The IITM Model: a Simple and Expressive Model for Universal Composability

Ralf Küsters [*] Max Tuengerthal [+] Daniel Rausch [*]

[*] University of Stuttgart, Germany

{ralf.kuesters,daniel.rausch}@sec.uni-stuttgart.de

[+] Siemens Mobility

max.tuengerthal@siemens.com

September 26, 2019

Abstract

The universal composability paradigm allows for the modular design and analysis of cryptographic protocols. It has been widely and successfully used in cryptography. However, devising a coherent yet simple and expressive model for universal composability is, as the history of such models shows, highly non-trivial. For example, several partly severe problems have been pointed out in the literature for the UC model.

In this work, we propose a coherent model for universal composability, called the IITM model ("Inexhaustible Interactive Turing Machine"). A main feature of the model is that it is stated without a priori fixing irrelevant details, such as a specific way of addressing of machines by session and party identifiers, a specific modeling of corruption, or a specific protocol hierarchy. In addition, we employ a very general notion of runtime. All reasonable protocols and ideal functionalities should be expressible based on this notion in a direct and natural way, and without tweaks, such as (artificial) padding of messages or (artificially) adding extra messages.

Not least because of these features, the model is simple and expressive. Also the general results that we prove, such as composition theorems, hold independently of how such details are fixed for concrete applications.

Being inspired by other models for universal composability, in particular the UC model and because of the flexibility and expressivity of the IITM model, conceptually, results formulated in these models directly carry over to the IITM model.

Keywords: *Universal Composability, Protocol Analysis, Foundations*

Contents

1 Introduction

In the universal composability paradigm [5,36] the security of protocols is defined in such a way that security is preserved even if the protocols are used as components of an arbitrary (polynomially bounded) distributed system. This strong composability property allows for the modular design and analysis of protocols. More specifically, the security of a protocol is defined in terms of an ideal protocol (also called an ideal functionality). A real protocol securely realizes the ideal protocol if every attack on the real protocol can be translated into an "equivalent" attack on the ideal protocol, where equivalence is specified based on an environment trying to distinguish the real attack from the ideal one. That is, for every real adversary on the real protocol there must exist an ideal adversary (also called a simulator) on the ideal protocol such that no environment can distinguish whether it interacts with the real protocol and the real adversary or the ideal protocol and the ideal adversary. So the real protocol is as secure as the ideal protocol (which, by definition, is secure) in all environments.

At the core of the universal composability paradigm are composition theorems which say that if a protocol uses one or more (independent) instances of an ideal functionality, then all instances of the ideal functionality can be replaced by instances of the real protocol that realizes the ideal functionality. In this way, more and more complex protocols can be designed and analyzed in a modular way based on ideal functionalities. Often different protocol instances share some state, such as long-term keys. Consider, for example, an ideal functionality for public-key encryption. Such a functionality typically encapsulates one public-key (and its corresponding private key), say pk, and models ideal encryption under pk. That is, when this functionality is used to encryption a message m (under pk), then the resulting ciphertext does not contain any information about m, except the length of m. Now, consider a protocol, for example, a key exchange protocol, that uses this ideal functionality. Then, in the universal composability paradigm, in different sessions of the protocol different independent instances of the ideal public-key encryption functionality would be used. But this means that different sessions of the protocol would use different public-keys. This is of course impractical and unrealistic. One would rather like to use the same public-key (and hence, the same private key) in all protocol sessions. Fortunately, so-called joint state theorems allow one to argue about protocols in a modular way, similar to the general composition theorem mentioned above, even if several instances of a protocol/functionality share some state [14,26,30] (see also Section 10.2 for more details). Moreover, in some cases it is necessary that all components of a system have access to some global state information, such as a common reference string (CRS). For this purpose, composition theorems with global state (also called global setup) have been proposed [9] (see also Section 10.3).

The universal composition paradigm has been widely and successfully used in cryptography to design and analyze complex protocols in a modular way (see, e.g., [6] for an overview).

However, devising a coherent yet simple and expressive model for this paradigm has turned out to be highly non-trivial (see Section 11 for a more detailed discussion). For example, several partly severe problems have been pointed out in the literature for the UC model [5], concerning, for instance, the validity of the composition and joint state theorems, the way corruption is handled, the notion of runtime, and the expressivity of the model.

Contribution. In this paper, we propose a coherent model for universal composability that is both simple and expressive. Our model coincides with the IITM model proposed in previous work [23] (where IITM stands for "Inexhaustible Interactive Turing Machine"), except that we now use a more general notion of runtime than the one used in the original version of the IITM model, namely one that is based on a runtime notion proposed by Hofheinz et al. in [21]. We therefore stick to the name IITM model also for the model proposed here. The main features of the IITM model are as follows.

- The IITM model is very simple and expressive, not least because it is formulated in a very general way without fixing irrelevant details:
  - The IITM model provides a very flexible and generic mechanism to address instances of machines. Unlike other models, in which a specific way of addressing of machines by session identifiers

(SIDs) and party identifiers (PIDs) is fixed, the IITM model does not hard-wire how machines are addressed.

  - Unlike other models, the IITM model also does not hard-wire corruption into the model. Corruption instead can be specified in a very general and flexible way as part of the protocol specification.
  - Unlike other models, the IITM model does not impose a specific structure on protocols, such as a hierarchical structure with protocols and subroutines. This is, for example, important for seamlessly dealing with joint and global state, and for faithfully modeling real-world protocols.
  - The runtime of machines and systems of machines is defined in a very general way. All reasonable (real and ideal) protocols should be expressible in a very natural way based on this runtime notion, without tweaks, such as (artificial) padding of messages or (artificially) adding extra messages, as necessary in other models.

- All common universal composability security notions, including (dummy) UC, strong simulatability, black-box simulatability, and reactive simulatability are equivalent in the IITM model.[1]

- The composition, joint state, and global setup theorems are very general: the class of protocols they cover is large, both in terms of the structure and the runtime of protocols. These theorems are stated and proven independently of many details fixed in other models (such as addressing of machines and corruption). Hence, they hold true no matter how these details are fixed in concrete applications. The generality of the theorems and the IITM model is also apparent in the fact that to state the joint state and global setup theorems the IITM does not have to be changed or extended, unlike other models. These theorems can smoothly be stated and proven within the model. Moreover, the joint state theorem and the main global setup theorems are even merely direct consequences of our general composition theorems. In other models, new notation is required to state those theorems and the theorems require (non-trivial) proofs. The flexibility of the IITM model and the generality of its composition theorems also allow us to directly support many forms of joint state and global setup, including arbitrary combinations of both, which have not been considered in the literature so far and would require extensions in other models (cf. Sections 10 and 11 for more details and discussion).

- Since the IITM model conceptually follows other models for universal composability, in particular the UC model, and because of its high expressivity and flexibility, results established in other models easily carry over to the IITM model.

Structure of the paper. In Section 2, we present the IITM model in a nutshell in order to provide a first impression of the model. We then, in Section 3, define the computational model on which the IITM model is based. This model is stated independently of the application to universal composability and is of independent interest. The runtime notions that we use are introduced in Section 4, along with basic properties. While these properties are as expected, some of the problems in other models stem from the fact that not all of these properties are satisfied in these models. For example, not in all models it is possible to specify a dummy machine which simply forwards messages back and forth between different system components. Also, in some models it is not possible to simulate arbitrary subsystems by one machine. In Section 5, we present general composition theorems for systems that are computationally indistinguishable by environments. These composition theorems are the core of the composition theorems for universal composition, presented in Section 7. The common notions for universal composability, namely (dummy) UC, strong simulatability, black-box simulatability, and reactive simulatability are introduced in Section 6, along with basic properties. We show that all these notions are equivalent, where, as already mentioned, for reactive simulatability this requires environments with external input. The general composition theorems are then, as mentioned, stated and proven in Section 7. They follow easily from those established in Section 5. The composability of the runtime notions that we use is discussed in Section 8. An alternative runtime notion and why it is unsuitable

[1] For reactive simulatability, this is true for environments with external input (non-uniform environments), as detailed in Section 6.3.

is explained in Section 9. In Section 10, we illustrate how the IITM model can be used to design and analyze (multi-party) protocols in a modular way. In particular, we present one way of addressing multiple sessions and modeling corruption of machines/instances/parties. We also briefly discuss joint state and global state composition theorems. Related work is discussed in Section 11. Details omitted in the main body of the paper can be found in the appendix.

Acknowledgments. We would like to thank Michael Backes, Ran Canetti, Anupam Datta, Dennis Hofheinz, John Mitchell, Olivier Pereira, Birgit Pfitzmann, and Dominique Unruh for many interesting discussions on models for universal composability. This work was in part funded by the *Deutsche Forschungsgemeinschaft (DFG)* through Grant KU 1434/9-1.

2 The IITM Model in a Nutshell

In this section, we provide a brief introduction to the IITM model, with full details presented in the subsequent sections. In the IITM model, security notions and composition/joint state theorems are formalized based on a very simple and at the same time very expressive general computational model, in which IITMs (inexhaustible interactive Turing machines) and systems of IITMs are defined. We first sketch the general computational model and then, based on it, formulate universal composability security notions and state composition theorems.

The General Computational Model. The general computational model is defined in terms of systems of IITMs. An *inexhaustible interactive Turing machine (IITM)* is a probabilistic Turing machine with named input and output tapes as well as an associated polynomial. The tape names determine how different machines are connected in a system of IITMs (see below). An IITM runs in one of two modes, CheckAddress and Compute. The CheckAddress mode is used as a generic mechanism for addressing instances of IITMs in a system of IITMs, as explained below. In this mode, an IITM may perform, in every activation, a deterministic polynomial time computation in the length of the security parameter plus the length of the current input plus the length of its current configuration, where the polynomial is the one associated with the IITM. The IITM is supposed to output "accept" or "reject" at the end of the computation in this mode, indicating whether the received message is processed further or ignored. The actual processing of the message, if accepted, is done in mode Compute. In mode Compute, a machine may output only at most one message on an output tape (and hence, only at most one other machine is triggered). The runtime in this mode is not a priori bounded. Later the runtime of systems and their subsystems will be defined in such a way that the overall runtime of a system of IITMs is polynomially bounded in the security parameter plus the length of the external input. We note that in both modes, an IITM cannot be exhausted (hence, the name): in every activation it can perform actions and cannot be forced to stop. This property, while not satisfied in all other models, is crucial to obtain a reasonable model for universal composability (see also Section 11).

A *system* $\mathcal{S}$ of IITMs is of the form $\mathcal{S} = M_1 \mid \cdots \mid M_k \mid !\,M'_1 \mid \cdots \mid !\,M'_{k'}$ where M_i, $i \in \{1, \ldots, k\}$, and M'_j, $j \in \{1, \ldots, k'\}$, are IITMs such that, for every tape name c, at most two of these IITMs have a tape named c and if two IITMs have a tape named c, then c is an input tape in one of the machines and an output tape in the other. We say that the IITMs M'_j are in the scope of a bang operator. This operator indicates that in a run of a system an unbounded number of (fresh) instances of a machine can be generated. Conversely, if a machine is not in the scope of a bang operator, there may be at most one instance of the machine in every run of the system. Systems in which multiple instances of a machine may be generated are often needed, e.g., in the case of multi-party protocols or in the case a system describes the concurrent execution of multiple instances of a protocol.

Before explaining runs of systems, we would like to emphasize the difference between a *description* of a machine and an *instance* (or *copy*) of a machine. The difference is the same as the one between program code and a process in an operating system: a process has state and performs the actual actions in a run of a system; it does so following its program code. In our setting, the description of a machine M specifies the behavior of a machine (its program code) and is part of the specification of a system $\mathcal{S}$. In a run of

$\mathcal{S}$, *instances* of M are created. These instances have a specific state (or configuration), receive input on their input tapes, process the input according to their specifications (program code), thereby updating their state, and produce output. In what follows, for simplicity, we often use the terms *IITMs* and *machines* to denote both static descriptions and instances, depending on the context. More specifically, in the context of systems we mean the static descriptions of machines. In the context of runs of systems or in the context of the runtime behavior of machines, we refer to instances of machines.

In a run of a system $\mathcal{S}$ at any time only one instance of an IITM is active and all other instances wait for new input. The first instance to be activated in a run of $\mathcal{S}$ is an instance of the so-called master IITM. A system has at most one master IITM, which may get external input; a run may have several instances of the master IITM, though, if this machine is in the scope of a bang operator (see below). By the definition of IITMs, the active machine may output only at most one message on one of its output tapes, and hence, at most one other machine is triggered after the activation of the currently active machine. To illustrate runs of systems, consider, for example, the system $\mathcal{S} = M_1 \,|\, !\, M_2$ and assume that M_1 has an output tape named c, M_2 has an input tape named c, and M_1 is the master IITM. (There may be other tapes connecting M_1 and M_2.) Furthermore, assume that in the run of $\mathcal{S}$ executed so far, two instances of M_2, say M_2' and M_2'', have been generated, with M_2' generated before M_2'', and that M_1 just sent a message m on tape c. This message is delivered to M_2' (as the first copy of M_2). First, M_2' runs in mode CheckAddress with input m; as mentioned, this is a deterministic polynomial time computation which outputs "accept" or "reject". If M_2' accepts m, then M_2' gets to process m in mode Compute and could, for example, send a message back to M_1. Otherwise, m is given to M_2'' which then runs in mode CheckAddress with input m. If M_2'' accepts m, then M_2'' gets to process m in mode Compute. Otherwise (if both M_2' and M_2'' do not accept m), a new copy M_2''' of M_2 with fresh randomness is generated and M_2''' runs in mode CheckAddress with input m. (Note that a new copy of M_2 is generated only because in the system description $\mathcal{S}$ M_2 is in the scope of a bang.) If M_2''' accepts m, then M_2''' gets to process m. Otherwise, M_2''' is removed again, the message m is dropped, and an instance of the master IITM is activated with empty input, in this case M_1. More precisely, (the single instance of) M_1 first runs in mode CheckAddress to determine whether it accepts the empty input. If M_1 does, it gets to process the empty input. Otherwise, no new instance of M_1 is generated, because M_1 is not in the scope of a bang operator. Now, since M_1 is a master IITM, this means that the run stops. In general, master IITMs can also be in the scope of a bang operator, and hence, analogously to M_2 in the example, new instances of such an IITM can be created. If none of the existing instances of a master IITM nor a newly created master instance accepts an input message, the system run stops. A run also stops if a master instance active in mode Compute does not produce output (and hence, does not trigger another machine). If a currently active non-master instance running in mode Compute does not produce output, a master instance is triggered (with empty input). In particular, this does not (immediately) terminate a run. Finally, a run also terminates if an instance of a machine (not necessarily a master instance) outputs a message on an output tape named decision.[2] Such a message is considered to be the *(overall) output of the system.*

As mentioned in the definition of systems, tape names of machines are unique in the context of the static description of a system: there are no two machines in a system which have an input tape with the same name; analogously for output tapes. For every tape name c, there may, however, exist at most two machines in a system where one has an input tape named c and the other has an output tape named c, which, as explained, means that the two machines are connected. For *instances* of machines, we do not have uniqueness of names, though. For example, both instances M_2' and M_2'' from above have an input tape named c as they are copies of M_2. Because (the static description of) M_1 has an output tape named c and (the static description of) M_2 has an input tape name c, (an instance of) M_1 can send a message via c to an instance of M_2. As explained, the CheckAddress mode of M_2 is used to determine which copy of M_2 actually gets to process the message sent by M_1. If M_1 did not have any output tape whose name coincides with the name of an input tape of M_2, then M_1 could not send a message to (an instance of) M_2.

Two systems $\mathcal{P}$ and $\mathcal{Q}$ are called *indistinguishable* ($\mathcal{P} \equiv \mathcal{Q}$) if and only if the difference between the probability that the output of $\mathcal{P}$ is 1 and the probability that the output of $\mathcal{Q}$ is 1 is negligible.

Note that we do not fix details such as addressing of machines by party/session IDs or corruption in

[2]Note that, by the definition of systems, only one of the specified IITMs can have such an output tape.

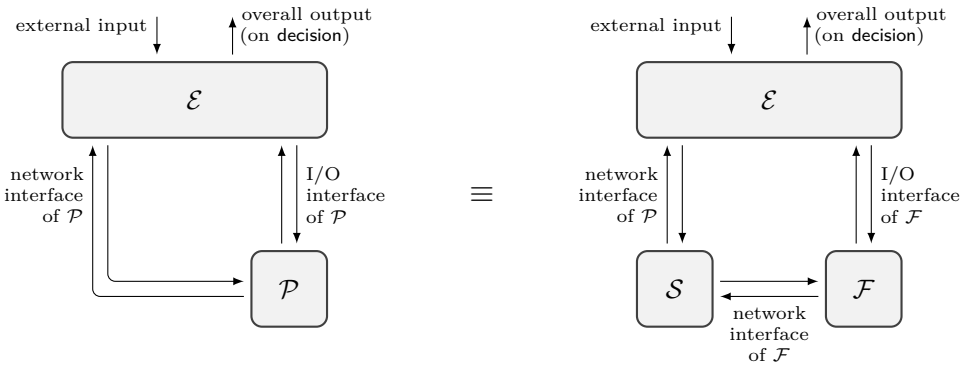

Figure 1: Strong simulatability (SS). We note that $\mathcal{P}$ and $\mathcal{F}$ have the same I/O interface.

this model. We also do not impose any specific structure, e.g., a hierarchical structure with protocols and subroutines, on systems. This makes the model both simpler and more expressive.

Universal Composability Security Notions. We need the following terminology. For a system $\mathcal{S}$, the input/output tapes of IITMs in $\mathcal{S}$ that do not have a matching output/input tape in $\mathcal{S}$ are called *external*, where an input/output tape of one IITM in $\mathcal{S}$ matches an output/input tape of another IITM in $\mathcal{S}$ if both tapes have the same name. External tapes are grouped into *I/O* and *network tapes*. I/O tapes are used to model secure direct connections between two machines, for example, to model subroutine relationships where one machines locally calls another one, whereas network tapes model untrusted communication with the adversary/environment or communication with a simulator (see below). We often refer to the sets of I/O and network tapes of $\mathcal{S}$ by *I/O and network interface*, respectively. We consider three different types of systems: protocol systems, adversarial systems, and environmental systems, modeling (i) real and ideal protocols/functionalities, (ii) adversaries and simulators, and (iii) environments, respectively. *Protocol systems*, *adversarial systems*, and *environmental systems* are systems which have an I/O and network interface, i.e., they may have I/O and network tapes. Adversarial systems may only connect to the network interface of a protocol system but not to its I/O interface. Environmental systems may contain a master machine and may produce output on the tape decision.

