily chosen by the two parties A and B themselves, by choosing locally unique strings r_A and r_B, respectively, which they exchange. The session id is defined to be $A \circ B \circ r_A \circ r_B$. Here an honest party is guaranteed to have a unique SID, even if the other party is cheating. Another example is within a closed network of multiple users executing multiple protocols, where the protocols have some ordering and thus can be allotted unique SIDs.

Nonetheless, it should be stressed that in some cases the requirement for unique SIDs is imperative for providing security. For instance, in [LLR02] it is shown that Byzantine Agreement can not be reached in some specific settings unless unique identifiers are provided.

3 Universal Composition with Joint State (JUC)

Recall that the universal composition operation requires replacing each copy of $\mathcal{F}$ with a different invocation of protocol ρ, where all the invocations of ρ in π^ρ must have disjoint states and independent random inputs. However, in our setting the copies of ρ have joint state. In fact, we wish to replace all the invocations of $\mathcal{F}$ with a *single* instance of some "joint protocol" $\hat{\rho}$. In order to do that, we essentially require that $\hat{\rho}$ has the same functionality as that of multiple independent copies of ρ. To formalize this requirement we define the multi-session extension of an ideal functionality. But first we define the composition operation.

The Composition Operation. The new composition operation, called universal composition with joint state (JUC), takes two protocols as arguments: a protocol π in the $\mathcal{F}$-hybrid model and a protocol $\hat{\rho}$ realizing the multi-session functionality. The result is a composed protocol, denoted $\pi^{[\hat{\rho}]}$, and described as follows. Essentially, universal composition with joint state is identical to universal composition, with two exception: First, each party P_i invokes only a single copy of $\hat{\rho}$ and replaces all calls to copies of $\mathcal{F}$ with activations of (the single copy of) $\hat{\rho}$. Second, now each activation of $\hat{\rho}$ includes *two* ids, the SID of $\hat{\rho}$ is set to some fixed, predefined value sid_0. The second id, SSID, is set to the original id of the invocation of $\mathcal{F}$. More specifically, protocol $\pi^{[\hat{\rho}]}$ behaves like π with the following changes.

1. *Modifications to the communication between $\pi^{[\hat{\rho}]}$ and $\hat{\rho}$ relative to the communication between π and $\mathcal{F}$:*
   (a) When activated for the first time within party P_i, $\pi^{[\hat{\rho}]}$ initiates a copy of protocol $\hat{\rho}$ with SID= sid_0.
   (b) Whenever π instructs party P_i to send a message (sid, v) to $\mathcal{F}(sid)$, protocol $\pi^{[\hat{\rho}]}$ instructs P_i to call $\hat{\rho}$ with input value (sid_0, sid, v).
   (c) When (the single copy of) $\hat{\rho}$ generates an output value (sid_0, sid, v) within $\pi^{[\hat{\rho}]}$, then $\pi^{[\hat{\rho}]}$ proceeds just as π proceeds upon receiving output message (sid, v) from $\mathcal{F}(sid)$.
2. *Operations required for communications between two parties in the system:*
   (a) Whenever protocol $\hat{\rho}$ wishes to send a message m, generated by the computation relating to (sid_0, sid), to some party P_j, then P_i writes the message (sid_0, sid, m) on its outgoing communication tape.
   (b) Upon delivery of a message (sid_0, sid, m) from P_j, party P_i activates $\hat{\rho}$ with incoming message (sid_0, sid, m).

The Multi-Session Extension of an Ideal Functionality. We formalize the security requirements from the "joint protocol" $\hat{\rho}$. Intuitively, the requirement is that it should have essentially the same functionality as multiple independent invocations of ρ. More formally, we define the following ideal functionality, $\hat{\mathcal{F}}$, which we want $\hat{\rho}$ to realize. Let $\mathcal{F}$ be an ideal functionality. (Intuitively, $\mathcal{F}$ is the functionality realized by a single instance of ρ.) According to the UC formalization, $\mathcal{F}$ expects each incoming message to contain a special field consisting of its session identifier (SID). All messages received by $\mathcal{F}$ are expected to have the same value of the SID. (Messages that have different session identifier than that of the first message are ignored.) Similarly, all outgoing messages generated by $\mathcal{F}$ carry the same SID.