So far, we have not restricted the runtime of IITMs in mode Compute in any way. The following constraints will be used to enforce that systems run in polynomial time (possibly except with negligible probability): (i) Every environmental system $\mathcal{E}$ has to be *universally bounded*, i.e., there exists a polynomial p such that for all systems $\mathcal{S}$ which connect only to the external tapes of $\mathcal{E}$, we have that the overall runtime of $\mathcal{E}$ in mode Compute in runs of the system $\mathcal{E} \mid \mathcal{S}$, with security parameter η and external input a, is bounded by $p(\eta + |a|)$. (ii) A protocol system $\mathcal{P}$ typically[3] has to be *environmentally bounded*, i.e., for all environmental systems $\mathcal{E}$ there exists a polynomial p such that the overall runtime of $\mathcal{P}$ in mode Compute in runs of the system $\mathcal{E} \mid \mathcal{P}$, with security parameter η and external input a, is bounded by $p(\eta + |a|)$ (in all runs except for a negligible set of runs). Since the runtime in mode CheckAddress is polynomially bounded, this guarantees that, for a protocol system $\mathcal{P}$ and environmental system $\mathcal{E}$, the overall runtime of $\mathcal{E} \mid \mathcal{P}$ is polynomially bounded in the security parameter plus the length of the external input (except with negligible probability). This runtime notion for protocol systems is very general. We claim that it includes all reasonable protocol systems that occur in applications, as further explained in [21] and subsequent sections. In most other models, such as Canetti's UC model, the runtime notions are more restricted and more complex.

We now informally define strong simulatability; other equivalent security notions, such as universal composability (UC) and dummy UC, are defined in a similar way in subsequent sections. The systems considered in the following definition are illustrated in Figure 1.

[3]Our formal definition (see Definition 11) is more general. See also the remarks following Definition 11.

Definition 1 (informal)**.** Let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems with the same I/O interface (i.e., with the same set of I/O tapes), the real and the ideal protocol, respectively. Then, $\mathcal{P}$ *realizes* $\mathcal{F}$ ($\mathcal{P} \leq^{SS} \mathcal{F}$) if and only if there exists an adversarial system $\mathcal{S}$ (called a simulator or an ideal adversary) such that $\mathcal{S}$ connects only to the network interface of $\mathcal{F}$, the systems $\mathcal{P}$ and $\mathcal{S} \,|\, \mathcal{F}$ have the same external interface, $\mathcal{S} \,|\, \mathcal{F}$ is environmentally bounded, and for all environmental systems $\mathcal{E}$, connecting only to the external interface of $\mathcal{P}$ (and hence, $\mathcal{S} \,|\, \mathcal{F}$), it holds that $\mathcal{E} \,|\, \mathcal{P} \equiv \mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}$.

We note that this relation is reflexive and transitive. We also emphasize that still details such as addressing of machines by party/session IDs, corruption, and the structure of protocols are not, and do not need to be fixed in order to define this realization relation.

Composition Theorems. Composition theorems allow for the modular analysis and design of systems and are one of the main features of the universal composability paradigm. Our first composition theorem handles concurrent composition of a fixed number of (different) protocol systems. The second one guarantees secure composition of an unbounded number of instances of a protocol system.

Theorem 1 (informal)**.** *Let $k \geq 1$. Let $\mathcal{Q}, \mathcal{P}_1, \ldots, \mathcal{P}_k, \mathcal{F}_1, \ldots, \mathcal{F}_k$ be protocol systems such that they connect only via their I/O interfaces, $\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \cdots \,|\, \mathcal{P}_k$ is environmentally bounded, and $\mathcal{P}_i \leq^{SS} \mathcal{F}_i$, for $i \in \{1, \ldots, k\}$. Then, $\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \cdots \,|\, \mathcal{P}_k \leq^{SS} \mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \cdots \,|\, \mathcal{F}_k$.*

Note that this theorem does not require that the protocols $\mathcal{P}_i/\mathcal{F}_i$ are subprotocols of $\mathcal{Q}$, i.e., that $\mathcal{Q}$ has matching external I/O tapes for all of these protocols. How these protocols connect to each other via their I/O interfaces is not restricted in any way, even the environment could connect directly to the parts of the I/O interfaces of these protocols that are not taken by another protocol.

For the following composition theorem, we introduce the notion of a session version of a protocol in order to be able to address instances of the protocol. Given an IITM M, the *session version* $\underline{M}$ of M is an IITM which internally simulates M and acts as a "wrapper" for M. More precisely, in mode CheckAddress, (an instance of) $\underline{M}$ accepts an incoming message m' only if the following conditions are satisfied: (i) $\underline{M}$ has not accepted a message yet (in mode CheckAddress), m' is of the form (id, m), and m is accepted by the simulated M in mode CheckAddress. (In this case, later when activated in mode Compute, the ID id will be stored by $\underline{M}$.) (ii) $\underline{M}$ has accepted a message before, m' is of the form (id', m), id' coincides with the ID id that $\underline{M}$ has stored before (in mode Compute), and m is accepted by M when simulated in mode CheckAddress. In mode Compute, if $\underline{M}$ is activated for the first time in this mode, i.e., the incoming message, say $m' = (id, m)$, was accepted in mode CheckAddress for the first time, then first id is stored and then M is simulated with input m. Otherwise (if $\underline{M}$ was activated in mode Compute before), M is directly simulated with input m. If the simulated M produces output on some tape, then $\underline{M}$ prefixes this output with id and then outputs the resulting message on the corresponding tape.

The ID id typically is some session ID (SID) or some party ID (PID) or a combination of both. Clearly, it is not essential that messages are of the form (id, m). Other forms are possible as well. In fact, everything checkable in polynomial time works. We sometimes require the ID to belong to a specific (polynomially decidable) domain.

To illustrate the notion of a session version of an IITM, assume that M specifies some ideal functionality. Then $!\,\underline{M}$ denotes the multi-session version of M, i.e., a system in which an unbounded number of instances of M can be created where every copy of M can be addressed by a unique ID, where the ID could be a PID (then an instance of $\underline{M}$ might model one party running M), an SID (then an instance of $\underline{M}$ models one session of M), or it could have a more complex structure, e.g., (sid, pid) (then $\underline{M}$ models an instance of party pid running M in session sid).

Given a system $\mathcal{S}$, its *session version* $\underline{\mathcal{S}}$ is obtained by replacing all IITMs in $\mathcal{S}$ by their session version. For example, we obtain $\underline{\mathcal{S}} = \underline{M} \,|\, !\,\underline{M'}$ for $\mathcal{S} = M \,|\, !\,M'$.

Now, the following composition theorem says that if a protocol $\mathcal{P}$ realizes $\mathcal{F}$, then the multi-session version of $\mathcal{P}$ realizes the multi-session version of $\mathcal{F}$.

Theorem 2 (informal)**.** *Let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems such that $!\,\underline{\mathcal{P}}$ is environmentally bounded and $\mathcal{P} \leq^{SS} \mathcal{F}$. Then, $!\,\underline{\mathcal{P}} \leq^{SS} !\,\underline{\mathcal{F}}$.*

Theorems 1 and 2 can be applied iteratively to construct more and more complex systems. For example, as a corollary of the above theorems, we immediately obtain that for any protocol system $\mathcal{Q}$: $\mathcal{P} \leq^{SS} \mathcal{F}$ implies $\mathcal{Q} \,|\, !\underline{\mathcal{P}} \leq^{SS} \mathcal{Q} \,|\, !\underline{\mathcal{F}}$, provided that $\mathcal{Q} \,|\, !\underline{\mathcal{P}}$ is environmentally bounded. In words: $\mathcal{Q}$ using an unbounded number of instances of $\mathcal{P}$ realizes $\mathcal{Q}$ using an unbounded number of instances of $\mathcal{F}$.

When addressing a session version $\underline{M}$ of a machine M, then the machine M simulated within $\underline{M}$ is not aware of its ID and cannot use it. For example, it cannot put the ID into a message that M creates. However, sometimes this is desirable. We therefore also consider another, more general composition theorem where machines are aware of their IDs. (Maybe a little surprisingly, this theorem is a corollary of the above theorem.) While these IDs can, as already mentioned above, be interpreted in different ways, we will often refer to them as SIDs.

To this end, we first generalize the notion of a session version. We consider a (polynomially computable) *session identifier function* σ which, given a message and a tape name, outputs an SID (a bit string) or $\bot$. For example, the following function takes the prefix of a message as its SID: $\sigma_{\text{prefix}}(m, c) := s$ if $m = (s, m')$ for some s, m' and $\sigma_{\text{prefix}}(m, c) := \bot$ otherwise, for all m, c. Clearly, many more examples are conceivable. The reason that σ, besides a message, also takes a tape name as input is that the way SIDs are extracted from messages may depend on the tape name a message is received from.

Now, we say that an IITM M is a *σ-session machine* (or a *σ-session version*) if the following conditions are satisfied: (i) M rejects (in mode CheckAddress) a message m on tape c if $\sigma(m, c) = \bot$. (ii) If m_0 is the first message that M accepted (in mode CheckAddress), say on tape c_0, in a run, then, M will reject all messages m received on some tape c (in mode CheckAddress) with $\sigma(m, c) \neq \sigma(m_0, c_0)$. (iii) Whenever M outputs a messages m on tape c (in mode Compute), then $\sigma(m, c) = \sigma(m_0, c_0)$, with m_0 and c_0 as before. We say that a system $\mathcal{Q}$ is a *σ-session system* (or a *σ-session version*) if every IITM occurring in $\mathcal{Q}$ is a σ-session machine.

It is easy to see that session versions are specific forms of σ-session versions: given an IITM M, we have that $\underline{M}$ is a σ_{prefix}-session version. The crucial difference is that while σ-session versions look like session versions from the outside, inside they are aware of their SID.

We call an environmental system $\mathcal{E}$ *σ-single session* if it only outputs messages with the same SID according to σ. Hence, when interacting with a σ-session version, such an environmental system invokes at most one protocol session.

Let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems, which in the setting considered here would typically describe multiple sessions of a protocol. Moreover, we assume that $\mathcal{P}$ and $\mathcal{F}$ are σ-session versions. Now, we define what it means that a single session of $\mathcal{P}$ realizes a single session of $\mathcal{F}$. This is defined just as $\mathcal{P} \leq^{SS} \mathcal{F}$, with the difference that we consider only σ-single session environments, and hence, environments that invoke at most one session of $\mathcal{P}$ and $\mathcal{F}$.

Definition 2 (informal)**.** Let $\mathcal{P}$, $\mathcal{F}$, and σ be as above. Then, *$\mathcal{P}$ single-session realizes $\mathcal{F}$ w.r.t. σ* ($\mathcal{P} \leq^{SS}_{\sigma\text{-single}} \mathcal{F}$) if and only if there exists an adversarial system $\mathcal{S}$ (a simulator or an ideal adversary) such that $\mathcal{E} \,|\, \mathcal{P} \equiv \mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}$ for every σ-single session environmental system $\mathcal{E}$. (The details concerning runtime and interfaces are similar to Definition 1.).

Now, analogously to Theorem 2, the following theorem says that if $\mathcal{P}$ realizes $\mathcal{F}$ w.r.t. a single session, then $\mathcal{P}$ realizes $\mathcal{F}$ w.r.t. multiple sessions. As mentioned before, in the setting considered here $\mathcal{P}$ and $\mathcal{F}$ would typically model multi-session versions of a protocol/functionality.

Theorem 3 (informal)**.** *Let σ, $\mathcal{P}$, and $\mathcal{F}$ be as above. Then, $\mathcal{P} \leq^{SS}_{\sigma\text{-single}} \mathcal{F}$ implies $\mathcal{P} \leq^{SS} \mathcal{F}$.*

As pointed out before, the proof of this theorem is in fact a corollary of Theorem 2. Clearly, this theorem can be combined with the other composition theorems to construct more and more complex systems.

We finally emphasize that still details such as addressing of machines by party/session IDs, corruption, and structure of protocols are not, and do not need to be fixed in order to prove the above composition theorems. In other words, these theorems hold true for all specific choices, and hence, are very general. Instead of fixing these details, as done in other models, the IITM model leaves their definition to the protocol designer, hence providing a great degree of freedom for protocol specifications. We provide a sample instantiation of

the IITM model in Section 10 where all details are fixed in a meaningful way; we also illustrate how this instantiation can be used for modeling and analyzing protocols. Another instantiation of the IITM model is the iUC model [2], which focuses on ease of use for protocol designers by providing a simple yet flexible and expressive set of conventions for modeling protocols.

Joint state theorems and composition theorems with global setup are discussed in Section 10.2 and 10.3, respectively. As already mentioned in the introduction, unlike other models, these theorems do not require to extend or change the IITM model. The general joint state theorem is even a trivial consequence of the composition theorems.

At first reading, the reader may want to skip the detailed description of the IITM model and jump directly to Section 10, where it is illustrated how the model can be used.

3 The General Computational Model

In this section, we define our general computational model. This model is defined independently of the application to universal composability and is of independent interest. It will, however, form the basis for our treatment of universal composability. We introduce single interactive Turing machines and systems of such machines, define runs of systems, and introduce further notation and terminology.

3.1 Inexhaustible Interactive Turing Machines

We first introduce the syntax of (inexhaustible) interactive Turing machines and then the way these machines perform their computations.

3.1.1 Syntax

An *(inexhaustible) interactive Turing machine (IITM, for short)* M is a probabilistic Turing machine with the following tapes and a polynomial q associated with it, where q will be used as a bound in computations of M in mode CheckAddress (see below): a read-only tape on which the mode the IITM M is supposed to run is written (the *mode tape*)—the possible modes are CheckAddress and Compute (see below)—, a read-only tape on which the random coins are written (the *random tape*), a read-only tape on which the security parameter is written (the *security parameter tape*), a write-only tape (the *address decision tape*, used in mode CheckAddress), zero or more *input* and *output tapes*, and *work tapes*. The input and output tapes have names and we require that different tapes of M have different names.

The set of (names of) input and output tapes of M is denoted by $\mathcal{T}(M)$, the set of input tapes by $\mathcal{T}_{in}(M)$, and the set of output tapes by $\mathcal{T}_{out}(M)$.

As further explained below, the names of input and output tapes will determine how IITMs are connected in a system of IITMs: if an instance of an IITM sends a message on an output tape named c, then only an instance of an IITM with an input tape named c can receive this message.

Tapes named start and decision will serve a particular purpose. We require that only input tapes can be named start and only output tapes can be named decision. We will later use start to provide a system with external input and to trigger an IITM if no other IITM was triggered. An IITM is triggered by another IITM if the latter sends a message to the former. An IITM with an input tape named start will be called *master IITM*. On tapes named decision the final output of a system of IITMs will be written.

As mentioned, an IITM M runs in one of two modes, CheckAddress or Compute. The mode in which M is supposed to run is written on the mode tape of M.

3.1.2 Computation

We describe the computation of an (instance of an) IITM M in mode CheckAddress and Compute, respectively.[4] Informally speaking, in mode CheckAddress an IITM M checks whether the incoming message is in fact

[4] As already pointed out in Section 2, for simplicity of presentation, we do not always strictly distinguish between the description of a machine and an instance of a machine. The exact meaning of the term "machine" (or "IITM") is clear from the

addressed to it. This mode is used for the following purpose: In a system of IITMs, tape names uniquely define connections between machines of a system: if a machine, say M', has an output tape named c and another machine, say M, in the system has an input tape named c, then both machines are connected and an instance of M' can send messages to instances of M. The multiple instances of machines may, for example, model different parties and/or different sessions of a protocol running the same piece of code. Thus, when an instance of M' sends a message m on c to M, there also needs to be an additional mechanism to decide which instance of M gets to process the incoming message m (again, the tape only determines that *some* instance of M must be the receiver). This is exactly the purpose of the CheckAddress mode. As already explained in Section 2, an instance of M runs this algorithm on m to decide whether this message is supposed to be handled by itself; the instances of M run the CheckAddress algorithm in the order of the creation of the instances. For example, the sender can add a party identifier (PID) and/or session identifier (SID) to the message m. Now, in mode CheckAddress, instances of M would then check whether the incoming message has the expected form (e.g., whether it is prefixed with the expected identifier), and either accept or reject that message. The first instance to accept m then enters mode Compute. In this mode, an instance can, depending on the incoming message and its current configuration, actually process the incoming message and write output on one of its output tapes, i.e., send a message to another (instance of an) IITM, and so on. Several examples of how both modes can be used to model various types of protocols are available in Section 10.

More formally, the computation in the two modes is defined as follows.

Mode CheckAddress: If M is activated in mode CheckAddress, then the following will always be the case: CheckAddress is written on the mode tape of M, the security parameter, say 1^η, is written on the security parameter tape, and one message, say m, is written on one of the input tapes, say c; the other input tapes and the output tapes are empty—or otherwise will be emptied before M starts to run—and the contents on the work tapes and the random tape represent the current configuration of M. We require that in mode CheckAddress (i) M always halts[5] and at the end of the activation has written accept or reject on the address decision tape, (ii) the computation performed by M in this mode is deterministic, i.e., it is independent of the content on the random tape, and (iii) the number of transitions taken in the activation is bounded from above by $q(\eta + |m| + l)$, where q is the polynomial associated with M and l is the length of the content of all work tapes at the beginning of the activation.

As explained above, the CheckAddress mode is used for being able to send messages to a specific instance of a machine, and as such is purely a modeling tool (just as the concept of tapes). The purpose of this mode is also the main reason for requiring determinism: as every message is sent to a specific intended receiver, machine instances are supposed to decide whether they are the intended receiver. We emphasize that in mode CheckAddress, M cannot be exhausted. That is, whenever M is activated in this mode, M is able to "scan" its complete current configuration (except for the random tape, on which an infinite string is written), including the incoming message.

Mode Compute: If M is activated in mode Compute, then the following will always be the case: Compute is written on the mode tape of M, the security parameter, say 1^η, is written on the security parameter tape, and one message, say m, is written on one of the input tapes, say c; the other input tapes and the output tapes are empty—or otherwise will be emptied before M starts to run—and the contents on the work tapes and the random tape represent the current configuration of M. The computation of M in mode Compute may be probabilistic (i.e., depend on the content on the random tape) and it might be the case that M does not halt. However, if M does halt in this activation, then we require that M has written *at most one* message on one of its output tapes (i.e., only one message can be sent to another IITM at a time).

We note that at this point, we do not restrict the runtime of IITMs in mode Compute. IITMs are a priori unbounded w.r.t. their runtime (number of transitions taken) and space (length of the content of all tapes).

context. That is, in the context of systems, we mean the static description, whereas in the context of runs of systems/runtime behavior, we refer to instances of machines.

[5] We emphasize that a machine that halts might become active again. For example, a machine that halts in mode CheckAddress and accepts a message will then resume its computation in mode Compute, see below.

Later, in Section 4, we will introduce a notion of polynomial runtime which will guarantee that the overall runtime of a system of IITMs is polynomially bounded in the security parameter.

3.2 Systems of IITMs

A *system* of IITMs can be built according to the following grammar, where M ranges over (descriptions of) IITMs:

$$\mathcal{S} ::= M \;\mid\; (\mathcal{S} \,\|\, \mathcal{S}) \;\mid\; !\,\mathcal{S} \;.$$

We require that for every tape name c, at most two of the machines in $\mathcal{S}$ have a tape named c and if two IITMs have a tape named c, then c is an input tape in one of the machines and an output tape in the other. This implies that in $\mathcal{S}$ only at most one IITM may be a master IITM, i.e., may have start as input tape; there may be several instances of such a machine in a run of a system though. We say that $\mathcal{S}'$ is a *subsystem* of $\mathcal{S}$ if $\mathcal{S}$ contains a subexpression (modulo commutativity and associativity of $\|$) of the form $\mathcal{S}'$. For example, $!\,M_3$ and $M_1 \,\|\, !\,M_3$ are subsystems of $\mathcal{S} = M_1 \,\|\, M_2 \,\|\, !\,M_3$, but $!\,M_1$ is not.

Intuitively, $\mathcal{S}_1 \,\|\, \mathcal{S}_2$ stands for the concurrent composition of the systems $\mathcal{S}_1$ and $\mathcal{S}_2$, and $!\,\mathcal{S}$ stands for the concurrent composition of an unbounded number of instances of (machines in) the system $\mathcal{S}$, where the actual number of instances generated during a run of the system is determined by external or internal machines invoking (machines of) $\mathcal{S}$ (see Section 3.3). We call '$!$' the *bang operator*, borrowing terminology from process calculus [17, 34].

We say that an IITM M *occurs in the scope of a bang* in $\mathcal{S}$ if $\mathcal{S}$ contains a subexpression of the form $!\,\mathcal{S}'$ such that M occurs in $\mathcal{S}'$.

It will be clear from the semantics of systems, i.e., the way a system of IITMs runs, that every system $\mathcal{S}$ can equivalently be written as $\mathcal{S} = M_1 \,\|\, \cdots \,\|\, M_k \,\|\, !\,M'_1 \,\|\, \cdots \,\|\, !\,M'_{k'}$, where $M_1, \ldots, M_k$ and $M'_1, \ldots, M'_{k'}$ are IITMs, i.e., every system consists of a set of machines, where some are and other are not in the scope of a bang.

3.3 Running a System

Throughout the rest of this paper, we denote by Rand the set of all mappings from the set of natural numbers $\mathbf{N}$ to the set of infinite bit strings $\{0,1\}^\omega$. We refer to $\alpha \in \mathsf{Rand}$ as *random coins*.

We now define how a system $\mathcal{S}$ runs given a security parameter η, a bit string a as external input, and random coins $\alpha \in \mathsf{Rand}$.[6] We denote such a system by $\mathcal{S}^{\langle\alpha\rangle}(1^\eta, a)$.

Informal description. Informally speaking, in the run of $\mathcal{S}^{\langle\alpha\rangle}(1^\eta, a)$ several instances are created from the (static descriptions of) IITMs in $\mathcal{S}$; these instances then interact with each other according to their program, local state, and the tapes defined in $\mathcal{S}$. At any time only one (instance of an) IITM is active and all other (instances of) IITMs wait for new input. The active instance, say M', which is an instance of a machine M defined in $\mathcal{S}$, may write at most one message, say m, on one of its output tapes, say c. This message is then delivered to another (instance of an) IITM with an input tape named c, say N is the machine specified in $\mathcal{S}$ with an input tape named c.[7] In the current configuration of the system, there may be several instances of N. In the order of creation, the instances of N are run in mode $\mathsf{CheckAddress}$ with input m. Once one instance accepts m, this instance gets to process m, i.e., it runs in mode $\mathsf{Compute}$ with input m, and in particular, may produce output on one output tape, which is then sent to another instance and so on. If no instance of N accepts m and N is in the scope of a bang, a fresh instance of N is created and run in mode $\mathsf{CheckAddress}$. If this instance accepts m, random coins $\alpha(i)$ for some new i are written on the random tape of N and it gets to process m in mode $\mathsf{Compute}$. Otherwise, the new instance of N is deleted, m is dropped, and a master IITM is activated (with empty input on tape start). If N is not in the scope of a bang

[6] We note that the external input can be omitted. All our results also hold true in the uniform case, that is, the setting without external input. Most proofs directly carry over to the uniform case and otherwise we explicitly provide the proof for the uniform case.

[7] Recall that by our convention on the names of input tapes in systems of IITMs, there can be at most one such machine.

(and—the only instance of—N does not accept m), then too a master IITM is activated. The first IITM to be activated in a run is a master IITM. It gets a as external input (on tape start) and is run with random coins $\alpha(1)$ written on its random tape. A master IITM is also activated if the currently active machine does not produce output. A run stops if a master IITM, after being activated, does not produce output or output was written by some machine on an output tape named $\mathsf{decision}$. The overall output of a finite run is the message written on $\mathsf{decision}$ (or the empty word if no such tape exists). An informal example of a run of a system was provided in Section 2.

Formal definition. Formally, the run of $\mathcal{S}^{\langle\alpha\rangle}(1^\eta, a)$ is defined as follows: The current (global) configuration of a system in the run is described by a tuple (A, i, P) where (i) A is a sequence of configurations of IITMs, the sequence of *(previously) activated machines*, (ii) $i \leq |A|$ (where $|A|$ denotes the length of the sequence A) is the index of the last active (instance of a) machine in A (where $i = 0$ if A is the empty sequence), and (iii) P is a system. The IITMs occurring in P are called *passive*. We emphasize that the configurations in A are not the configurations of the machines that are currently active, i.e., currently performing some computation—only the i-th configuration (machine) was just active. The configurations in A rather belong to those machines that were active at some point in the run so far. Furthermore, A contains only the most recent configuration for each instance of a machine, i.e., A is a dynamic array that is updated after each computation of an instance of a machine; it is *not* a concatenation of *all* previous configurations of IITMs. As usual, the configuration of (an instance of) a machine is the content of all of its tapes, the position of the heads on these tapes, and its state. In what follows, we often do not distinguish between (the description/specification of) an IITM M and the current configuration of one of its instances: by abuse of notation, we write M for both the machine (description) and the current configuration of one of its instances.

In order to define the run of the system $\mathcal{S}^{\langle\alpha\rangle}(1^\eta, a)$, we first need to introduce some more notation. Given (a configuration of) an IITM M, we write

$$M(\mathsf{CheckAddress}, c, m) = \mathsf{accept}$$

to say that when running the IITM M in mode $\mathsf{CheckAddress}$ starting from its current configuration with m written on the input tape c and the empty bit string written on all other input tapes, on all output tapes, and the address decision tape, then M returns accept on its address decision tape. (Note that by definition, see Section 3.1.2, M always halts when running in mode $\mathsf{CheckAddress}$ and runs in polynomial time. Also recall that the run of M in this mode is deterministic.) Analogously, we write

$$M(\mathsf{CheckAddress}, c, m) = \mathsf{reject}.$$

Similarly, given (a configuration of) an IITM M, we write

$$M(\mathsf{Compute}, c, m) \to M'$$

to say that when running the IITM M in mode $\mathsf{Compute}$ starting from its current configuration with m written on the input tape c and the empty bit string written on all other input tapes, on all output tapes, and the address decision tape, M halts in configuration M'. (Note that we assume that random coins have been written on the random tape of M. By this, the run of M is fully determined.) If in the above setting M does not halt, we write

$$M(\mathsf{Compute}, c, m) \to \infty \ .$$

We now describe how a configuration (A, i, P) evolves when a message m which was output on an output tape c is read by one of the IITMs in the system, given random coins α. We will write $(A, i, P) \to^{\alpha}_{(c,m)} (A', i', P')$ to say that we obtain (A', i', P') as a successor configuration of (A, i, P) after m was read on c (by some IITM). Accordingly, we call (A', i', P') a $\to^{\alpha}_{(c,m)}$-*successor* of (A, i, P). Note that such a successor (if any) is uniquely determined.

Let $c \neq \mathsf{decision}$ be a name of a tape, m be a message, α be random coins, (A, i, P) and (A', i', P') be configurations such that A does not contain ∞ (which would mean that some of the machines in A did not halt in an activation). Then, we have

$$(A, i, P) \to^{\alpha}_{(c,m)} (A', i', P')$$

if one of the following conditions is satisfied where we assume that $A = M_1, \ldots, M_n$.

1. *One of the activated machines accepts m on tape c:* It holds that $c \in \mathcal{T}_{in}(M_{i'})$, $M_{i'}(\mathsf{CheckAddress}, c, m) = \mathsf{accept}$, and i' is minimal with this property, i.e., $M_j(\mathsf{CheckAddress}, c, m) = \mathsf{reject}$ for all $j < i'$ with $c \in \mathcal{T}_{in}(M_j)$. Furthermore, there exists a configuration $M'_{i'}$ (possibly ∞) such that $M_{i'}(\mathsf{Compute}, c, m) \to M'_{i'}$ and A' is obtained from A by replacing the content of every input and output tape of a configuration in A by the empty bit string and then replacing $M_{i'}$ by $M'_{i'}$ where, if $M'_{i'} \neq \infty$, the content of all input tapes of $M'_{i'}$ are replaced by the empty bit string. (Output tapes of $M'_{i'}$ are not emptied. One such tape may contain a non-empty message.) Moreover, $P' = P$.

2. *None of the activated machines accepts m on tape c, but a fresh instance of a machine does:* It holds that $i' = n + 1$ and for all $j \leq n$ with $c \in \mathcal{T}_{in}(M_j)$ it holds that $M_j(\mathsf{CheckAddress}, c, m) = \mathsf{reject}$ but there is an IITM M in P such that $c \in \mathcal{T}_{in}(M)$ and $M(\mathsf{CheckAddress}, c, m) = \mathsf{accept}$, where we identify M with its initial configuration, with 1^η written on its security parameter tape and with $\alpha(n+1)$ written on the random tape of M. Furthermore, there exists a configuration M' (possibly ∞) such that $M(\mathsf{Compute}, c, m) \to M'$ and A' is obtained from A by replacing the content of every input and output tape of a configuration of A by the empty bit string and appending M' at the end of A where the contents of all input tapes of M' are also deleted (if $M' \neq \infty$). (Output tapes of $M'_{i'}$ are not emptied. One such tape may contain a non-empty message.) If M is in the scope of a bang in P, then $P' = P$. Otherwise P' is obtained from P by removing M from P.

If neither 1. nor 2. is applicable, i.e., no machine (not even a fresh one) accepted m on tape c, (A, i, P) does not have a $\to^{\alpha}_{(c,m)}$-successor. Note that this includes the special case where there is no machine with input tape c, i.e., the output tape c is not connected to another machine in the system $\mathcal{S}$.

We emphasize that both in 1. and 2. the configuration in which the mode $\mathsf{Compute}$ is performed is the same as the configuration in which the machine was *before* running in mode $\mathsf{CheckAddress}$: in 1., both the mode $\mathsf{CheckAddress}$ and $\mathsf{Compute}$ are executed in the configuration $M_{i'}$, and in 2., both the mode $\mathsf{CheckAddress}$ and $\mathsf{Compute}$ are executed in the configuration M. We also note that one of the output tapes of $M'_{i'}/M'$ may contain a non-empty bit string and all other output tapes, including those of other IITMs, are empty. Finally, we point out that in 2., if some M occurs in P with $c \in \mathcal{T}_{in}(M)$, then M is uniquely determined. This is so because by definition of systems we assume that the set of names of input tapes of different occurrences of IITMs in a system are disjoint.

Having defined how a configuration (A, i, P) evolves when a machine reads a message m from tape c, we now define how a configuration (A, i, P) evolves in general. For this purpose, let α be random coins, (A, i, P) and (A', i', P') be configurations such that A does not contain ∞ and all output tapes of the configurations occurring in A are empty, except for at most one output tape $c \neq \mathsf{decision}$. We write

$$(A, i, P) \to^{\alpha} (A', i', P')$$

if one of the following conditions is satisfied where we assume that $A = M_1, \ldots, M_n$.

1. *The last active IITM did not produce output and was not a master IITM. Then a master IITM is triggered:* All output tapes of the configurations in A are empty, $\mathsf{start} \notin \mathcal{T}_{in}(M_i)$, and $(A, i, P) \to^{\alpha}_{(\mathsf{start},\varepsilon)} (A', i', P')$ where ε denotes the empty bit string. (If a master IITM did not produce output ($\mathsf{start} \in \mathcal{T}_{in}(M_i)$), then (A, i, P) does not have a successor configuration.)

2. *The last active IITM produced output and this output is accepted by another machine:* The content of some output tape $c \neq \mathsf{decision}$ is some non-empty message m and $(A, i, P) \to^{\alpha}_{(c,m)} (A', i', P')$.

3. *The last active IITM produced output which, however, is not accepted by the intended machine (including the case that there is no machine with a corresponding input tape). Then a master IITM is triggered:* The content of some output tape $c \neq \mathsf{decision}$ is some non-empty message m, but (A, i, P) does not have a $\to^{\alpha}_{(c,m)}$-successor, and $(A, i, P) \to^{\alpha}_{(\mathsf{start},\varepsilon)} (A', i', P')$ where ε denotes the empty bit string. (Note that this means that the master IITM accepted the empty message. Otherwise (A, i, P) does not have a successor.)

We refer to (A', i', P') as a $\rightarrow^{\alpha}$-successor of (A, i, P). If none of the above cases applies, then (A, i, P) does not have a $\rightarrow^{\alpha}$-successor. Also, if A contains ∞ or a non-empty message is written on the output tape $\mathsf{decision}$, and hence, there is no non-empty message on another output tape, (A, i, P) does not have a $\rightarrow^{\alpha}$-successor either.

The *(complete) run* ρ of a system $\mathcal{S}$ given the security parameter η, external input a, and random coins α (*the run of* $\mathcal{S}^{\langle\alpha\rangle}(1^\eta, a)$, for short) is the finite sequence of configurations $(A_0, i_0, P_0), (A_1, i_1, P_1), \ldots, (A_k, i_k, P_k)$ or the infinite sequence of configurations $(A_0, i_0, P_0), (A_1, i_1, P_1), \ldots$ such that the following conditions are satisfied.

1. (A_0, i_0, P_0) is the initial configuration, i.e., A_0 is the empty sequence, $i_0 = 0$, and $P_0 = \mathcal{S}$.
2. $(A_0, i_0, P_0) \rightarrow^{\alpha}_{(\mathsf{start},a)} (A_1, i_1, P_1)$.
3. $(A_j, i_j, P_j) \rightarrow^{\alpha} (A_{j+1}, i_{j+1}, P_{j+1})$ for every $j \in \{1, \ldots, k-1\}$ if ρ is finite and for every $j \geq 1$ if ρ is infinite.
4. If the sequence is finite, then (A_k, i_k, P_k) does not have a $\rightarrow^{\alpha}$-successor.

For a finite run ρ, we call k the *length of* ρ. The *overall output* of a run ρ is undefined if ρ is infinite or if in ρ some IITM does not halt (i.e., A_k contains a configuration ∞). Otherwise, the overall output of a run ρ is the message written on the tape named $\mathsf{decision}$ at the end of the run. (This message could be empty.) If no such tape exists, then the overall output is the empty message (in the following, we will treat this special case as if the empty message was written onto a tape named $\mathsf{decision}$). Note that since a run stops when a non-empty message has been written on $\mathsf{decision}$, the overall output is uniquely determined.

3.4 Probability Space and Relevant Random Variables

We consider the standard probability space over $\{0,1\}^\omega$ (the set of infinite bit strings), where the probability of a cone $\overline{\beta}$ of $\beta \in \{0,1\}^*$, which is the set of all infinite bit strings that have β as a prefix, is defined to be $2^{-|\beta|}$. Now, the probability space for Rand is defined in a standard way as the probability space on the infinite product of the probability space for $\{0,1\}^\omega$. For example, given $\beta_1, \beta_2, \beta_3 \in \{0,1\}^*$, the probability for the event $E = \{\alpha \in \mathsf{Rand} \mid \alpha(i) \text{ is prefixed with } \beta_i \text{ for all } i \in \{1,2,3\}\}$ is $2^{-|\beta_1|} \cdot 2^{-|\beta_2|} \cdot 2^{-|\beta_3|}$. This probability space over Rand guarantees that all random variables defined next are measurable.

By $\mathcal{S}(1^\eta, a)\colon \mathsf{Rand} \rightarrow \{0,1\}^* \cup \{\bot\}$ we denote the random variable that describes the overall output (i.e., output on $\mathsf{decision}$) of runs of the system $\mathcal{S}(1^\eta, a)$. More precisely, for $\alpha \in \mathsf{Rand}$ we define $\mathcal{S}(1^\eta, a)(\alpha)$ to be the overall output of the run of $\mathcal{S}^{\langle\alpha\rangle}(1^\eta, a)$, where $\bot$ denotes undefined output.

Given this random variable, the probability that a run has overall output 1 is

$$\mathrm{Prob}[\mathcal{S}(1^\eta, a) = 1] \ .$$

By $\mathrm{Time}(\mathcal{S}(1^\eta, a))\colon \mathsf{Rand} \rightarrow \mathbf{N} \cup \{\infty\}$ we denote the random variable that describes the overall number of machine transitions that have been taken by IITMs in mode $\mathsf{Compute}$ in runs of $\mathcal{S}(1^\eta, a)$. More precisely, for $\alpha \in \mathsf{Rand}$ we define $\mathrm{Time}(\mathcal{S}(1^\eta, a))(\alpha)$ to be the overall number of transitions that have been taken by IITMs in mode $\mathsf{Compute}$ in a run of $\mathcal{S}^{\langle\alpha\rangle}(1^\eta, a)$. This number is ∞ if the run is infinite or some IITM did not halt. Note that transitions taken in mode $\mathsf{CheckAddress}$ are not counted and also the emptying of input or output tapes before IITMs are activated is not counted.