The multi-session extension of $\mathcal{F}$, denoted $\hat{\mathcal{F}}$, is defined as follows. $\hat{\mathcal{F}}$ expects each incoming message to contain *two* special fields. The first is the usual session identifier field as in any ideal functionality. The second field is called the sub-session identifier (SSID) field. Upon receiving a message $(sid, ssid, v)$ (where sid is the SID, $ssid$ is the SSID, and v is an arbitrary value or list of values), $\hat{\mathcal{F}}$ first checks if there is a running copy of $\mathcal{F}$ whose (single) session identifier is $ssid$. If so, then $\hat{\mathcal{F}}$ activates that copy of $\mathcal{F}$ with incoming message $(ssid, v)$, and follows the instructions of this copy. Otherwise, a new copy of $\mathcal{F}$ is invoked (within $\hat{\mathcal{F}}$) and immediately activated with input $(ssid, v)$. From now on, this copy is associated with sub-session identifier $ssid$. (That is, $ssid$ is the session identifier of this copy of $\mathcal{F}$.) Whenever a copy of $\mathcal{F}$ sends a message $(ssid, v)$ to some party P_i, $\hat{\mathcal{F}}$ sends $(sid, ssid, v)$ to P_i, and sends $ssid$ to the adversary. Sending $ssid$ to the adversary implies that $\hat{\mathcal{F}}$ does not hide which copy of $\mathcal{F}$ is being activated within $\hat{\mathcal{F}}$.

It is stressed that $\hat{\mathcal{F}}$ is not explicitly used by π. It only serves as a criterion for the security of $\hat{\rho}$. Furthermore, while $\hat{\mathcal{F}}$ consists of several copies of $\mathcal{F}$ with disjoint states, there is no requirement that the protocol $\hat{\rho}$ that realizes $\hat{\mathcal{F}}$ would have such structure. Indeed, $\hat{\rho}$ may use some joint state for realizing all the copies of $\mathcal{F}$ within $\hat{\mathcal{F}}$. Clearly, the case where $\hat{\rho}$ uses some joint state is the case of interest for this work, as the other instance falls under the regular universal composition.

Theorem 1 (Universal Composition with Joint State). *Let $\mathcal{F}$ be an ideal functionality. Let π be a protocol in the $\mathcal{F}$-hybrid model, and let $\hat{\rho}$ be a protocol that securely realizes $\hat{\mathcal{F}}$, the multi-session extension of $\mathcal{F}$, in the real-life model. Then the composed protocol $\pi^{[\hat{\rho}]}$ in the real-life model emulates protocol π in the $\mathcal{F}$-hybrid model. That is, for any adversary $\mathcal{A}$ that interacts with parties running $\pi^{[\hat{\rho}]}$ in the real-life model, there exists an adversary $\mathcal{A}'$ that interacts with parties running π in the $\mathcal{F}$-hybrid model, such that no environment machine $\mathcal{Z}$ can tell whether it is interacting with $\pi^{[\hat{\rho}]}$ and $\mathcal{A}$, or alternatively with π and $\mathcal{A}'$.*

Proof. Let $\mathcal{F}, \pi, \hat{\rho}$ be as in the theorem statement. We show that $\pi^{[\hat{\rho}]}$ in the real-life model emulates protocol π in the $\mathcal{F}$-hybrid model. This is done in two steps: first we define a protocol $\tilde{\pi}$ and show that $\pi^{[\hat{\rho}]}$ in the real-life model emulates protocol $\tilde{\pi}$ in the $\hat{\mathcal{F}}$-hybrid model. Next we show that protocol $\tilde{\pi}$ in the $\hat{\mathcal{F}}$-hybrid model emulates protocol π in the $\mathcal{F}$-hybrid model.

Protocol $\tilde{\pi}$ is a slight variation of protocol π, operating in the $\hat{\mathcal{F}}$-hybrid model. Specifically, $\tilde{\pi}$ is identical to π, with the following exceptions. (Note that $\tilde{\pi}$ invokes only a single copy of $\hat{\mathcal{F}}$ throughout the computation.)

1. Whenever π instructs P_i to send a message (sid, v) to some copy $\mathcal{F}(sid)$ of $\mathcal{F}$, $\tilde{\pi}$ instructs P_i to send (sid_0, sid, v) to $\hat{\mathcal{F}}$.
2. Whenever some party P_i, running $\tilde{\pi}$, receives a message (sid_0, sid, v) from $\hat{\mathcal{F}}$, it follows the instructions of π upon receipt of the message (sid, v) from $\mathcal{F}(sid)$.

Recall that $\tilde{\pi}^{\hat{\rho}}$ is the protocol obtained by applying the universal composition operation of [Can01] to protocols $\tilde{\pi}$ and $\hat{\rho}$. Since protocol $\hat{\rho}$ securely realizes $\hat{\mathcal{F}}$, it follows from the universal composition theorem that protocol $\tilde{\pi}^{\hat{\rho}}$ in the real-life model emulates protocol $\tilde{\pi}$ in the $\hat{\mathcal{F}}$-hybrid model. Furthermore, it is easy to see that protocol $\tilde{\pi}^{\hat{\rho}}$ is identical to protocol $\pi^{[\hat{\rho}]}$. (These are two different descriptions of exactly the same protocol.) We thus have that protocol $\pi^{[\hat{\rho}]}$ in the real-life model emulates protocol $\tilde{\pi}$ in the $\hat{\mathcal{F}}$-hybrid model.

It remains to show that protocol $\tilde{\pi}$ in the $\hat{\mathcal{F}}$-hybrid model emulates protocol π in the $\mathcal{F}$-hybrid model. This is done as follows. Let $\hat{\mathcal{A}}$ be an adversary that interacts with parties running $\tilde{\pi}$ in the $\hat{\mathcal{F}}$-hybrid model. We construct an adversary $\mathcal{A}$ such that no environment will be able to tell whether it is interacting with $\mathcal{A}$ and π in the $\mathcal{F}$-hybrid model or with $\hat{\mathcal{A}}$ and $\tilde{\pi}$ in the $\hat{\mathcal{F}}$-hybrid model. Adversary $\mathcal{A}$ follows the instructions of $\hat{\mathcal{A}}$, with the following exceptions:

1. Whenever $\mathcal{A}$ is notified that some copy $\mathcal{F}(sid)$ of $\mathcal{F}$ has sent a message with identifier id to some party P_i, $\mathcal{A}$ records the pair (sid, id) and notifies $\hat{\mathcal{A}}$ that $\hat{\mathcal{F}}_{(sid_0)}$ has sent a message with identifier id to P_i. Note that $\mathcal{A}$ does not see the actual content of the message, but is only aware of the fact that a message has been sent.
2. Whenever $\hat{\mathcal{A}}$ delivers a message with identifier id from $\hat{\mathcal{F}}$ to P_i (in the $\hat{\mathcal{F}}$-hybrid model), $\mathcal{A}$ looks up the pair (sid, id) and delivers the message with identifier id from $\mathcal{F}(sid)$ to P_i.
3. When $\hat{\mathcal{A}}$ corrupts a party P_i (running $\tilde{\pi}$ in the $\hat{\mathcal{F}}$-hybrid model), $\mathcal{A}$ corrupts P_i (running π in the $\mathcal{F}$-hybrid model) and obtains the internal state of P_i for protocol π. It then 'translates' the internal state of P_i for protocol π to be consistent with an internal state of P_i for protocol $\tilde{\pi}$, and hand this information to $\hat{\mathcal{A}}$. (The translation is straightforward: calls to multiple copies of $\mathcal{F}$ are translated into calls to a single copy of $\hat{\mathcal{F}}$, with a fixed session identifier sid_0, and the corresponding SSIDs.)

It is straightforward to verify that the view of $\mathcal{Z}$ in an interaction with $\mathcal{A}$ and π in the $\mathcal{F}$-hybrid model is distributed identically to the view of $\mathcal{Z}$ in an interaction with $\hat{\mathcal{A}}$ and $\tilde{\pi}$ in the $\hat{\mathcal{F}}$-hybrid model.

We remark that if $\hat{\rho}$ operates in the $\mathcal{G}$-hybrid model for some ideal functionality $\mathcal{G}$, rather than in the real-life model, then $\pi^{[\hat{\rho}]}$ also operated in the $\mathcal{G}$-hybrid model. In this case, we have that $\pi^{[\hat{\rho}]}$ in the $\mathcal{G}$-hybrid model emulates protocol π in the $\mathcal{F}$-hybrid model.

4 Application to Protocols in the CRS Model

This section exemplifies the use of the JUC theorem for protocols in the common reference string model. There are two ways to approach this issue, the first is to take a single CRS and "stretch" it into multiple independent CRSs, which will later be utilized by protocols which require independent CRS's. The second is to examine a specific multi-session functionality which employs a CRS and to directly generate the multi-session functionality from the single CRS, and prove its security directly. While the first approach is more general, the second approach will result in more efficient protocols for specific functionalities.

Though the introduction starts with the first method, here we start with a specific example of commitments in the CRS model. This order of presentation is motivated by the fact that our protocol for stretching the CRS will need to employ a multi-session commitment functionality.

4.1 Commitment in the CRS Model

Let us first recall the ideal commitment functionality, $\mathcal{F}_{com}$, as defined in [CF01] (see Figure 2). Each copy of $\mathcal{F}_{com}$ handles the process of a single commitment followed by its single decommitment.

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

$\mathcal{F}_{com}$ proceeds as follows, running with parties $P_1, ..., P_n$ and an adversary $\mathcal{S}$.