Similarly, given a subsystem $\mathcal{Q}$ of $\mathcal{S}$, by $\mathrm{Time}_{\mathcal{Q}}(\mathcal{S}(1^\eta, a))$ we denote the random variable that describes the overall number of transitions in mode $\mathsf{Compute}$ that have been taken by IITMs in $\mathcal{Q}$ in runs of $\mathcal{S}(1^\eta, a)$. Clearly, $\mathrm{Time}_{\mathcal{S}}(\mathcal{S}(1^\eta, a)) = \mathrm{Time}(\mathcal{S}(1^\eta, a))$ for all systems $\mathcal{S}$.

3.5 Equivalence/Indistinguishability of Systems

We first introduce negligible functions following [4].

Definition 3. A function $f\colon \mathbf{N} \times \{0,1\}^* \to \mathbf{R}_{\geq 0}$ is called *negligible* if for all $c, d \in \mathbf{N}$ there exists $\eta_0 \in \mathbf{N}$ such that for all $\eta > \eta_0$ and all $a \in \bigcup_{\eta' \leq \eta^d} \{0,1\}^{\eta'}$: $f(\eta, a) < \eta^{-c}$.[8]

A function $f\colon \mathbf{N} \times \{0,1\}^* \to [0,1]$ is called *overwhelming* if $1 - f$ is negligible.[9]

Two systems that produce overall output[10] 1 with almost the same probability are called equivalent or indistinguishable:

Definition 4. Let $f\colon \mathbf{N} \times \{0,1\}^* \to \mathbf{R}_{\geq 0}$ be a function. Two systems $\mathcal{P}$ and $\mathcal{Q}$ are called *f-equivalent* or *f-indistinguishable* ($\mathcal{P} \equiv_f \mathcal{Q}$) if and only if for every security parameter $\eta \in \mathbf{N}$ and external input $a \in \{0,1\}^*$:

$$|\mathrm{Prob}[\mathcal{P}(1^\eta, a) = 1] - \mathrm{Prob}[\mathcal{Q}(1^\eta, a) = 1]| \leq f(\eta, a) \ .$$

Two systems $\mathcal{P}$ and $\mathcal{Q}$ are called *equivalent* or *indistinguishable* ($\mathcal{P} \equiv \mathcal{Q}$) if and only if there exists a negligible function f (Definition 3) such that $\mathcal{P} \equiv_f \mathcal{Q}$.

It is easy to see that for every two functions f, f' as in Definition 4 the relation $\equiv_f$ is reflexive and that $\mathcal{P} \equiv_f \mathcal{Q}$ and $\mathcal{Q} \equiv_{f'} \mathcal{S}$ implies $\mathcal{P} \equiv_{f+f'} \mathcal{S}$. In particular, $\equiv$ is reflexive and transitive.

4 Polynomial Time and Properties of Systems

In this section, we introduce notions of polynomial runtime for arbitrary systems, environmental systems, and protocol systems. We also state basic properties about such systems. We begin with further notation and terminology.

4.1 Further Notation and Terminology

Let $\mathcal{S}$ be a system and M be an IITM. Recall that $\mathcal{T}(M)$, $\mathcal{T}_{in}(M)$, and $\mathcal{T}_{out}(M)$ denote the set of (names of) input and output tapes, the set (of names) of input tapes, and the set (of names) of output tapes of M, respectively.

A tape c in $\mathcal{T}(\mathcal{S})$ is called *internal* if there exist two IITMs M and M' in $\mathcal{S}$ such that $c \in \mathcal{T}_{out}(M) \cap \mathcal{T}_{in}(M')$. Otherwise, c is called *external*. The set of *internal tapes* of $\mathcal{S}$ is denoted by $\mathcal{T}_{int}(\mathcal{S})$ and the set of *external tapes* of $\mathcal{S}$ by $\mathcal{T}_{ext}(\mathcal{S})$. We call c an *(external) input tape* of $\mathcal{S}$ if $c \in \mathcal{T}_{ext}(\mathcal{S})$ and $c \in \mathcal{T}_{in}(M)$ for some IITM M in $\mathcal{S}$. Analogously, c is called an *(external) output tape* of $\mathcal{S}$ if $c \in \mathcal{T}_{ext}(\mathcal{S})$ and $c \in \mathcal{T}_{out}(M)$ for some IITM M in $\mathcal{S}$. The set of (external) input and output tapes of $\mathcal{S}$ is denoted by $\mathcal{T}_{in}(\mathcal{S})$ and $\mathcal{T}_{out}(\mathcal{S})$, respectively.

Note that for every $\mathcal{S}$ we have that $\mathsf{start} \in \mathcal{T}(\mathcal{S})$ implies $\mathsf{start} \in \mathcal{T}_{in}(\mathcal{S})$ and $\mathsf{decision} \in \mathcal{T}(\mathcal{S})$ implies $\mathsf{decision} \in \mathcal{T}_{out}(\mathcal{S})$.

We call two systems compatible if they provide the same external interface:

Definition 5. Two systems $\mathcal{P}$ and $\mathcal{Q}$ are *compatible* iff $\mathcal{T}_{in}(\mathcal{P}) = \mathcal{T}_{in}(\mathcal{Q})$ and $\mathcal{T}_{out}(\mathcal{P}) = \mathcal{T}_{out}(\mathcal{Q})$, i.e., $\mathcal{P}$ and $\mathcal{Q}$ coincide on their input and output tapes.

Given two systems $\mathcal{P}$ and $\mathcal{Q}$, by

$$\mathcal{P} \,|\, \mathcal{Q}$$

we denote the parallel composition $\mathcal{P}' \,\|\, \mathcal{Q}'$ where $\mathcal{P}'$ and $\mathcal{Q}'$ are obtained from $\mathcal{P}$ and $\mathcal{Q}$ by renaming the internal tapes of $\mathcal{P}$ and $\mathcal{Q}$, respectively, such that $\mathcal{T}(\mathcal{P}') \cap \mathcal{T}_{int}(\mathcal{Q}') = \emptyset$ and $\mathcal{T}_{int}(\mathcal{P}') \cap \mathcal{T}(\mathcal{Q}') = \emptyset$. The intuition is that $\mathcal{P}$ and $\mathcal{Q}$ are different systems (e.g., a protocol and its environment) which communicate via their external tapes; they should not interfere on their internal tapes.

[8] We note that this definition of negligible is equivalent to the following: f is negligible if and only if for all positive polynomials p and q (i.e., $p(n) > 0$ and $q(n) > 0$ for all $n \in \mathbf{N}$) there exists $\eta_0 \in \mathbf{N}$ such that for all $\eta > \eta_0$ and all $a \in \bigcup_{\eta' \leq q(\eta)} \{0,1\}^{\eta'}$: $f(\eta, a) < \frac{1}{p(\eta)}$. We further note that such negligible functions have the following properties: (i) If f and g are negligible, then $f + g$ is negligible. (ii) If f is negligible and p is a positive polynomial, then $g(\eta, a) := p(\eta + |a|) \cdot f(\eta, a)$ for all η, a is negligible.

[9] By $[0,1]$ we denote the interval of all real numbers x such that $0 \leq x \leq 1$.

[10] Recall from Section 3.3 that, if a system does not have a $\mathsf{decision}$ tape, the output of terminating runs is the empty word.

Definition 6. The systems $\mathcal{Q}$ and $\mathcal{P}$ are *connectable* if each common external tape of $\mathcal{P}$ and $\mathcal{Q}$ has complementary directions (input or output), i.e., for all $c \in \mathcal{T}_{ext}(\mathcal{P}) \cap \mathcal{T}_{ext}(\mathcal{Q})$, we have that $c \in \mathcal{T}_{in}(\mathcal{P}) \cap \mathcal{T}_{out}(\mathcal{Q})$ or $c \in \mathcal{T}_{out}(\mathcal{P}) \cap \mathcal{T}_{in}(\mathcal{Q})$. If $\mathcal{Q}$ and $\mathcal{P}$ are *connectable*, then we also say that $\mathcal{P}$ *can be connected to* $\mathcal{Q}$. We denote by $\mathrm{Con}(\mathcal{Q})$ the set of all systems $\mathcal{P}$ that can be connected to $\mathcal{Q}$.

The systems $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are *connectable* if they are pairwise connectable (i.e., $\mathcal{S}_i$ and $\mathcal{S}_j$ are connectable for every $i, j \leq n$ such that $i \neq j$).

We note that if $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are connectable, then every common external tape of the systems is the external input tape of exactly one system $\mathcal{S}_i$ and the external output tape of exactly one other system $\mathcal{S}_j$. So, there is no ambiguity about how these systems connect to each other in a parallel composition. In particular, the order in which these systems are composed does not matter. In fact, the composition operator ' $|$ ' is associative (i.e., $\mathcal{P} \,|\, (\mathcal{Q} \,|\, \mathcal{S}) \equiv_0 (\mathcal{P} \,|\, \mathcal{Q}) \,|\, \mathcal{S}$) and commutative (i.e., $\mathcal{P} \,|\, \mathcal{Q} \equiv_0 \mathcal{Q} \,|\, \mathcal{P}$).

4.2 Polynomially Bounded Systems

The following notion captures that a system runs in polynomial time (except maybe with negligible probability), i.e., the overall number of transitions taken by the IITMs is bounded from above by a polynomial (in the security parameter plus the length of the external input).

Definition 7. A system $\mathcal{S}$ is *almost bounded* if there exists a polynomial p such that:

$$f(\eta, a) := \mathrm{Prob}[\mathrm{Time}(\mathcal{S}(1^\eta, a)) > p(\eta + |a|)] \text{ for all } \eta \in \mathbf{N} \text{ and } a \in \{0,1\}^*$$

is negligible (as a function in η and a).

We say that a system $\mathcal{S}$ is *strictly bounded* if there exists a polynomial p such that for every security parameter $\eta \in \mathbf{N}$ and external input $a \in \{0,1\}^*$: $\mathrm{Time}(\mathcal{S}(1^\eta, a))(\alpha) \leq p(\eta + |a|)$ for all $\alpha \in \mathsf{Rand}$.

Clearly, every strictly bounded system is almost bounded. The next lemma states that every almost bounded system can be simulated by a single strictly bounded IITM M except for a negligible error. In particular, it can be simulated (with negligible error) by a probabilistic polynomial time Turing machine.

Lemma 1. *For every almost bounded system $\mathcal{S}$ there exists an IITM M such that M (as a system) is strictly bounded and $\mathcal{S} \equiv M$.*

Proof. This lemma is a special case of Lemma 7. □

4.3 Environments and Environmental Indistinguishability

We first introduce environmental systems and then define environmental indistinguishability, i.e., indistinguishability where the distinguisher is an environmental system. In the context of universal composability, environmental systems will play the role of an environment.

Intuitively, an environmental system (also called a universally bounded system) is a system that runs in (strict) polynomial time no matter to which system it is connected.

Definition 8. A system $\mathcal{E}$ is called an *environmental system* or *universally strictly bounded*, or simply *universally bounded* if there exists a polynomial p such that for every system $\mathcal{S}$ that can be connected to $\mathcal{E}$ (i.e., $\mathcal{S} \in \mathrm{Con}(\mathcal{E})$) it holds that $\mathrm{Time}_{\mathcal{E}}((\mathcal{E} \,|\, \mathcal{S})(1^\eta, a))(\alpha) \leq p(\eta + |a|)$ for all security parameter $\eta \in \mathbf{N}$, external input $a \in \{0,1\}^*$, and random coins $\alpha \in \mathsf{Rand}$.[11]

Given a system $\mathcal{S}$, by $\mathrm{Env}(\mathcal{S})$ we denote the set of all environmental systems $\mathcal{E}$ such that $\mathcal{E}$ and $\mathcal{S}$ are connectable, i.e., $\mathcal{E} \in \mathrm{Con}(\mathcal{S})$, and $\mathcal{E}$ is universally bounded.

We call two systems environmentally indistinguishable if they cannot be distinguished by any environmental system:

[11] Recall that $\mathrm{Time}_{\mathcal{E}}((\mathcal{E} \,|\, \mathcal{S})(1^\eta, a))$ is the random variable that denotes the number of machine transitions taken by IITMs in $\mathcal{E}$ (i.e., not considering $\mathcal{S}$) in mode $\mathsf{Compute}$ in a run of $(\mathcal{E} \,|\, \mathcal{S})(1^\eta, a)$.

Definition 9. Two systems $\mathcal{P}$ and $\mathcal{Q}$ are called *environmentally equivalent* or *environmentally indistinguishable* ($\mathcal{P} \cong \mathcal{Q}$) if and only if

1. $\mathcal{P}$ and $\mathcal{Q}$ are compatible and
2. $\mathcal{E} \,|\, \mathcal{P} \equiv \mathcal{E} \,|\, \mathcal{Q}$ for all $\mathcal{E} \in \mathrm{Env}(\mathcal{P})$.

Environmental indistinguishability talks about arbitrary systems $\mathcal{P}$ and $\mathcal{Q}$. In particular, $\mathcal{E} \,|\, \mathcal{P}$ or $\mathcal{E} \,|\, \mathcal{Q}$ for some $\mathcal{E} \in \mathrm{Env}(\mathcal{P})$ are not necessarily almost bounded. In typical applications and in the context of universal composability, we are interested in systems $\mathcal{P}$ and $\mathcal{Q}$ such that $\mathcal{E} \,|\, \mathcal{P}$ and $\mathcal{E} \,|\, \mathcal{Q}$ are almost bounded for all environmental systems $\mathcal{E}$. This class of systems—environmentally bounded systems—will be defined in the next subsection. But first we make some more remarks about the notion of environmental indistinguishability.

Using the fact that the relationship $\equiv$ is reflexive and transitive, the following lemma is easy to see.

Lemma 2. *The relationship $\cong$ is reflexive and transitive.*

Remark 1. Analogously to Definition 8, one could define the notion of a *universally almost bounded system* which may take more than a polynomial number of steps in a negligible set of runs, i.e., there exists a polynomial p such that for every system $\mathcal{S} \in \mathrm{Con}(\mathcal{E})$ it holds that

$$\mathrm{Prob}[\mathrm{Time}_{\mathcal{E}}((\mathcal{E} \,|\, \mathcal{S})(1^\eta, a)) > p(\eta + |a|)]$$

is negligible (as a function in η and a). But this does not make environmental systems more powerful. In particular, it would not change the notion of environmental indistinguishability because for every universally almost bounded system $\mathcal{E}$ one could easily construct a universally strictly bounded system $\mathcal{E}'$ such that $\mathcal{E} \,|\, \mathcal{S} \equiv \mathcal{E}' \,|\, \mathcal{S}$ for every system $\mathcal{S}$ that can be connected to $\mathcal{E}$.

Remark 2. We note that the notion of a universally bounded system $\mathcal{E}$ is equivalent to the following notion (stated informally): There exists a polynomial p such that for every sequence of incoming messages, $\mathcal{E}$ "interacting" with this sequence of messages takes at most $p(\eta + |a|)$ transitions in mode Compute.

Remark 3. Environmental indistinguishability is defined with respect to one-bit output. That is, the probabilities of the environment outputs 1 (on tape decision) are compared. An alternative definition would be to require that the output (which may be more than a bit) produced by the environment when interacting with the systems is computationally indistinguishability, i.e., it cannot (with more than negligible probability) be distinguished by any polynomial time distinguisher. It is easy to show that this alternative definition is equivalent to the one-bit definition we use here, provided that $\mathsf{start}, \mathsf{decision} \notin \mathcal{T}(\mathcal{P}) = \mathcal{T}(\mathcal{Q})$: The idea is that in our version with one-bit output, the environment also plays the role of the distinguisher (see also remarks in [4]).

4.4 Protocols and Environmentally Bounded Systems

We now define the class of systems that run in polynomial time (except maybe with negligible probability) when combined with any environmental system. This is a very expressive class of systems and the one we will always consider in the context of universal composability. The runtime notion conceptually follows the one introduced in [21]. We also introduce protocol systems, which, as the name suggests, are systems which describe protocols; in the context of universal composability, these systems model ideal and real protocols.

Definition 10. A system $\mathcal{S}$ is *environmentally (almost) bounded* if for every $\mathcal{E} \in \mathrm{Env}(\mathcal{S})$: $\mathcal{E} \,|\, \mathcal{S}$ is almost bounded.

A system $\mathcal{S}$ is *environmentally strictly bounded* if for every $\mathcal{E} \in \mathrm{Env}(\mathcal{S})$: $\mathcal{E} \,|\, \mathcal{S}$ is strictly bounded.

Clearly, every environmentally strictly bounded system is environmentally bounded.

Remark 4. The following statements, which follow directly from the definitions, highlight the difference between environmentally and universally bounded systems:

1. A system $\mathcal{S}$ is universally bounded if and only if

$$\exists \text{ polynomial } p \; \forall \, \mathcal{Q} \in \mathrm{Con}(\mathcal{S}) \; \forall \, \eta \in \mathbf{N}, a \in \{0,1\}^* : p_{\mathrm{Time}_{\mathcal{S}}(\mathcal{Q} \mid \mathcal{S}) > p}(\eta, a) = 0$$

   where

$$p_{\mathrm{Time}_{\mathcal{S}}(\mathcal{Q} \mid \mathcal{S}) > p}(\eta, a) := \mathrm{Prob}[\mathrm{Time}_{\mathcal{S}}((\mathcal{Q} \mid \mathcal{S})(1^\eta, a)) > p(\eta + |a|)] \text{ for all } \eta \in \mathbf{N} \text{ and } a \in \{0,1\}^*.$$

   We note that here $\mathcal{Q}$ is any system that connects with $\mathcal{S}$, not necessarily an environmental system. Also note that $p_{\mathrm{Time}_{\mathcal{S}}(\mathcal{Q} \mid \mathcal{S}) > p}(\eta, a)$ is the probability that in a run of $(\mathcal{Q} \mid \mathcal{S})(1^\eta, a)$ the number of machine transitions taken by IITMs in $\mathcal{S}$ (i.e., not considering $\mathcal{Q}$) in mode Compute is greater than $p(\eta + |a|)$.