1. Upon receiving a value $(\texttt{Commit}, sid, P_i, P_j, x)$ from P_i, record the value x and send the message $(\texttt{Receipt}, sid, P_i, P_j)$ to P_j and $\mathcal{S}$. Ignore any subsequent `Commit` messages.
2. Upon receiving a value $(\texttt{Open}, sid, P_i, P_j)$ from P_i, proceed as follows: If some value x was previously recoded, then send the message $(\texttt{Open}, sid, P_i, P_j, x)$ to P_j and $\mathcal{S}$ and halt. Otherwise halt.

Fig. 2. The Ideal Commitment functionality

The universal composition theorem allows us to write a protocol π in the $\mathcal{F}_{com}$-hybrid model, and then replace each copy of $\mathcal{F}_{com}$ with an instance of a protocol ρ that securely realizes $\mathcal{F}_{com}$. However, we only know how to realize $\mathcal{F}_{com}$ in the common reference string model, or in other words in the $\mathcal{F}_{crs}$-hybrid model, where $\mathcal{F}_{crs}$ is the ideal functionality that provides all parties with a common string taken from some pre-specified distribution. Consequently, if the universal composition theorem is used as is, then each instance of ρ in the composed protocol π^ρ would use a different copy of $\mathcal{F}_{crs}$ (i.e., an independent copy of the reference string).

Instead, we are looking for a protocol where all copies of the commitment protocol use the same short reference string. In [CF01], this is solved as follows. First, they define an additional ideal functionality, $\mathcal{F}_{mcom}$, where a single copy handles multiple commitments and decommitments by different parties and to different messages. Next, they construct a protocol, $\mathsf{UCC}_{\mathsf{ReUse}}$, that securely realizes $\mathcal{F}_{mcom}$ and uses a single copy of $\mathcal{F}_{crs}$ for all the commitments. Thus, if the high-level protocol π is written in the $\mathcal{F}_{mcom}$-hybrid rather than the $\mathcal{F}_{com}$-hybrid model, then the composed protocol, $\pi^{\mathsf{UCC}_{\mathsf{ReUse}}}$, uses a single copy of the reference string for multiple commitments.

However, having the high-level protocol π use $\mathcal{F}_{mcom}$ rather than $\mathcal{F}_{com}$ comes at a price in the analysis of the protocol. In order to guarantee that a single copy of $\mathcal{F}_{crs}$ is used throughout the computation, π has to use only a single copy of $\mathcal{F}_{mcom}$ (since each copy of $\mathcal{F}_{mcom}$ uses a different copy of $\mathcal{F}_{crs}$). This puts a considerable restriction on the analysis of π, as the security needs to be proven for π as a single unit. This holds even if π consists of multiple instances of some simpler protocol ρ. Thus, much of the advantage of using the UC theorem is lost.

These restrictions can be avoided using universal composition with joint state. We first observe that the functionality $\mathcal{F}_{mcom}$ is nothing but a reformulation of $\hat{\mathcal{F}}_{com}$. The JUC Theorem thus says that if π is a protocol in the $\mathcal{F}_{com}$-hybrid model (and uses as many copies of $\mathcal{F}_{com}$ as it wishes) then the composed protocol $\pi^{[\mathsf{UCC}_{\mathsf{ReUse}}]}$ runs in the $\mathcal{F}_{crs}$-hybrid model, emulates π, *and uses only a single copy of* $\mathcal{F}_{crs}$. In other words, we can allow protocol π (and all the higher level protocols that may use π as a subroutine) to operate in an idealized model where commitments are completely independent of each other, and then use the JUC Theorem to implement all the commitments using a single short common string.

4.2 Protocols for Stretching the Common Reference String

This section investigates the possibility of realizing $\hat{\mathcal{F}}_{crs}$ in the $\mathcal{F}_{crs}$-hybrid model. More specifically, we present a protocol ρ that securely realizes $\hat{\mathcal{F}}_{crs}$ in the $\mathcal{F}_{crs}$-hybrid model, and uses only a single copy of $\mathcal{F}_{crs}$. Protocol ρ allows the parties to generate multiple, computationally independent copies of the random string, using a single copy of the string. Then we can design protocols in the $\mathcal{F}_{crs}$-hybrid model, where each protocol instance can assume that no other instances have access to the reference string it is using, and then replace all copies of $\mathcal{F}_{crs}$ with a single copy of the reference string.

For simplicity we consider only the case where the reference string is taken from the uniform distribution over $\{0,1\}^t$ for some t. Also, we restrict attention to protocols where only *two* parties need to have access to the reference string. It seems that this special case captures much of the essence of the problem.

We first show a simple protocol that securely realizes $\hat{\mathcal{F}}_{crs}$ (for two parties and with uniform distribution) using a single copy of $\mathcal{F}_{crs}$. The protocol requires interaction between the two parties in order to generate each new copy of the reference string. We then demonstrate that *any* protocol that realizes $\hat{\mathcal{F}}_{crs}$ must involve sending at least one message by each of the participants. Furthermore, a new message must be sent for obtaining essentially any new copy of the reference string.

Let us first formulate functionality $\mathcal{F}_{crs}$ for the restricted case described above (Fig. 3). The functionality is parameterized by t, the length of the reference string. Note that $\mathcal{F}_{crs}$ does not limit the identities of the parties that may obtain the common random value r. The number (and identities) or parties that actually obtain the string is determined by the protocol that realizes $\mathcal{F}_{crs}$.

Functionality $\mathcal{F}^t_{crs}$

$\mathcal{F}_{crs}$ proceeds as follows, running with parties $P_1, ..., P_n$, and an adversary $\mathcal{S}$, parameterized by an integer t.

1. When receiving ($\texttt{CRS}$,sid, P_i, P_j) from P_i, choose a value $r \stackrel{\text{R}}{\leftarrow} \{0,1\}^t$, send ($\texttt{CRS}$,$sid, r$) to P_i, and send (sid, P_i, P_j, r) to $\mathcal{S}$. Next, when receiving ($\texttt{CRS}$,sid, P_i, P_j) from P_j (and only P_j), send ($\texttt{CRS}$,sid, r) to P_j and $\mathcal{S}$, and halt.

Fig. 3. The Common Random String functionality

Realizing $\hat{\mathcal{F}}_{crs}$. We show how to realize $\hat{\mathcal{F}}_{crs}$, using a single copy of $\mathcal{F}_{crs}$, for the case where only two parties obtain each copy of the reference string. The protocol is essentially the coin-tossing-into-the-well protocol (*ct*) of Blum [Blu82], which employs a commitment scheme. We use a universally composable commitment that utilize the single common random string. We first present the protocol in the $\mathcal{F}_{com}$-hybrid model, and then use the JUC Theorem to compose this protocol with protocol $\mathsf{UCC}_{\mathsf{ReUse}}$ of [CF01] (as discussed in Section 4.1), and obtain the desired protocol. This means that the distribution needed for our protocol is the distribution needed for $\mathsf{UCC}_{\mathsf{ReUse}}$. The protocol in the $\mathcal{F}_{com}$-hybrid model is denoted ct^t. Whenever parties P_i and P_j are invoked to generate a new copy of the reference string (say, with SID *sid*), they proceed as follows.

Message 1: When activated with input $\texttt{CRS}$,sid, P_i, P_j), P_i chooses a random string $r_i \stackrel{\text{R}}{\leftarrow} \{0,1\}^t$ and commits to r_i for P_j, using a new copy of $\mathcal{F}_{com}$ with SID *sid*. (That is, P_i sends $(\texttt{Commit}, sid, P_i, P_j, r_i)$ to $\mathcal{F}_{com}$).

Message 2: When activated with input $\texttt{CRS}$,sid, P_i, P_j), P_j waits to receive $(\texttt{Receipt}, sid, P_i, P_j)$ from $\mathcal{F}_{com}$. Next, P_j chooses $r_j \stackrel{\text{R}}{\leftarrow} \{0,1\}^t$ and sends to P_i.

Message 3: Upon receiving r_j, P_i decommits to r_i. (That is, P_i sends $(\texttt{Open}, sid, P_i, P_j)$ to $\mathcal{F}_{com}$).

Output: Both parties output the string $r_i \oplus r_j$.

Claim. Protocol ct^t securely realizes $\hat{\mathcal{F}}^t_{crs}$ in the $\mathcal{F}_{com}$-hybrid model.

Proof. Let $\mathcal{A}$ be an adversary that interacts with protocol ct^t in the $\mathcal{F}_{com}$-hybrid model. We construct an adversary $\mathcal{S}$ so that no environment can tell whether it is interacting with $\mathcal{S}$ in the ideal process for $\hat{\mathcal{F}}^t_{crs}$ or with $\mathcal{A}$ and *ct* in the $\mathcal{F}_{com}$-hybrid model. Adversary $\mathcal{S}$ runs a simulated copy of $\mathcal{A}$, and proceeds as follows. (Without loss of generality we assume that $\mathcal{A}$ does not run the protocol between two corrupted parties.)