2. A system $\mathcal{S}$ is environmentally strictly bounded if and only if

$$\forall \, \mathcal{E} \in \mathrm{Env}(\mathcal{S}) \; \exists \text{ polynomial } p_{\mathcal{E}} \; \forall \, \eta \in \mathbf{N}, a \in \{0,1\}^* : p_{\mathrm{Time}(\mathcal{E} \mid \mathcal{S}) > p_{\mathcal{E}}}(\eta, a) = 0$$

   where

$$p_{\mathrm{Time}(\mathcal{E} \mid \mathcal{S}) > p_{\mathcal{E}}}(\eta, a) := \mathrm{Prob}[\mathrm{Time}((\mathcal{E} \mid \mathcal{S})(1^\eta, a)) > p_{\mathcal{E}}(\eta + |a|)] \text{ for all } \eta \in \mathbf{N} \text{ and } a \in \{0,1\}^*.$$

   We note that $p_{\mathrm{Time}(\mathcal{E} \mid \mathcal{S}) > p_{\mathcal{E}}}(\eta, a)$ is the probability that in a run of $(\mathcal{E} \mid \mathcal{S})(1^\eta, a)$ the number of machine transitions taken by IITMs in $\mathcal{E} \mid \mathcal{S}$ in mode Compute is greater than $p_{\mathcal{E}}(\eta + |a|)$.

3. A system $\mathcal{S}$ is environmentally (almost) bounded if and only if

$$\forall \, \mathcal{E} \in \mathrm{Env}(\mathcal{S}) \; \exists \text{ polynomial } p_{\mathcal{E}} : p_{\mathrm{Time}(\mathcal{E} \mid \mathcal{S}) > p_{\mathcal{E}}} \text{ is negligible (as a function in } \eta \text{ and } a)$$

   where $p_{\mathrm{Time}(\mathcal{E} \mid \mathcal{S}) > p_{\mathcal{E}}}$ is defined as above.

While we define two different types of environmentally bounded systems, the composition theorems of our model will be based only on environmentally almost bounded systems. As shown in Section 9, we need to allow for a negligible number of runs that can exceed the runtime bound in order to obtain useful composition theorems. Comparing environmentally almost and strictly bounded runtime notions and showing that one of them is not suitable for obtaining composition theorems is part of our contribution, which sheds light on potential runtime notions and provides additional motivation for using one notion rather than the other.

Note that the property of environmentally bounded systems, which will be used primarily for bounding the runtime of protocols, is typically easy to check for natural definitions of protocols. For example, every system that runs in polynomial time in the length of the security parameter and its inputs, including inputs of previous activations, (except maybe in a negligible set of runs) is environmentally bounded. This should already cover all interesting protocols. In fact, we are not aware of any real world protocol that is not environmentally bounded (see also the discussion in Section 8 and concrete examples in Section 10.4).

The property of being environmentally bounded is defined on the level of systems (instead of individual machines) to provide extra generality and flexibility, i.e., it does not matter how exactly the runtime is distributed internally within the system. This also has the advantage that a protocol designer does not have to deal with manually transferring runtime between individual machines, as is the case for notions defined on the level of single machines. We note that our runtime notion actually captures as special cases certain runtime notions that are defined on the level of individual machines, including the one of the UC model. Informally speaking, the UC runtime notion interprets input bits as runtime tokens that can be forwarded to other machines by sending a message. The runtime of each machine is required to be bounded by a fixed polynomial in the number of runtime tokens that it currently holds, i.e., the number of received tokens minus the number of forwarded tokens. A system consisting only of machines that follow the UC runtime notion is also environmentally bounded. Thus, if a protocol designer desires, he is free to use the machine based UC runtime notion also within the IITM model with its system based runtime notion (see Section 11 for a more detailed explanation and discussion of the UC runtime notion, including some of its downsides).

Finally, we note that quantifying over all universally *almost* bounded systems $\mathcal{E}$ would not have strengthened the notion of environmentally bounded systems (see Remark 1). Moreover, while the notion "environmentally bounded" talks about the overall runtime of the composed system $\mathcal{E} \mid \mathcal{S}$, we could equivalently have defined it by only restricting the runtime of $\mathcal{S}$, since $\mathcal{E}$ is universally bounded anyway:

Lemma 3. *A system $\mathcal{S}$ is environmentally bounded if and only if for every $\mathcal{E} \in \mathrm{Env}(\mathcal{S})$ there exists a polynomial p such that $\mathrm{Prob}[\mathrm{Time}_{\mathcal{S}}((\mathcal{E} \mid \mathcal{S})(1^\eta, a)) > p(\eta + |a|)]$ is negligible (as a function in η and a); similarly for environmentally strictly bounded systems.*

We now define protocol systems.

Definition 11. A *protocol system* $\mathcal{P}$ is a system such that (i) no tape in $\mathcal{P}$ is named start or decision and (ii) for every IITM M occurring in $\mathcal{P}$ such that M is not in the scope of a bang, we require that M accepts every incoming message in mode CheckAddress.

The motivation behind condition (ii) is that if M does not occur in the scope of a bang, then in every run of $\mathcal{P}$ (in some context) there will be at most one instance of M. Hence, there is no reason to address different instances of M, and therefore, in mode CheckAddress, M should accept every incoming message. This condition will be used in the proofs of the composition theorems for unbounded self-composition.

In the above definition of protocol systems, we do not explicitly require that such systems are environmentally bounded—although this will typically be the case—in order to obtain more general results. For example, in the composition theorems we typically do not require the ideal protocol system to be environmentally bounded.

4.5 Properties of Systems

In this section, we summarize some fundamental and useful properties of systems. Some of these properties are not satisfied in some models for universal composability, causing partly severe problems (see also the discussion in Section 11).

The following lemma, which easily follows from the definition of systems, says that consistently changing the names of tapes in a system does not change the behavior of the system.

Lemma 4. *Let $\mathcal{S}_1, \ldots, \mathcal{S}_k$ be connectable systems. Furthermore, for all $i \leq k$, let $\mathcal{S}'_i$ be derived from $\mathcal{S}_i$ by consistently (w.r.t. the other $\mathcal{S}'_j$) renaming external tapes, where, however,* start *and* decision *may not be renamed. Then,*

$$\mathcal{S}_1 \mid \cdots \mid \mathcal{S}_k \equiv_0 \mathcal{S}'_1 \mid \cdots \mid \mathcal{S}'_k$$

and

$$\mathcal{S}_1 \mid \cdots \mid \mathcal{S}_k \text{ is almost/strictly bounded} \quad \text{iff} \quad \mathcal{S}'_1 \mid \cdots \mid \mathcal{S}'_k \text{ is almost/strictly bounded.}$$

We now consider what we call a *dummy IITM* $\mathcal{D}$ which simply forwards messages between entities: The dummy IITM has for all of its input tapes a corresponding output tape. The concrete set of input and output tapes that $\mathcal{D}$ has depends on the entities between which $\mathcal{D}$ is put. The dummy IITM accepts all messages on input tapes in mode CheckAddress and in mode Compute it simply copies a message received on an input tape to the corresponding output tape. We note that, except for the set of input and output tapes, $\mathcal{D}$ does not depend on the entities between which it is put.

More precisely, let $\mathcal{T}_{in}$ and $\mathcal{T}_{out}$ be disjoint finite sets of tapes. Moreover, let $\mathcal{T}'_{in} = \{c' \mid c \in \mathcal{T}_{in}\}$ and $\mathcal{T}'_{out} = \{c' \mid c \in \mathcal{T}_{out}\}$ where c' is a new copy of c, i.e., a new tape with a new name.

We define

$$\mathcal{D} = \mathcal{D}(\mathcal{T}_{in}, \mathcal{T}_{out})$$

to be an IITM with input tapes $\mathcal{T}_{out} \cup \mathcal{T}'_{in}$ and output tapes $\mathcal{T}_{in} \cup \mathcal{T}'_{out}$. In mode CheckAddress, $\mathcal{D}$ always accepts. In mode Compute, $\mathcal{D}$ copies every message received on $c \in \mathcal{T}_{out}$ onto $c' \in \mathcal{T}'_{out}$ and every message received on $c' \in \mathcal{T}'_{in}$ onto $c \in \mathcal{T}_{in}$.

The following lemma says that the dummy IITM can be plugged between two systems without changing the behavior of the overall system. The proof of the lemma is straightforward.

Lemma 5. *Let $\mathcal{P}$ and $\mathcal{Q}$ be two connectable systems. Let $\mathcal{T}_{ext} = \mathcal{T}_{ext}(\mathcal{P}) \cap \mathcal{T}_{ext}(\mathcal{Q})$, $\mathcal{D} = \mathcal{D}(\mathcal{T}_{ext} \cap \mathcal{T}_{in}(\mathcal{P}), \mathcal{T}_{ext} \cap \mathcal{T}_{out}(\mathcal{P}))$, and $\mathcal{Q}'$ be obtained from $\mathcal{Q}$ by renaming all tapes c in $\mathcal{T}_{ext}$ by c'. Then,*

$$\mathcal{P} \mid \mathcal{Q} \equiv_0 \mathcal{P} \mid \mathcal{D} \mid \mathcal{Q}'$$

and

$$\mathcal{P} \mid \mathcal{Q} \text{ is almost/strictly bounded} \quad \textit{iff} \quad \mathcal{P} \mid \mathcal{D} \mid \mathcal{Q}' \text{ is almost/strictly bounded.}$$

While the above lemma is obvious and expected, it does not hold in all other models for universal composability, which is often very problematic. For instance, it does not hold in general in Canetti's UC model. However, the lemma is, for example, needed in order to prove that UC and dummy UC are equivalent security notions (see also Section 6.3), which in turn is needed to prove the composition theorem in the UC model (see also Section 11.1).

The following three lemmas also state fundamental properties of our general computational model and the runtime notions that we use. These properties are again essential for many general results, such as composition and joint state theorems. They are all about replacing a subsystem by a single IITM. Unfortunately, they are not satisfied in all other models for universal composability, causing severe problems in these models (see below and Section 11 for a discussion).

Lemma 6. *For every system $\mathcal{S}$ there exists an IITM M such that the following conditions are satisfied:*

1. *M and $\mathcal{S}$ are compatible.*
2. *M accepts every message in mode* CheckAddress.
3. *For every system $\mathcal{Q}$ that can be connected to $\mathcal{S}$:*
   (a) *$\mathcal{S} \mid \mathcal{Q} \equiv_0 M \mid \mathcal{Q}$ and*
   (b) *$\mathcal{S} \mid \mathcal{Q}$ is almost/strictly bounded if and only if $M \mid \mathcal{Q}$ is almost/strictly bounded.*

Proof. We define an IITM M which simulates $\mathcal{S}$. The input and output tapes of M are the tapes in $\mathcal{T}_{in}(\mathcal{S})$ and $\mathcal{T}_{out}(\mathcal{S})$, respectively. The machine M accepts every message in mode CheckAddress. Hence, by construction, M is compatible with $\mathcal{S}$ and accepts every message in mode CheckAddress. In mode Compute, M internally simulates the system $\mathcal{S}$ according to the definition of running a system. In particular, M stores the configurations of all (previously) activated machines in $\mathcal{S}$. Upon input, M simulates the according machines in mode CheckAddress and in mode Compute. If, in mode CheckAddress, a message is rejected, M stops the current activation and does not produce output. Otherwise, the computation is continued in mode Compute (as usual, with the configuration of the simulated machine set to the one before the simulation of the CheckAddress mode started.) If necessary, M creates new activated machines following the definition of running a system. If simulated machines produce internal output (i.e., output on internal tapes of $\mathcal{S}$), M continues the internal simulation. If some simulated machine produces external output (i.e., output on an external tape of $\mathcal{S}$), M simply produces this external output.

Clearly, M is an IITM because the runtime of M in mode CheckAddress is bounded by some polynomial (in fact, the runtime of M in mode CheckAddress is constant because M accepts every message). We note that the runtime of M in mode Compute might not be bounded but this only becomes relevant later; it is not a requirement for IITMs per se.

Since M perfectly simulates $\mathcal{S}$, we have

$$\mathcal{S} \mid \mathcal{Q} \equiv_0 M \mid \mathcal{Q}$$

for every system $\mathcal{Q}$ that can be connected to $\mathcal{S}$.

Now, we prove 3. (b) and here we have to deal with the runtime of M in mode Compute. First, we note that $\mathcal{Q}$ has exactly the same view in both systems because M perfectly simulates $\mathcal{S}$. In particularly, we have that

$$\mathrm{Time}_{\mathcal{Q}}((\mathcal{S} \mid \mathcal{Q})(1^\eta, a)) = \mathrm{Time}_{\mathcal{Q}}((M \mid \mathcal{Q})(1^\eta, a)) \ ,^{12} \tag{1}$$

for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$.

The implication from right to left in 3. (b) is easy to prove: Since M perfectly simulates in particular the mode Compute of all machines in $\mathcal{S}$ (recall that $\mathrm{Time}((\mathcal{S} \mid \mathcal{Q})(1^\eta, a))$ only counts the runtime in mode

[12] That is, $\mathrm{Time}_{\mathcal{Q}}((\mathcal{S} \mid \mathcal{Q})(1^\eta, a))(\alpha) = \mathrm{Time}_{\mathcal{Q}}((M \mid \mathcal{Q})(1^\eta, a))(\alpha)$ for all random coins $\alpha \in$ Rand.

Compute), we have that $\text{Time}_{\mathcal{S}}((\mathcal{S} \mid \mathcal{Q})(1^\eta, a)) \leq \text{Time}_M((M \mid \mathcal{Q})(1^\eta, a))$, for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$, and we obtain:

$$\begin{aligned}\text{Time}((\mathcal{S} \mid \mathcal{Q})(1^\eta, a)) &= \text{Time}_{\mathcal{S}}((\mathcal{S} \mid \mathcal{Q})(1^\eta, a)) + \text{Time}_{\mathcal{Q}}((\mathcal{S} \mid \mathcal{Q})(1^\eta, a)) \\ &\leq \text{Time}_M((M \mid \mathcal{Q})(1^\eta, a)) + \text{Time}_{\mathcal{Q}}((M \mid \mathcal{Q})(1^\eta, a)) \\ &= \text{Time}((M \mid \mathcal{Q})(1^\eta, a))\end{aligned}$$

We conclude that $\mathcal{S} \mid \mathcal{Q}$ is almost/strictly bounded if $M \mid \mathcal{Q}$ is almost/strictly bounded.

To prove the direction from left to right, we assume that $\mathcal{S} \mid \mathcal{Q}$ is almost bounded (the case of strict boundedness is similar). We note that M, in mode Compute, also has to do all the maintenance work such as emptying all input and output tapes before simulated machines are activated in mode CheckAddress or Compute. So, to show that $M \mid \mathcal{Q}$ is almost bounded, we have to show that M can perform the simulation of the computations in mode CheckAddress and Compute and the maintenance work in polynomial time.

Since $\mathcal{S} \mid \mathcal{Q}$ is almost bounded, there exists a polynomial p such that

$$f(\eta, a) := \text{Prob}[\text{Time}((\mathcal{S} \mid \mathcal{Q})(1^\eta, a)) > p(\eta + |a|)] \tag{2}$$

is negligible (as a function in η and a). Hence, we have that the following holds in almost every run of $(\mathcal{S} \mid \mathcal{Q})(1^\eta, a)$ (i.e., except for a negligible set of runs): (i) the overall length of all messages that are sent (internally in $\mathcal{S}$ and $\mathcal{Q}$ and externally between $\mathcal{S}$ and $\mathcal{Q}$) is bounded by $p(\eta + |a|)$, (ii) the number of activated instances of machines in $\mathcal{S}$ is bounded by $p(\eta + |a|)$, and (iii) the size of all configurations (where only used random coins are counted) of activated instances of machines in $\mathcal{S}$ is bounded by $p(\eta + |a|)$.

The runtime in mode CheckAddress of every IITM is polynomially bounded in the length of the configuration plus the security parameter plus the input message. Hence, because of (i), (ii), and (iii) and because M perfectly simulates $\mathcal{S}$, we conclude that the runtime of M in mode Compute is polynomially bounded. More precisely, there exists a polynomial p' such that

$$f'(\eta, a) := \text{Prob}[\text{Time}_M((M \mid \mathcal{Q})(1^\eta, a)) > p'(\eta + |a|)]$$

is negligible (as a function in η and a). From this, using (1) and (2), we conclude that for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$:

$$\text{Prob}[\text{Time}((M \mid \mathcal{Q})(1^\eta, a)) > p(\eta + |a|) + p'(\eta + |a|)] \leq f(\eta, a) + f'(\eta, a) \ .$$

Hence, $M \mid \mathcal{Q}$ is almost bounded. □

The above lemma, which is also used in the proofs of the following two lemmas, is not satisfied in the UC model, which, as shown in [26] (see also the discussion in Section 11.1), causes the general joint state theorem to fail in the UC model.

The following lemma shows how one subsystem of an almost bounded system can be replaced by one universally bounded IITM (i.e., an environmental system which consists of a single IITM).

Lemma 7. *Let $\mathcal{S}$ and $\mathcal{Q}$ be connectable systems such that $\mathcal{S} \mid \mathcal{Q}$ is almost bounded and* $\mathsf{start} \in \mathcal{T}(\mathcal{S})$ *(i.e., $\mathcal{S}$ contains a master IITM, and hence, $\mathcal{Q}$ does not). Then there exists an IITM M such that the following conditions are satisfied:*

1. *M and $\mathcal{S}$ are compatible.*
2. *M accepts every message in mode* CheckAddress*.*
3. *M is universally bounded.*
4. *$M \mid \mathcal{Q}$ is almost bounded.*
5. *$M \mid \mathcal{Q} \equiv \mathcal{S} \mid \mathcal{Q}$.*

Proof. By Lemma 6, we may assume that $\mathcal{S} = M'$ is a single IITM that accepts every message in mode CheckAddress. We define an IITM M which simulates $\mathcal{S}$. The input and output tapes of M are the tapes in $\mathcal{T}_{in}(\mathcal{S})$ and $\mathcal{T}_{out}(\mathcal{S})$, respectively.

Before we specify how M works in mode Compute, observe that, because $\mathcal{S} \mid \mathcal{Q}$ is almost bounded, there exists a polynomial p such that

$$f(\eta, a) := \text{Prob}[\text{Time}((\mathcal{S} \mid \mathcal{Q})(1^\eta, a)) > p(\eta + |a|)] \tag{3}$$

is negligible (as a function in η and a).