1. Throughout, whenever $\mathcal{S}$ receives an input value from $\mathcal{Z}$, it copies this value to $\mathcal{A}$'s input tape. Whenever $\mathcal{A}$ writes a value on its output tape, $\mathcal{S}$ copies this value to its own output tape (to be read by $\mathcal{Z}$).
2. **Corrupted initiator:** If the simulated $\mathcal{A}$ generates a message $(\texttt{Commit}, sid, P_i, P_j, r_i)$ from a corrupted party P_i to $\mathcal{F}_{com}$, then $\mathcal{S}$ records r_i, and sends a message $(\texttt{CRS}, 0, sid, P_i, P_j)$ from P_i to $\hat{\mathcal{F}}_{crs}$ in the ideal process. (That is, the SID of this message is 0, and the SSID is *sid*.) Upon receiving a value r from $\hat{\mathcal{F}}_{crs}$, $\mathcal{S}$ waits to receive a message $(\texttt{CRS},(sid, P_j)$ from $\hat{\mathcal{F}}_{crs}$. This message will notify $\mathcal{S}$ that P_j was activated with input $(\texttt{CRS}, 0, sid, r)$. Upon receiving this message, $\mathcal{S}$ sets $r_j = r \oplus r_i$ and activates $\mathcal{A}$ to receive the message r_j from P_j. Next, when $\mathcal{A}$ generates the message $(\texttt{Open}, sid, P_i, P_j)$ from P_i to $\mathcal{F}_{com}$, $\mathcal{S}$ delivers the message that $\hat{\mathcal{F}}_{crs}$ sent to P_j. (This message contains the value r.)
3. **Corrupted responder:** If $\mathcal{S}$ receives the message $(0, sid, P_i, P_j, r)$ from $\hat{\mathcal{F}}_{crs}$ in the ideal process where P_j is corrupted, then $\mathcal{S}$ activates the simulated $\mathcal{A}$ to receive message $(\texttt{Receipt}, sid, P_i, P_j)$ from $\mathcal{F}_{com}$. When $\mathcal{A}$ generates a message r_j from P_j, $\mathcal{S}$ delivers the message from $\hat{\mathcal{F}}_{crs}$ to P_i in the ideal process, and activates $\mathcal{A}$ to receive the message $(\texttt{Open}, sid, P_i, P_j, r_i)$ from $\mathcal{F}_{com}$, where $r_i = r \oplus r_j$.
4. **Both parties uncorrupted:** If $\mathcal{S}$ receives the message $(0, sid, P_i, P_j, r)$ from $\hat{\mathcal{F}}_{crs}$ in the ideal process where both P_i and P_j are uncorrupted, then it simulates for $\mathcal{A}$ the information it sees when two uncorrupted parties run the protocol and obtain a common string r. This information consists of a notice from $\mathcal{F}_{com}$ that P_i committed to a value to P_j, the random value r_j sent by P_j, and the opening of the commitment to r_i such that $r_i \oplus r_j = r$.
5. **Party corruption:** If at any point the simulated $\mathcal{A}$ corrupts a party P_i then $\mathcal{S}$ corrupts P_i in the ideal process, and provides $\mathcal{A}$ with the simulated internal state of P_i. (It is easy to verify that this state is always implied by the information already known to $\mathcal{S}$ at the time of corruption.)

Let $\mathcal{Z}$ be an environment machine. It is straightforward to verify that the view of $\mathcal{Z}$ in the ideal process for $\hat{\mathcal{F}}_{crs}$ with $\mathcal{S}$ is distributed identically to its view of an execution of ct in the $\mathcal{F}_{com}$-hybrid model. Note that this holds even if $\mathcal{Z}$ is computationally unbounded.

Using the JUC Theorem, we have that protocol $ct^{[\mathsf{UCC}_{\mathsf{ReUse}}]}$ securely realizes $\hat{\mathcal{F}}_{crs}$ and uses only a single copy of $\mathcal{F}_{crs}$.

Limits on Protocols for Realizing $\hat{\mathcal{F}}_{crs}$. We show that any protocol that realizes $\hat{\mathcal{F}}_{crs}$ in the $\mathcal{F}_{crs}$-hybrid model, and uses only a few copies of $\mathcal{F}_{crs}$, must involve interaction. More specifically, we consider protocols that realize $\mathcal{F}^s_{crs}$ using a single copy of $\mathcal{F}^t_{crs}$, where $s > t$. (This is a considerable restriction of the original problem: Clearly, any protocol that securely realizes $\hat{\mathcal{F}}_{crs}$ realizes also

$\mathcal{F}_{crs}^{s}$ for any s that is polynomial in t.) We show that any such protocol must require that each of the parties send at least $s-t$ bits of information. Translated back to the task of realizing $\hat{\mathcal{F}}_{crs}^{t}$ using a single copy of $\mathcal{F}_{crs}^{t}$, this bound implies that each party must send at least t bits in order to generate each new copy of the string. (Note that the bound holds also for protocols generating a string among more than two parties.) That is:

Claim. Let π be a protocol that securely realizes $\mathcal{F}_{crs}^{s}$ using a single copy of $\mathcal{F}_{crs}^{t}$, and assume that π requires one of the parties to send no more than u bits of information. Then $s \leq t+u$.