We now define M to simulate the system $\mathcal{S}$ as follows: If invoked in mode CheckAddress, M will, just as $\mathcal{S}$, accept every message. In mode Compute, M will simulate $\mathcal{S}$ in mode Compute where, however, not more than $p(\eta + |a|)$ transitions of the IITM $\mathcal{S}$ are simulated overall (note that M knows $|a|$ because $\mathsf{start} \in \mathcal{T}(\mathcal{S})$, and hence, $\mathsf{start} \in \mathcal{T}_{in}(M)$). If this bound is reached, M goes into some final state from which M cannot take any transition. In particular, from that point on, in mode Compute, M ignores all incoming messages, does not make any further step, and does not produce output.

By construction, M is compatible with $\mathcal{S}$ and it is easy to see that M is universally bounded. It remains to show that $M \mid \mathcal{Q}$ is almost bounded and that $M \mid \mathcal{Q} \equiv \mathcal{S} \mid \mathcal{Q}$. For this purpose, let $B(1^\eta, a)$ denote the event that in a run of $(\mathcal{S} \mid \mathcal{Q})(1^\eta, a)$ the system $\mathcal{S}$ takes more than $p(\eta + |a|)$ transitions (in mode Compute), i.e., $B(1^\eta, a) = \{\alpha \in \mathsf{Rand} \mid \text{Time}_{\mathcal{S}}((\mathcal{S} \mid \mathcal{Q})(1^\eta, a))(\alpha) > p(\eta + |a|)\}$.[13] By (3), we have that $\text{Prob}[B(1^\eta, a)] \leq f(\eta, a)$. Let $B'(1^\eta, a)$ denote the event that in a run of $(M \mid \mathcal{Q})(1^\eta, a)$ the bound $p(\eta + |a|)$ is reached. By $\overline{B(1^\eta, a)}$ and $\overline{B'(1^\eta, a)}$, we denote the complement of event $B(1^\eta, a)$ and $B'(1^\eta, a)$, respectively.

Now, since M perfectly simulates the machine $\mathcal{S}$ in mode CheckAddress and in mode Compute until $B'(1^\eta, a)$ occurs, we conclude that:

$$\text{Prob}[B'(1^\eta, a)] = \text{Prob}[B(1^\eta, a)] \tag{4}$$

$$\text{Prob}[\text{Time}_{\mathcal{Q}}((M \mid \mathcal{Q})(1^\eta, a)) > p(\eta + |a|) \wedge \overline{B'(1^\eta, a)}] = \text{Prob}[\text{Time}_{\mathcal{Q}}((\mathcal{S} \mid \mathcal{Q})(1^\eta, a)) > p(\eta + |a|) \wedge \overline{B(1^\eta, a)}] \tag{5}$$

$$\text{Prob}[(M \mid \mathcal{Q})(1^\eta, a) = 1 \wedge \overline{B'(1^\eta, a)}] = \text{Prob}[(\mathcal{S} \mid \mathcal{Q})(1^\eta, a) = 1 \wedge \overline{B(1^\eta, a)}] \ . \tag{6}$$

From this, we obtain:

$$\begin{aligned}
&\text{Prob}[\text{Time}_{\mathcal{Q}}((M \mid \mathcal{Q})(1^\eta, a)) > p(\eta + |a|)] \\
&\quad \leq \ \text{Prob}[B'(1^\eta, a)] + \text{Prob}[\text{Time}_{\mathcal{Q}}((M \mid \mathcal{Q})(1^\eta, a)) > p(\eta + |a|) \wedge \overline{B'(1^\eta, a)}] \\
&\quad \overset{(4),(5)}{=} \text{Prob}[B(1^\eta, a)] + \text{Prob}[\text{Time}_{\mathcal{Q}}((\mathcal{S} \mid \mathcal{Q})(1^\eta, a)) > p(\eta + |a|) \wedge \overline{B(1^\eta, a)}] \\
&\quad \leq \ \text{Prob}[B(1^\eta, a)] + \text{Prob}[\text{Time}_{\mathcal{Q}}((\mathcal{S} \mid \mathcal{Q})(1^\eta, a)) > p(\eta + |a|)] \\
&\quad \overset{(3)}{\leq} \ 2 \cdot f(\eta, a) \ .
\end{aligned}$$

Hence, since M is universally bounded, $M \mid \mathcal{Q}$ is almost bounded. Furthermore, we obtain:

$$\begin{aligned}
&|\text{Prob}[(M \mid \mathcal{Q})(1^\eta, a) = 1] - \text{Prob}[(\mathcal{S} \mid \mathcal{Q})(1^\eta, a) = 1]| \\
&\quad \overset{(6)}{=} |\text{Prob}[(M \mid \mathcal{Q})(1^\eta, a) = 1 \wedge B'(1^\eta, a)] - \text{Prob}[(\mathcal{S} \mid \mathcal{Q})(1^\eta, a) = 1 \wedge B(1^\eta, a)]| \\
&\quad \overset{(4)}{\leq} \text{Prob}[B(1^\eta, a)] \\
&\quad \overset{(3)}{\leq} f(\eta, a) \ .
\end{aligned}$$

Hence, $M \mid \mathcal{Q} \equiv \mathcal{S} \mid \mathcal{Q}$. □

[13] Recall that we can identify a run ρ with security parameter η and external input a with the random coins $\alpha \in \mathsf{Rand}$ used in this run since a run is uniquely determined by the random coins.

Definition 12. We denote the IITM M as constructed in the proof of Lemma 7 by $[\mathcal{S}]_{\mathcal{Q}}$.

The following lemma will allow us to "open" $[\mathcal{S}]_{\mathcal{Q}}$, i.e., replace $[\mathcal{S}]_{\mathcal{Q}}$ by $\mathcal{S}$, in a context different from $\mathcal{Q}$.

Lemma 8. *Let $\mathcal{S}$ and $\mathcal{Q}$ be connectable systems such that $\mathcal{S} \mid \mathcal{Q}$ is almost bounded,* $\mathsf{start} \in \mathcal{T}(\mathcal{S})$ *(i.e., $\mathcal{S}$ contains a master IITM), and* $\mathsf{decision} \notin \mathcal{T}(\mathcal{Q})$*. Furthermore, let $\mathcal{Q}'$ be a system which is compatible with $\mathcal{Q}$ and satisfies the following condition: $\mathcal{E} \mid \mathcal{Q} \equiv \mathcal{E} \mid \mathcal{Q}'$ for every $\mathcal{E} \in \mathrm{Env}(\mathcal{Q})$ such that $\mathcal{E} \mid \mathcal{Q}$ is almost bounded. Then,*

$$[\mathcal{S}]_{\mathcal{Q}} \mid \mathcal{Q}' \equiv \mathcal{S} \mid \mathcal{Q}' \ .$$

Moreover, if $[\mathcal{S}]_{\mathcal{Q}} \mid \mathcal{Q}'$ is almost bounded, then $\mathcal{S} \mid \mathcal{Q}'$ is almost bounded too.

Proof. Just as in the proof of Lemma 7, by Lemma 6 we may assume that $\mathcal{S}$ is a single IITM that accepts every message in mode CheckAddress. Now, recall that by definition, $[\mathcal{S}]_{\mathcal{Q}}$ exactly simulates all transitions of $\mathcal{S}$ up to a certain polynomial bound and that when running $[\mathcal{S}]_{\mathcal{Q}} \mid \mathcal{Q}$ this bound is reached with only negligible probability. It follows that the probability that this bound is reached when running the system $[\mathcal{S}]_{\mathcal{Q}} \mid \mathcal{Q}'$ is negligible as well. Otherwise, one could easily construct a universally bounded system $\mathcal{E}$ such that $\mathcal{E}$ can be connected to $\mathcal{Q}$ (i.e., $\mathcal{E} \in \mathrm{Env}(\mathcal{Q})$) and $\mathcal{E} \mid \mathcal{Q} \not\equiv \mathcal{E} \mid \mathcal{Q}'$, in contradiction to the assumption in Lemma 8: The system $\mathcal{E}$ is defined to simulate $[\mathcal{S}]_{\mathcal{Q}}$ and output 1 on decision if and only if the bound is reached. (If in the simulation of $[\mathcal{S}]_{\mathcal{Q}}$ output shall be written on decision before the bound is reached, $\mathcal{E}$ writes 0 on decision and halts.) Since $\mathsf{decision} \notin \mathcal{T}(\mathcal{Q})$, 1 is output on decision in runs of $\mathcal{E} \mid \mathcal{Q}$ and $\mathcal{E} \mid \mathcal{Q}'$, respectively, if and only if the bound is reached.

It follows that with overwhelming probability $[\mathcal{S}]_{\mathcal{Q}}$ exactly simulates $\mathcal{S}$ in the system $[\mathcal{S}]_{\mathcal{Q}} \mid \mathcal{Q}'$. Thus, we obtain $[\mathcal{S}]_{\mathcal{Q}} \mid \mathcal{Q}' \equiv \mathcal{S} \mid \mathcal{Q}'$. Similarly, it is easy to see that if $[\mathcal{S}]_{\mathcal{Q}} \mid \mathcal{Q}'$ is almost bounded, then $\mathcal{S} \mid \mathcal{Q}'$ is almost bounded too. □

We note that in the above lemma the condition that Q and Q' are indistinguishable for all environments such that $\mathcal{E} \mid \mathcal{Q}$ are almost bounded is, although sufficient for our purposes, much stronger than what is needed for this lemma. As can be seen from the proof, it suffices to require that $[\mathcal{S}]^{\mathrm{alert}}_{\mathcal{Q}} \mid \mathcal{Q} \equiv [\mathcal{S}]^{\mathrm{alert}}_{\mathcal{Q}} \mid \mathcal{Q}'$ where $[\mathcal{S}]^{\mathrm{alert}}_{\mathcal{Q}}$ is the environment $\mathcal{E}$ constructed in the proof, i.e., it simulates $[\mathcal{S}]_{\mathcal{Q}}$ and outputs 1 on decision if and only if the bound is reached.

5 Composition Theorems for Environmental Indistinguishability

We now prove general composition theorems for environmental indistinguishability ($\cong$). They are the core of the composition theorems for universal composability, which are presented in Section 7. In fact, the composition theorems for universal composability security are merely corollaries of the theorems presented in this section.

In a nutshell, these theorems say that the concurrent composition of environmentally indistinguishable systems is environmentally indistinguishable. In particular, these theorems can be used to establish environmentally indistinguishability in a modular way: it suffices to show environmental indistinguishability for subsystems in order to conclude that the concurrent composition of the subsystems are environmentally indistinguishable as well.

We first, in Section 5.1, present a composition theorem for the composition of a constant number of, possibly different, systems. We then state a theorem which says that if two systems are environmentally indistinguishable, then they are environmentally indistinguishable even if an unbounded number of concurrent sessions/copies of these systems may be executed (unbounded self-composition). We actually prove two versions of this theorem, one in which the systems are not aware of the session identifiers that are used to address the different sessions (see Section 5.2) and one in which they are aware of the session identifiers (see Section 5.3), where the latter theorem is a corollary of the former one. The theorems for a constant number of protocols and for unbounded self-composition can freely be combined to establish, in a modular way, environmental indistinguishability of more and more complex systems.

5.1 Composition Theorem for a Constant Number of Systems

We now present the composition theorem for the composition of a constant number of (possibly different) systems. The proof of this theorem uses merely the equational principles (Lemma 7 and 8) established in Section 4.5.

Theorem 4. *Let $k \geq 1$ and $\mathcal{S}, \mathcal{P}_1, \ldots, \mathcal{P}_k, \mathcal{Q}_1, \ldots, \mathcal{Q}_k$ be protocol systems[14] such that the following conditions are satisfied:*

1. *For all $j \leq k$: $\mathcal{P}_j$ and $\mathcal{Q}_j$ are environmentally bounded and $\mathcal{P}_j \cong \mathcal{Q}_j$.*
2. *$\mathcal{S}, \mathcal{P}_1, \ldots, \mathcal{P}_k$ are connectable (hence, $\mathcal{S}, \mathcal{Q}_1, \ldots, \mathcal{Q}_k$ are connectable) and $\mathcal{S} \,|\, \mathcal{P}_1 \,|\, \cdots \,|\, \mathcal{P}_k$ is environmentally bounded.*

Then, $\mathcal{S} \,|\, \mathcal{P}_1 \,|\, \cdots \,|\, \mathcal{P}_k \cong \mathcal{S} \,|\, \mathcal{Q}_1 \,|\, \cdots \,|\, \mathcal{Q}_k$ and $\mathcal{S} \,|\, \mathcal{Q}_1 \,|\, \cdots \,|\, \mathcal{Q}_k$ is environmentally bounded.

Before we prove this theorem, we make some remarks.

Remark 5. We note that the system $\mathcal{S}$ in the above theorem could be a simple forwarder or even a system without external tapes, so the theorem also holds if $\mathcal{S}$ is omitted. That is, if $\mathcal{P}_1 \,|\, \cdots \,|\, \mathcal{P}_k$ is environmentally bounded and the assumptions of the theorem hold, then $\mathcal{P}_1 \,|\, \cdots \,|\, \mathcal{P}_k \cong \mathcal{Q}_1 \,|\, \cdots \,|\, \mathcal{Q}_k$ and $\mathcal{Q}_1 \,|\, \cdots \,|\, \mathcal{Q}_k$ is environmentally bounded.

Remark 6. We note that Condition 2. in the above theorem is easy to check. Connectability is a simple syntactic condition. Moreover, for typical applications, it is easy to check whether a system is environmentally bounded (see also the discussion in Section 8, in particular Lemma 15).

Proof of Theorem 4. We first show the theorem for the case $k = 1$. The proof of this case is depicted in Figure 2. For any $k \geq 1$, the theorem follows by applying the same argument iteratively, see below.

Assume that $k = 1$ and let $\mathcal{E} \in \mathrm{Env}(\mathcal{S} \,|\, \mathcal{P}_1) = \mathrm{Env}(\mathcal{S} \,|\, \mathcal{Q}_1)$. We may assume that $\mathsf{start} \in \mathcal{T}(\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{P}_1) = \mathcal{T}(\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{Q}_1)$. Otherwise, there would not exist a master IITM, and hence, a run would always directly halt with empty overall output. Since $\mathcal{S} \,|\, \mathcal{P}_1$ is environmentally bounded, $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{P}_1$ is almost bounded and we have that:

$$\begin{aligned} \mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{P}_1 &\equiv [\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}_1} \,|\, \mathcal{P}_1 && \text{(Lemma 7)} \\ &\equiv [\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}_1} \,|\, \mathcal{Q}_1 && (\mathcal{P}_1 \cong \mathcal{Q}_1) \\ &\equiv \mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{Q}_1 && (\mathcal{P}_1 \cong \mathcal{Q}_1, \text{Lemma 8}). \end{aligned}$$

We note that, in order to apply Lemmas 7 and 8, we need that $\mathsf{start} \in \mathcal{T}(\mathcal{E} \,|\, \mathcal{S})$ and $\mathsf{decision} \notin \mathcal{T}(\mathcal{P}_1) = \mathcal{T}(\mathcal{Q}_1)$, which is true because $\mathcal{P}_1$ and $\mathcal{Q}_1$ are protocol systems. Moreover, since $\mathcal{Q}_1$ is environmentally bounded and $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}_1}$ is universally bounded, $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}_1} \,|\, \mathcal{Q}_1$ is almost bounded and, by Lemma 8, we conclude that $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{Q}_1$ is almost bounded. Since this holds for all $\mathcal{E} \in \mathrm{Env}(\mathcal{S} \,|\, \mathcal{P}_1) = \mathrm{Env}(\mathcal{S} \,|\, \mathcal{Q}_1)$, we conclude that $\mathcal{S} \,|\, \mathcal{P}_1 \cong \mathcal{S} \,|\, \mathcal{Q}_1$ and $\mathcal{S} \,|\, \mathcal{Q}_1$ is environmentally bounded, which proves the theorem for $k = 1$.

We now prove the theorem for any $k \geq 1$. For every $r \leq k$, we define the r-th hybrid system:

$$\mathcal{H}_r := \mathcal{S} \,|\, \mathcal{Q}_1 \,|\, \cdots \,|\, \mathcal{Q}_{r-1} \,|\, \mathcal{P}_{r+1} \,|\, \cdots \,|\, \mathcal{P}_k \ ,$$

[14] We note that a stronger variant of this theorem holds where $\mathcal{S}, \mathcal{P}_1, \ldots, \mathcal{P}_k, \mathcal{Q}_1, \ldots, \mathcal{Q}_k$ are arbitrary systems (not necessarily protocol systems) such that $\mathsf{start}, \mathsf{decision} \notin \mathcal{T}(\mathcal{P}_j)$ and $\mathsf{start}, \mathsf{decision} \notin \mathcal{T}(\mathcal{Q}_j)$ for all $j \leq k$.

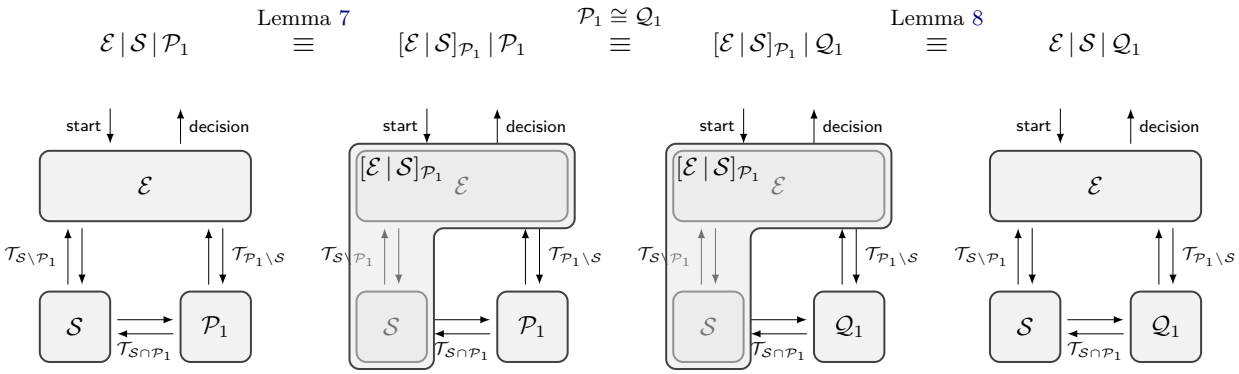

Figure 2: The proof of Theorem 4 for the case $k = 1$. We use the following abbreviations: $\mathcal{T}_{\mathcal{S}\setminus\mathcal{P}_1} := \mathcal{T}_{ext}(\mathcal{S}) \setminus \mathcal{T}_{ext}(\mathcal{P}_1)$, $\mathcal{T}_{\mathcal{P}_1\setminus\mathcal{S}} := \mathcal{T}_{ext}(\mathcal{P}_1) \setminus \mathcal{T}_{ext}(\mathcal{S})$, and $\mathcal{T}_{\mathcal{S}\cap\mathcal{P}_1} := \mathcal{T}_{ext}(\mathcal{S}) \cap \mathcal{T}_{ext}(\mathcal{P}_1)$.

which can be connected to $\mathcal{P}_r$ or $\mathcal{Q}_r$. By applying the case "$k = 1$" k times, we obtain that:

$$
\begin{aligned}
\mathcal{S} \mid \mathcal{P}_1 \mid \cdots \mid \mathcal{P}_k &= \mathcal{H}_1 \mid \mathcal{P}_1 && \text{(syntactic reordering of subsystems)} \\
&\cong \mathcal{H}_1 \mid \mathcal{Q}_1 && \text{(case "}k = 1\text{", with } \mathcal{H}_1 \text{ playing the role of } \mathcal{S}) \\
&= \mathcal{H}_2 \mid \mathcal{P}_2 && \text{(syntactic reordering of subsystems)} \\
&\cong \mathcal{H}_2 \mid \mathcal{Q}_2 && \text{(case "}k = 1\text{")} \\
&\vdots \\
&= \mathcal{H}_k \mid \mathcal{P}_k && \text{(syntactic reordering of subsystems)} \\
&\cong \mathcal{H}_k \mid \mathcal{Q}_k && \text{(case "}k = 1\text{")} \\
&= \mathcal{S} \mid \mathcal{Q}_1 \mid \cdots \mid \mathcal{Q}_k \ .
\end{aligned}
$$

We note that in the above argument, the equalities (=) hold true up to syntactic reordering of subsystems. Furthermore, all these systems are environmentally bounded: By assumption $\mathcal{S} \mid \mathcal{P}_1 \mid \cdots \mid \mathcal{P}_k = \mathcal{H}_1 \mid \mathcal{P}_1$ is environmentally bounded. For all $\mathcal{H}_r \mid \mathcal{Q}_r$, $r \geq 1$, this follows from the case "$k = 1$" and that $\mathcal{H}_r \mid \mathcal{P}_r = \mathcal{H}_{r-1} \mid \mathcal{Q}_{r-1}$ is environmentally bounded. This concludes the proof of the theorem. □

5.2 Composition Theorem for Unbounded Self-Composition

We now present the composition theorem for the unbounded self-composition of systems that are not aware of the session identifiers used to address the different sessions. As mentioned before, below we will present a composition theorem, as a corollary of this composition theorem, where systems are aware of the session identifiers.

We first need to introduce the notion of a session version of a system.

5.2.1 Session Versions of Systems

To model multiple copies/sessions of a system one can use the mode CheckAddress of IITMs combined with session identifiers (SIDs) as follows.

Given a system $\mathcal{Q}$, we define its session version $\underline{\mathcal{Q}}$, which allows us to address different copies of (IITMs occurring in) $\mathcal{Q}$ by a particular SID. We first define the session version of a single IITM.

The *session version* $\underline{M}$ of an IITM M is an IITM that exactly simulates M except that all messages received have to be prefixed by a particular SID (i.e., in mode CheckAddress the IITM $\underline{M}$ will reject all messages not prefixed by the particular SID) and all messages sent out are prefixed by this SID. The SID

$\underline{M}$ will use is the one with which $\underline{M}$ is first activated (hence, in the first activation, $\underline{M}$ will accept the incoming message in mode CheckAddress and then store the SID). More precisely, $\underline{M}$ behaves as follows in mode CheckAddress and Compute, respectively:

- When activated in mode CheckAddress, $\underline{M}$ does the following: If $\underline{M}$ has never been activated before, it accepts an incoming message m' only if the following is satisfied: (i) m' is of the form (s, m)[15] where s is interpreted as an SID, and (ii) the simulated M accepts m in mode CheckAddress. (In mode Compute, s will be stored by $\underline{M}$.) If $\underline{M}$ was activated before, then $\underline{M}$ will accept an incoming message m' only if the following is satisfied: (i) m' is of the form (s, m) where s is the SID that was stored in the first activation (in mode Compute), and (ii) m is accepted by the simulated M in mode CheckAddress.
- When activated in mode Compute, $\underline{M}$ does the following: If $\underline{M}$ has never been activated before (in mode Compute), then by the definition of $\underline{M}$ in mode CheckAddress it follows that the incoming message is of the form (s, m). Now, $\underline{M}$ first stores s and then simulates M on input m in mode Compute. If M produces output, say m', then $\underline{M}$ sends the output (s, m'), i.e., prefixes m' with s. If $\underline{M}$ was activated before (in mode Compute), then by definition of $\underline{M}$ in mode CheckAddress it follows that the incoming message is of the form (s, m) where s is the SID that was stored in the first activation. Now, as before, $\underline{M}$ simulates M on input m in mode Compute and prefixes the output produced (if any) with s.

Note that the IITM M, simulated within $\underline{M}$, is not aware of the SID that is used to address (a copy of) M. In particular, it cannot use the SID as part of the messages it produces.

Now, the *session version* $\underline{\mathcal{Q}}$ of a system $\mathcal{Q}$ is obtained from $\mathcal{Q}$ by replacing every IITM occurring in $\mathcal{Q}$ by its session version. For instance, if $\mathcal{Q} = M_1 \,|\, !\, M_2$, then $\underline{\mathcal{Q}} = \underline{M_1} \,|\, !\, \underline{M_2}$.

While in session versions as defined above, SIDs are always prefixed to messages, it clearly does not matter where exactly the SIDs occur in a message. The CheckAddress mode, being an arbitrary deterministic polynomial computation, is flexible enough to allow for many variants; for example, SIDs could be appended instead of prefixed to messages. The results on session versions do not depend on specific details.

It is sometimes useful to define a session version of a single IITM or an entire system with respect to a domain of SIDs (parameterized by the security parameter). In mode CheckAddress only those SIDs would be accepted that belong to the specified domain. (Clearly, we need to require that it can be decided efficiently in mode CheckAddress whether an SID belongs to the domain.) With this, we could, for example, define a session version $\underline{M}$ of an IITM M which only accepts SIDs of the form (sid, pid), where pid denotes a party and sid identifies the session in which this party runs. Hence, in a run of the system $!\,\underline{M}$ (in some environment) all instances of $\underline{M}$ would have SIDs of this form. In statements involving session versions, such as composition theorems, details of how the domains of SIDs are chosen are typically not important, as long as they are chosen consistently. We therefore omit such details in the statements.

We now state a fundamental property about the runtime of session versions of two indistinguishable systems.

Lemma 9. *Let $\mathcal{P}$ and $\mathcal{Q}$ be protocol systems such that $!\,\underline{\mathcal{P}}$ and $\mathcal{Q}$ are environmentally bounded and $\mathcal{P} \cong \mathcal{Q}$. Then, $!\,\underline{\mathcal{Q}}$ is environmentally bounded.*

While this property might be expected, the proof of this theorem, which is presented in Appendix A and mainly follows ideas presented in [21], is non-trivial. On a high level, in the proof we replace single sessions of $\underline{\mathcal{P}}$ (in runs of the system $\mathcal{E} \,|\, !\,\underline{\mathcal{P}}$ for some environment $\mathcal{E}$) with sessions of $\underline{\mathcal{Q}}$ via a hybrid argument. We then argue that the runtime of each of the hybrid systems is negligibly close to the previous/next one, as otherwise one could distinguish $\mathcal{P}$ and $\mathcal{Q}$ via the runtime difference. While it is quite easy to establish such a negligible bound for each individual hybrid step, this is insufficient as there are polynomially many steps in the hybrid argument. Instead, we need a single negligible function that bounds *all steps*. This is actually the main difficulty in proving this lemma and requires elaborate constructions of environments that

[15] We assume that (s, m) is a bit string which encodes the concatenation of the bit strings s and m in such a way that given s and m, the bit string (s, m) is computable in polynomial time (in $|s| + |m|$), and that given (s, m) the components s and m can be retrieved in polynomial time.

permute instances and simulate random hybrid steps to obtain bounds that hold true for all hybrid steps. Once we have established such a bound, we directly obtain that $!\underline{\mathcal{Q}}$ is environmentally bounded because $!\underline{\mathcal{P}}$ is environmentally bounded.

5.2.2 The Composition Theorem for Session Versions

The following theorem says that if two systems are environmentally indistinguishable, then an unbounded number of copies of one system are environmentally indistinguishable from an unbounded number of copies of the other system, where to address different copies session versions are considered. In particular, SIDs are merely used as a means to address (IITMs belonging to) copies of systems. The systems (more precisely, the IITMs these systems consist of) are not and do not need to be aware of the SIDs that are used to address their copies, and the specific addressing mechanism used. As mentioned before, in Section 5.3 we will present a composition theorem, as a corollary of Theorem 5, where systems are aware of their own SID.

Theorem 5. *Let $\mathcal{P}$ and $\mathcal{Q}$ be protocol systems such that $!\underline{\mathcal{P}}$ and $\mathcal{Q}$ are environmentally bounded and $\mathcal{P} \cong \mathcal{Q}$. Then, $!\underline{\mathcal{P}} \cong !\underline{\mathcal{Q}}$ and $!\underline{\mathcal{Q}}$ is environmentally bounded*

Before we prove this theorem, let us mention that typically $\mathcal{P}$ and $\mathcal{Q}$ will be environmentally *strictly* bounded and that this is easy to verify. By Lemma 17 it then follows immediately that $!\underline{\mathcal{P}}$ is environmentally strictly bounded as well, and hence, environmentally bounded. So Theorem 5 can be applied directly to typical protocol systems without additional effort.

Proof of Theorem 5. By Lemma 9, we know that $!\underline{\mathcal{Q}}$ is environmentally bounded.

Now, let $\mathcal{E} \in \mathrm{Env}(!\underline{\mathcal{P}}) = \mathrm{Env}(!\underline{\mathcal{Q}})$. To prove the theorem, it remains to show that $\mathcal{E} \,|\, !\underline{\mathcal{P}} \equiv \mathcal{E} \,|\, !\underline{\mathcal{Q}}$. By Lemma 7, we may assume that $\mathcal{E}$ is a single IITM which, in mode CheckAddress, accepts all messages. Moreover, we may assume, without loss of generality, that $\mathsf{start}, \mathsf{decision} \in \mathcal{T}(\mathcal{E})$ and that $\mathcal{E}$ is such that every message m that $\mathcal{E}$ outputs (except if m is output on tape decision) is prefixed by some SID, i.e., $m = (s, m')$ for some bit strings s and m':[16] since $\mathcal{E}$ will only interact with session versions, messages not of the form (s, m') would be rejected by these session versions anyway.

Since $\mathcal{E}$ is universally bounded, it follows that there exists a polynomial $p_{\mathcal{E}}$ such that the number of different sessions (i.e., messages with distinct SIDs output by $\mathcal{E}$) is bounded from above by $p_{\mathcal{E}}(\eta + |a|)$ (where η is the security parameter and a is the external input). In particular, for every η, a and every run of $(\mathcal{E} \,|\, !\underline{\mathcal{P}})(1^\eta, a)$ or $(\mathcal{E} \,|\, !\underline{\mathcal{Q}})(1^\eta, a)$, there exist at most $p_{\mathcal{E}}(\eta + |a|)$ copies of $\underline{\mathcal{Q}}$ and $\underline{\mathcal{P}}$ in such a run.

The proof proceeds by a hybrid argument. In what follows, let $\underline{\mathcal{P}}'$ be the variant of $\underline{\mathcal{P}}$ obtained from $\underline{\mathcal{P}}$ by renaming every tape c occurring in $\underline{\mathcal{P}}$ as c'. Analogously, let $\underline{\mathcal{P}}''$ be obtained from $\underline{\mathcal{P}}$ by renaming every tape c occurring in $\underline{\mathcal{P}}$ as c''. Similarly for $\underline{\mathcal{Q}}'$ and $\underline{\mathcal{Q}}''$.

We define an IITM $\mathcal{E}_r$ (for every $r \in \mathbf{N}$) which basically simulates $\mathcal{E}$ and which will run in the system $\mathcal{E}_r \,|\, !\underline{\mathcal{P}}'' \,|\, !\underline{\mathcal{Q}}' \,|\, \mathcal{P}$ or $\mathcal{E}_r \,|\, !\underline{\mathcal{P}}'' \,|\, !\underline{\mathcal{Q}}' \,|\, \mathcal{Q}$, respectively. The first $r-1$ copies of the protocol invoked by $\mathcal{E}$ will be copies of $\underline{\mathcal{Q}}'$, the r-th copy will be the external system $\mathcal{P}$ or $\mathcal{Q}$, respectively, and the remaining copies will be copies of $\underline{\mathcal{P}}''$.

Formally, $\mathcal{E}_r$ is obtained from $\mathcal{E}$ as follows. (Recall that we assume that $\mathcal{E}$ is a single IITM which accepts every message in mode CheckAddress). The IITM $\mathcal{E}_r$ will always accept in mode CheckAddress. The behavior of $\mathcal{E}_r$ in mode Compute is specified next.

First, we need to make sure that $\mathcal{E}_r$ has the appropriate tapes to connect to the different entities. The IITM $\mathcal{E}$ already has tapes to connect to the external tapes of $\mathcal{P}$ and $\mathcal{Q}$. For each such tape c, we add to $\mathcal{E}_r$ a tape c' and c'' to connect to the external tapes of $\underline{\mathcal{Q}}'$ and $\underline{\mathcal{P}}''$, respectively.

Next, we need to specify how $\mathcal{E}_r$ redirects protocol invocations of $\mathcal{E}$ in the way described above: $\mathcal{E}_r$ keeps a list L of SIDs, which initially is empty, and the length l of the list, which initially is 0. By definition of $p_{\mathcal{E}}$, it will always hold that $l \leq p_{\mathcal{E}}(\eta + |a|)$. In the first activation with security parameter $\eta \in \mathbf{N}$ and external input $a \in \{0,1\}^*$, $\mathcal{E}_r$ starts a simulation of $\mathcal{E}$ with security parameter η and external input a. In particular, if $\mathcal{E}$ produces output, then so does $\mathcal{E}_r$, and if $\mathcal{E}_r$ receives input, then $\mathcal{E}$ is simulated with this input. However, as

[16] More formally, for every system $\mathcal{S} \in \mathrm{Con}(\mathcal{E})$ and all parameters η, a in every run of $(\mathcal{E} \,|\, \mathcal{S})(1^\eta, a)$ the system $\mathcal{E}$ should only output messages of the described form.

explained next, the behavior of $\mathcal{E}_r$ deviates from that of $\mathcal{E}$ when it comes to sending and receiving messages to the different copies of protocols.

1. If $\mathcal{E}$ produces output m on some external output tape c to $\underline{\mathcal{P}}$ (and hence, $\underline{\mathcal{Q}}$) prefixed with s, then $\mathcal{E}_r$ checks whether s occurs in L. If s does not occur in L, s is appended at the end of L and l is increased by 1. Let $j \in \{1, \ldots, l\}$ be the position where s occurs in L.
   (a) If $j < r$, then $\mathcal{E}_r$ writes m on tape c'.
   (b) If $j = r$, then $\mathcal{E}_r$ outputs m' on c where m' is a message such that $m = (s, m')$, i.e., s is removed from m.
   (c) If $j > r$, then $\mathcal{E}_r$ writes m on tape c''.
2. If $\mathcal{E}_r$ receives input on tape c'' where c'' is an external tape of $\underline{\mathcal{P}}''$ corresponding to an external tape c of $\underline{\mathcal{P}}$, then $\mathcal{E}_r$ behaves as $\mathcal{E}$ in case input was received on tape c.
3. If $\mathcal{E}_r$ receives input on tape c' where c' is an external tape of $\underline{\mathcal{Q}}'$ corresponding to an external tape c of $\underline{\mathcal{Q}}$, then $\mathcal{E}_r$ behaves as $\mathcal{E}$ in case input was received on tape c.
4. If $\mathcal{E}_r$ receives input m on tape c where c is an external tape of $\mathcal{P}$ (and hence, $\mathcal{Q}$), then $\mathcal{E}_r$ behaves as $\mathcal{E}$ in case input $(L[r], m)$ was received on tape c where $L[r]$ denotes the r-th entry of L. By construction, this entry exists in L (i.e., $r \leq l$) since $\mathcal{E}$ must have invoked the r-th copy.

We also consider a variant $\mathcal{E}_\$$ of $\mathcal{E}_r$ which in the first activation chooses $r \in \{1, \ldots, p_{\mathcal{E}}(\eta + |a|)\}$ uniformly at random and then behaves exactly like $\mathcal{E}_r$. Moreover, we consider the variant $\mathcal{E}_{p_\mathcal{E}}$ of $\mathcal{E}_r$ which uses $r = p_{\mathcal{E}}(\eta + |a|)$.

We define the following hybrid systems, for every $r \in \mathbf{N}$:

$$\begin{aligned} \mathcal{H}_r &:= \mathcal{E}_r \,|\, !\,\underline{\mathcal{P}}'' \,|\, !\,\underline{\mathcal{Q}}' \text{ and} \\ \mathcal{H}_{p_\mathcal{E}} &:= \mathcal{E}_{p_\mathcal{E}} \,|\, !\,\underline{\mathcal{P}}'' \,|\, !\,\underline{\mathcal{Q}}' \text{ and} \\ \mathcal{H}_\$ &:= \mathcal{E}_\$ \,|\, !\,\underline{\mathcal{P}}'' \,|\, !\,\underline{\mathcal{Q}}' \ , \end{aligned}$$

which can be connected to $\mathcal{P}$ (and hence $\mathcal{Q}$). The system $\mathcal{H}_r$ is illustrated in Figure 3.