Proof. Let π, s, t, u be as in the premise of the claim. It follows that for any adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish between an interaction with π and $\mathcal{A}$ in the $\mathcal{F}_{crs}^{t}$-hybrid model and an interaction with $\mathcal{S}$ in the ideal process for $\mathcal{F}_{crs}^{s}$. Let P_i be the party that sends at most u bits.

Consider the following adversary $\mathcal{A}$ and environment $\mathcal{Z}$. $\mathcal{Z}$ instructs $\mathcal{A}$ to corrupt all parties except for P_i, uniformly chooses random inputs $\alpha_1, ..., \alpha_n$ for the corrupted parties, and instructs $\mathcal{A}$ to have each corrupted party P_j run π with random input α_j. $\mathcal{A}$ follows the instructions of $\mathcal{Z}$ and reports the gathered information to $\mathcal{Z}$. This information consists of the t-bit value r_t obtained from $\mathcal{F}_{crs}^{t}$, and the u-bit concatenation, m, of all messages received from P_i. Next, $\mathcal{Z}$ obtains the s-bit output of P_i, r_s, picks a corrupted party P_j, and outputs 1 iff P_j outputs r_s after running π on random input α_j, and having received r_t from $\mathcal{F}_{crs}^{t}$ and messages m from P_i.

Clearly, if $\mathcal{Z}$ interacts with parties running π in the $\mathcal{F}_{crs}^{t}$-hybrid model, then it always outputs 1. On the other hand, let $\mathcal{S}$ be an ideal-process adversary. We claim that if $\mathcal{Z}$ interacts with $\mathcal{S}$ in the ideal process for $\mathcal{F}_{crs}^{s}$ then $\mathcal{Z}$ outputs 1 with probability at most 2^{t+u-s}. To see this, fix a value of $\boldsymbol{\alpha}$, and recall that $r_s \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^s$ is chosen by $\mathcal{F}_{crs}^{s}$, and that the output of P_j is uniquely determined by r_t, $\boldsymbol{\alpha}$, and m. Then, the probability over the choice of r_s that there *exist* a u-bit value m and a t-bit value r_t such that the output of P_j is r_s is at most 2^{t+u-s}. Note that the claim holds even if $\mathcal{Z}$ and $\mathcal{A}$ are restricted to polynomial time and even if P_i and $\mathcal{S}$ are unbounded.

4.3 Zero-knowledge in the CRS Model.

We describe the use of the JUC theorem to Zero-Knowledge protocols in the CRS model. The use is very similar to the case of commitment. Let us first recall the zero-knowledge functionality, $\mathcal{F}_{zk}$, as defined in [Can01], (see Fig. 4).

As in the case of $\mathcal{F}_{com}$, we only know how to realize functionality $\mathcal{F}_{zk}$ in the CRS model (i.e., in the $\mathcal{F}_{crs}$-hybrid model). Also here, straightforward composition of a protocol π in the $\mathcal{F}_{zk}$-hybrid model with a protocol ρ that securely realizes $\mathcal{F}_{zk}$ in the $\mathcal{F}_{crs}$-hybrid model would result in a composed protocol π^{ρ} that is highly wasteful of the reference string. We solve the problem by using universal composition with joint state. That is, given a protocol $\hat{\rho}$ that securely

Functionality $\mathcal{F}_{zk}^R$

$\mathcal{F}_{zk}$ proceeds as follows, running with parties $P_1, ..., P_n$ and an adversary $\mathcal{S}$, given a binary relation R.

1. Upon receipt of a value $(\texttt{prover}, sid, P_i, P_j, x, w)$ from some party P_i, Send $(sid, P_i, x, R(x, w))$ to P_j and $\mathcal{S}$, and halt.

Fig. 4. The Zero-Knowledge functionality, $\mathcal{F}_{zk}$

realizes $\hat{\mathcal{F}}_{zk}$, the multi-session extension of $\mathcal{F}_{zk}$, we conclude that the composed protocol $\pi^{[\hat{\rho}]}$ runs in the $\mathcal{F}_{crs}$-hybrid model and emulates π. Furthermore, if $\hat{\rho}$ uses only few copies of $\mathcal{F}_{crs}$ then so does $\pi^{[\hat{\rho}]}$.

We complete the discussion by pointing to two protocols that securely realize $\hat{\mathcal{F}}_{zk}$ in the $\mathcal{F}_{crs}$-hybrid model, and use only a single copy of $\mathcal{F}_{crs}$. First, recall protocol hc in [CF01] that securely realizes $\mathcal{F}_{zk}$ in the $\mathcal{F}_{com}$-hybrid model. (We remark that the formalization of $\mathcal{F}_{zk}$ in [CF01] is slightly different than the one here. Nonetheless, it is easy to see that the two formalizations are equivalent.) We claim that this protocol in effect realizes also $\hat{\mathcal{F}}_{zk}$. (Simply run the protocol separately for each interaction.) Thus, using the JUC Theorem, we obtain that the composed protocol, $hc^{[\mathsf{UCC}_{\mathsf{ReUse}}]}$, securely realizes $\hat{\mathcal{F}}_{zk}$ in the $\mathcal{F}_{crs}$-hybrid model. That is, we have:

Claim. Protocol $hc^{[\mathsf{UCC}_{\mathsf{ReUse}}]}$ securely realizes $\hat{\mathcal{F}}_{zk}$ in the $\mathcal{F}_{crs}$-hybrid model. Furthermore, it uses only a single copy of $\mathcal{F}_{crs}$.

Next, we note that the simulation-sound non-interactive zero-knowledge proof of knowledge protocol of De-Santis et al. [DCO+01] can be written as a protocol that securely realizes $\hat{\mathcal{F}}_{zk}$ in the $\mathcal{F}_{crs}$-hybrid model, with respect to non-adaptive adversaries:

Claim. The protocol of [DCO+01]securely realizes $\hat{\mathcal{F}}_{zk}$ in the $\mathcal{F}_{crs}$-hybrid model, with respect to non-adaptive adversaries. Furthermore, it uses only a single copy of $\mathcal{F}_{crs}$.

References

[Bea91] D. Beaver. Secure Multiparty Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority. *Journal of Cryptology*, 4:75–122, 1991.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Noncryptographic Fault-Tolerant Distributed Computations. In *Proc. 20th STOC*, pages 1–10. ACM, 1988.

[Blu82] M. Blum. Coin Flipping by Telephone . In *IEEE Spring COMPCOM*, pages 133–137, 1982.

[BR93] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Crypto '93*, pages 232–249, 1993. LNCS No. 773.

[Can00] R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[Can01] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proc. 42st FOCS*, pages 136–145. IEEE, 2001. http://eprint.iacr.org/2000/067.

[CF01] R. Canetti and M. Fischlin. Universally Composable Commitments. In *Crypto '01*, pages 19–40. LNCS No. 2139.

[CK01] R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels . In *Eurocrypt '01*, pages 453–474, 2001. LNCS No. 2045.

[CK02] R. Canetti and H. Krawczyk. Universally Composable Key Exchange and Secure Channels . In *Eurocrypt '02*, pages 337–351, 2002. LNCS No. 2332.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *Proc. 34th STOC*, pages 494–503.

[CR03] R. Canetti and T. Rabin. Universal Composition with Joint State. Available online, http://eprint.iacr.org.

[DCO+01] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust Non-interactive Zero-Knowledge. In *Crypto '01* LNCS No. 2139.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Non-malleable Cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[DM00] Y. Dodis and S. Micali. Parallel Reducibility for Information-Theoretically Secure Computation. In *Crypto '00*, pages 74–92, 2000. LNCS No. 1880.

[DN02] I. Damgard and J. Nielsen. Universally Composable Commitment Schemes with Constant Expansion Factor. In *Crypto '02* LNCS No. 2442.

[DNS98] C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *Proc. 30th STOC*, pages 409–418.

[GK96] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM. J. Computing*, 25(1):169–192, 1996.

[GL90] S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In *Crypto '90*, 1990. LNCS No. 537.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to Play Any Mental Game. In *Proc. 19th STOC*, pages 218–229. ACM, 1987.

[GO94] O. Goldreich and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. *Journal of Cryptology*, 7(1):1–32, 1994. Preliminary version by Y. Oren in FOCS87.

[Gol02] O. Goldreich. Concurrent Zero-Knowledge With Timing Revisited. In *Proc. 34th STOC*.

[LLR02] Y. Lindell, A. Lysyanskya, and T. Rabin. On the Composition of Authenticated Byzantine Agreement. In *Proc. 34th STOC*, pages 514–523.

[MR91] S. Micali and P. Rogaway. Secure Computation. In *Crypto '91*, pages 392–404, 1991. Manuscript available.

[PSW00] B. Pfitzmann, M. Schunter, and M. Waidner. Secure Reactive Systems. IBM Research Report RZ 3206 (#93252), IBM Research, Zurich, May 2000.

[Sho99] V. Shoup. On Formal Models for Secure Key Exchange. Available at: http://www.shoup.org, 1999.