By construction, for every $r \in \mathbf{N}$, the systems $\mathcal{E} \,|\, !\,\underline{\mathcal{P}}$ and $\mathcal{H}_1 \,|\, \mathcal{P}$, $\mathcal{H}_r \,|\, \mathcal{Q}$ and $\mathcal{H}_{r+1} \,|\, \mathcal{P}$, and $\mathcal{E} \,|\, !\,\underline{\mathcal{Q}}$ and $\mathcal{H}_{p_\mathcal{E}} \,|\, \mathcal{Q}$, respectively, behave exactly the same. In particularly, for all $r \in \mathbf{N}$, we have that:

$$\mathcal{E} \,|\, !\,\underline{\mathcal{P}} \equiv_0 \mathcal{H}_1 \,|\, \mathcal{P} \ , \tag{7}$$

$$\mathcal{H}_r \,|\, \mathcal{Q} \equiv_0 \mathcal{H}_{r+1} \,|\, \mathcal{P} \ , \text{ and} \tag{8}$$

$$\mathcal{E} \,|\, !\,\underline{\mathcal{Q}} \equiv_0 \mathcal{H}_{p_\mathcal{E}} \,|\, \mathcal{Q} \ . \tag{9}$$

For (7) we use that $\mathcal{P}$ is a protocol system. In particular, we use property (ii) of protocol systems (see Definition 11). If this property were not satisfied, i.e., $\mathcal{P}$ contains an IITM M which is not in the scope of a bang but which could reject a message in mode CheckAddress, the following could happen. In a run of $(\mathcal{H}_1 \,|\, \mathcal{P})(1^\eta, a)$ a message is sent to M, but it is rejected by M (in mode CheckAddress). Then, since M is not in the scope of a bang, no new copy of M will be generated. Conversely, if in a run of $\mathcal{E} \,|\, !\,\underline{\mathcal{P}}$ a message is sent to a copy of the session version $\underline{M}$ of M prefixed with a SID generated by $\mathcal{E}$ and the simulated M in $\underline{M}$ would reject the message, then it could happen that a new copy of $\underline{M}$ is generated (since $\underline{M}$ is in the scope of a bang in $\mathcal{E} \,|\, !\,\underline{\mathcal{P}}$) which then would not have a corresponding entity in a run of the system $(\mathcal{H}_1 \,|\, \mathcal{P})(1^\eta, a)$. In short, by property (ii) of protocol systems it is guaranteed that for IITMs that do not occur in the scope of a bang in $\mathcal{P}$ only at most one copy is generated per SID in the run of $\mathcal{E} \,|\, !\,\underline{\mathcal{P}}$. A similar argument is used to prove (8) and (9).

We now show that $!\,\underline{\mathcal{P}}'' \,|\, !\,\underline{\mathcal{Q}}'$ is environmentally bounded: Since $!\,\underline{\mathcal{P}}'$ is environmentally bounded, it is easy to see that $!\,\underline{\mathcal{P}''} \,|\, !\,\underline{\mathcal{P}}'$ is environmentally bounded too, and in particular $\mathcal{P}'' \,|\, \mathcal{P}'$ is environmentally bounded. By

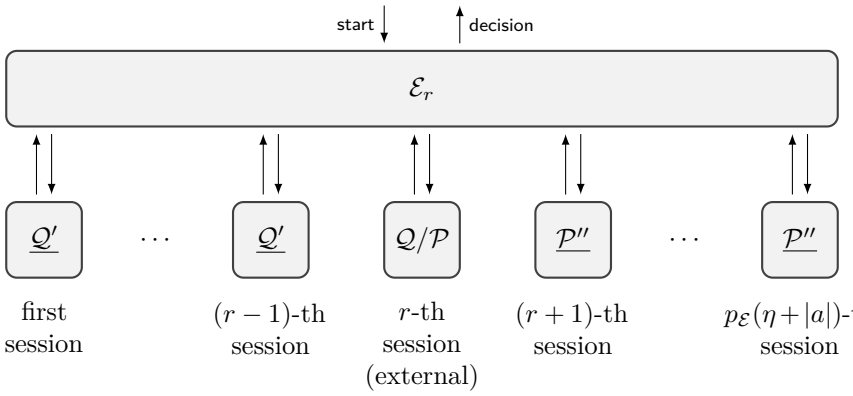

Figure 3: The hybrid system $\mathcal{H}_r = \mathcal{E}_r \,|\, !\,\underline{\mathcal{Q}'} \,|\, !\,\underline{\mathcal{P}''}$ composed with the external session $\mathcal{Q}$ or $\mathcal{P}$, respectively.

Theorem 4 (with $\mathcal{S} = \mathcal{P}''$ and $\mathcal{P}' \cong \mathcal{Q}'$), we obtain that $\mathcal{P}'' \,|\, \mathcal{P}' \cong \mathcal{P}'' \,|\, \mathcal{Q}'$ and that $\mathcal{P}'' \,|\, \mathcal{Q}'$ is environmentally bounded. Then, by Lemma 9, we conclude that $!\,\underline{\mathcal{P}''} \,|\, !\,\underline{\mathcal{Q}'}$ is environmentally bounded.

From this it easily follows that $!\,\underline{\mathcal{P}''} \,|\, !\,\underline{\mathcal{Q}'} \,|\, \mathcal{P}$ and $!\,\underline{\mathcal{P}''} \,|\, !\,\underline{\mathcal{Q}'} \,|\, \mathcal{Q}$ are environmentally bounded as well. Since $\mathcal{P} \cong \mathcal{Q}$, by Theorem 4 (with $\mathcal{S} = !\,\underline{\mathcal{P}''} \,|\, !\,\underline{\mathcal{Q}'}$), we now obtain that $!\,\underline{\mathcal{P}''} \,|\, !\,\underline{\mathcal{Q}'} \,|\, \mathcal{P} \cong !\,\underline{\mathcal{P}''} \,|\, !\,\underline{\mathcal{Q}'} \,|\, \mathcal{Q}$.

It is easy to see that $\mathcal{E}_\$$, $\mathcal{E}_{p_{\mathcal{E}}(\eta+|a|)}$, and $\mathcal{E}_r$, for every $r \in \mathbf{N}$, are universally bounded. So, we obtain that $\mathcal{H}_\$ \,|\, \mathcal{P} \equiv \mathcal{H}_\$ \,|\, \mathcal{Q}$. Hence, there exists a negligible function f such that for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$:

$$\begin{aligned}
f(\eta,a) &\geq |\mathrm{Prob}[(\mathcal{H}_\$ \,|\, \mathcal{P})(1^\eta,a)=1] - \mathrm{Prob}[(\mathcal{H}_\$ \,|\, \mathcal{Q})(1^\eta,a)=1]| \\
&= \frac{1}{p_{\mathcal{E}}(\eta+|a|)} \cdot \left| \sum_{r=1}^{p_{\mathcal{E}}(\eta+|a|)} \mathrm{Prob}[(\mathcal{H}_r \,|\, \mathcal{P})(1^\eta,a)=1] - \mathrm{Prob}[(\mathcal{H}_r \,|\, \mathcal{Q})(1^\eta,a)=1] \right| \\
&\overset{(8)}{=} \frac{1}{p_{\mathcal{E}}(\eta+|a|)} \cdot \left| \mathrm{Prob}[(\mathcal{H}_1 \,|\, \mathcal{P})(1^\eta,a)=1] - \mathrm{Prob}[(\mathcal{H}_{p_{\mathcal{E}}(\eta+|a|)} \,|\, \mathcal{Q})(1^\eta,a)=1] \right| \\
&\overset{(7),(9)}{=} \frac{1}{p_{\mathcal{E}}(\eta+|a|)} \cdot |\mathrm{Prob}[(\mathcal{E} \,|\, !\,\underline{\mathcal{P}})(1^\eta,a)=1] - \mathrm{Prob}[(\mathcal{E} \,|\, !\,\underline{\mathcal{Q}})(1^\eta,a)=1]| \quad .
\end{aligned}$$

We conclude that $\mathcal{E} \,|\, !\,\underline{\mathcal{P}} \equiv \mathcal{E} \,|\, !\,\underline{\mathcal{Q}}$. □

We note that by iteratively applying this composition theorem and Theorem 4, one can establish environmental indistinguishability for more and more complex systems in a modular way.

5.3 Composition Theorem for Unbounded Self-Composition of SID Dependent Systems

As explained before, in Theorem 5 SIDs are used merely as a means to address (IITMs belonging to) copies/sessions of systems. In particular, a session of a system is not aware of the SID that is used to address it, and the specific addressing mechanism employed. We now present a composition theorem for the unbounded self-composition of a system where sessions *are* aware of their SIDs. This theorem is basically a corollary of Theorem 5.

We first need to generalize the notion of a session version of an IITM and a system and introduce further notions in the context of session versions.

5.3.1 Generalized Session Versions

A function $\sigma\colon \{0,1\}^* \times \mathcal{T} \to \{0,1\}^* \cup \{\bot\}$ (where $\mathcal{T}$ is a set of tape names) is called a *session identifier (SID) function* if it is computable in polynomial time (in the length of its input). Intuitively, an SID function

assigns an SID to a message w.r.t. some tape (or $\bot$, if the message does not have an SID). For example, $\sigma_0(m,c) := 0$, for all m, c, assigns the SID 0 to every message. The following function takes the prefix of a message as its SID: $\sigma_{\text{prefix}}(m,c) := s$ if $m = (s, m')$ for some s, m' and $\sigma_{\text{prefix}}(m,c) := \bot$ otherwise, for all m, c. Clearly, many more examples are conceivable. We note that an SID function, besides the message, also gets a tape name as input because the way SIDs are extracted from messages may depend on the tape on which a message is received; this gives even more flexibility, although in most cases an SID of a message will be determined independently of the tape on which the message is received.

We now define the notion of a σ-session version, which generalizes the notion of session versions introduced in Section 5.2.1. Intuitively, a machine M is a σ-session version, if M in mode CheckAddress accepts only messages that have the same SID w.r.t. σ and M outputs only messages with the same SID. For a system to be a σ-session version, we require that all machines in that system are σ-session versions. More precisely, σ-session versions are defined as follows:

Definition 13. Let σ be an SID function and let M be an IITM such that $\mathcal{T}(M) \subseteq \mathcal{T}$ where $\mathcal{T}$ is the set of tape names σ is defined on (i.e., σ is defined on all names of input and output tapes of M). Then, M is a *σ-session machine* (also called a *σ-session version*) if for every system $\mathcal{S}$ such that $\mathcal{S}$ and M are connectable the following conditions are satisfied for every η, a and every run ρ of $(\mathcal{S} \mid M)(1^\eta, a)$:

1. Whenever M is activated in ρ in mode CheckAddress with an input message m on tape c, then M rejects m if $\sigma(m,c) = \bot$.
2. If the first input message that M accepted in ρ in mode CheckAddress is m_0 on tape c_0 and (later) M is activated in mode CheckAddress in ρ with an input message m on tape c, then M rejects m if $\sigma(m,c) \neq \sigma(m_0,c_0)$.
3. Whenever M outputs a message m on tape c in ρ in mode Compute, then $\sigma(m,c) = \sigma(m_0,c_0)$ (where the first accepted message was m_0 on tape c_0, see above).

A system $\mathcal{Q}$ is a *σ-session system/version* if every IITM occurring in $\mathcal{Q}$ is a σ-session version.

We emphasize that by 2. above, the fact that $\sigma(m,c) = \sigma(m_0,c_0) \neq \bot$ does not mean that a machine has to accept m on c. This is only a necessary condition for the machine to be able to accept m on c in mode CheckAddress. In other words, every system which is a σ-session version can accept "less" messages than "suggested" by σ; in particular, every system $\mathcal{S}$ is a σ_0-session version, since σ_0 allows a machine to accept every message.

The following two lemmas capture some more basic properties of σ-session versions. The first lemma says that if a system is a σ-session version, then it is also a σ'-session version for a function σ' that is at least as permissive as σ.

Lemma 10. *Let σ and σ' be SID functions such that for all $m, m' \in \{0,1\}^*, c, c' \in \mathcal{T}$: $\sigma'(m,c) = \sigma'(m',c') \neq \bot$ if $\sigma(m,c) = \sigma(m',c') \neq \bot$. Then, every σ-session system is a σ'-session system.*

The following lemma shows how standard session versions, as introduced in Section 5.2.1, and σ-session versions can be combined.

Lemma 11. *Let σ be an SID function and $\mathcal{S}$ be a σ-session system. Then, $!\underline{\mathcal{S}}$ is a σ'-session system where for all $m \in \{0,1\}^*, c \in \mathcal{T}$:*

$$\sigma'(m,c) := \begin{cases} (s, \sigma(m',c)) & \exists s, m' \colon m = (s,m') \wedge \sigma(m',c) \neq \bot, \\ \bot & \textit{otherwise} \ . \end{cases}$$

Combining the two lemmas above and using that every system is a σ_0-session version, we obtain that the multi-session version $!\underline{\mathcal{S}}$ of every system $\mathcal{S}$ is a σ_{prefix}-session version. Moreover, applying Lemma 11 to $!\underline{\mathcal{S}}$ yields that the multi-session version of the multi-session version of every system $\mathcal{S}$, i.e., $!\underline{!\underline{\mathcal{S}}}$, is a $\sigma^{(2)}_{\text{prefix}}$-session version where $\sigma^{(2)}_{\text{prefix}}(m,c) := (s,s')$ if $m = (s,(s',m'))$ for some s, s', m' and $\sigma^{(2)}_{\text{prefix}}(m,c) := \bot$ otherwise, for

all m, c. Note that by Lemma 10 every $\sigma^{(2)}_{\text{prefix}}$-session version (e.g., $!\underline{\underline{\mathcal{S}}}$) is a σ_{prefix}-session version. This is analogous to the (standard) session versions from Section 5.2.1: $!\underline{\underline{\mathcal{S}}}$ is a session version itself.

Now, given two systems $\mathcal{P}$ and $\mathcal{Q}$ which both are σ-session versions, the composition theorem basically states that if $\mathcal{P}$ and $\mathcal{Q}$ are single-session indistinguishable, i.e., $\mathcal{P}$ and $\mathcal{Q}$ are environmentally indistinguishable for environments that call a single session of $\mathcal{P}/\mathcal{Q}$ only, then $\mathcal{P}$ and $\mathcal{Q}$ are environmentally indistinguishable ($\mathcal{P} \cong \mathcal{Q}$). We now formalize "single-session indistinguishability". We first need to formalize systems that invoke only a single session of another system. Such systems are called σ-single session and they output messages with the same SID only. More precisely:

Definition 14. Let σ be an SID function. A system $\mathcal{Q}$ is *σ-single session* if for every system $\mathcal{S}$ such that $\mathcal{S}$ and $\mathcal{Q}$ are connectable the following is true for every η, a and in every run ρ of $(\mathcal{S} \mid \mathcal{Q})(1^\eta, a)$: Let $m_0 \neq \varepsilon$ (where ε is the empty bit string) be the first message output by $\mathcal{Q}$ on some external tape c_0 (except the decision tape) in ρ. Then $\sigma(m_0, c_0) \neq \bot$ and every message $m \neq \varepsilon$ output by $\mathcal{Q}$ on an external tape c (except the decision tape) in ρ satisfies $\sigma(m, c) = \sigma(m_0, c_0)$.

Let σ be an SID function. Given a system $\mathcal{S}$, by $\text{Env}_{\sigma\text{-single}}(\mathcal{S})$ we denote the set of all systems $\mathcal{E} \in \text{Env}(\mathcal{S})$ such that $\mathcal{E}$ is σ-single session, i.e., $\text{Env}_{\sigma\text{-single}}(\mathcal{S})$ is the set of all σ-single session environmental systems that can be connected to $\mathcal{S}$.

Definition 15. Let σ be an SID function. Two systems $\mathcal{P}$ and $\mathcal{Q}$ are called *σ-environmentally indistinguishable* or *indistinguishable w.r.t. σ-single session environments*, denoted by $\mathcal{P} \cong_{\sigma\text{-single}} \mathcal{Q}$, if and only if

1. $\mathcal{P}$ and $\mathcal{Q}$ are compatible and
2. $\mathcal{E} \mid \mathcal{P} \equiv \mathcal{E} \mid \mathcal{Q}$ for all $\mathcal{E} \in \text{Env}_{\sigma\text{-single}}(\mathcal{P})$.

We will also need the following definition.

Definition 16. Let σ be an SID function. A system $\mathcal{S}$ is *σ-environmentally (almost) bounded* or *environmentally bounded w.r.t. σ-single session environments* if for every $\mathcal{E} \in \text{Env}_{\sigma\text{-single}}(\mathcal{S})$: $\mathcal{E} \mid \mathcal{S}$ is almost bounded.

Clearly, every environmentally bounded system is σ-environmentally bounded for every SID function σ.

5.3.2 A Composition Theorem for σ-Session Versions

We are now able to formulate the composition theorem for the unbounded self-composition of systems that may depend on their SID:

Theorem 6. *Let σ be an SID function. Let $\mathcal{P}$ and $\mathcal{Q}$ be two protocol systems such that $\mathcal{P}$ and $\mathcal{Q}$ are σ-session versions, $!\underline{\mathcal{P}}$ is environmentally bounded, $\mathcal{Q}$ is σ-environmentally bounded, and $\mathcal{P} \cong_{\sigma\text{-single}} \mathcal{Q}$. Then, $\mathcal{P} \cong \mathcal{Q}$ and $\mathcal{Q}$ is environmentally bounded.*

We note that a stronger variant of the above theorem, where it is assumed that only $\mathcal{P}$ (instead of $!\underline{\mathcal{P}}$) is environmentally bounded, also holds (in this case, the claim still is that $\mathcal{P} \cong \mathcal{Q}$ and $\mathcal{Q}$ is environmentally bounded). Such a theorem can be proven by redoing the proofs of Theorem 5 and Lemma 9.[17] Here we want to prove Theorem 6 as a corollary of Theorem 5 and this requires the additional assumption.[18] However, as discussed below Theorem 5, typically $\mathcal{P}$ is environmentally strictly bounded and in this case, by Lemma 17, it follows immediately that $!\underline{\mathcal{P}}$ is environmentally bounded.

[17] As for the lemma, one assumes that (i) $\mathcal{P}$ and $\mathcal{Q}$ are σ-session versions, (ii) $\mathcal{P}$ is environmentally bounded, (iii) $\mathcal{Q}$ is σ-environmentally bounded, and (iv) $\mathcal{P} \cong_{\sigma\text{-single}} \mathcal{Q}$. For Lemma 9 the statement then is that $\mathcal{Q}$ is environmentally bounded.

[18] In general, assuming that $!\underline{\mathcal{P}}$ is environmentally bounded is a stronger condition than just assuming that $\mathcal{P}$ is environmentally bounded (see Lemma 19).

Proof of Theorem 6. The basic idea of the proof is as follows: We first define single IITMs $[\mathcal{P}]_\sigma$ and $[\mathcal{Q}]_\sigma$ which, in every environment, simulate a single session of $\mathcal{P}$ and $\mathcal{Q}$ (w.r.t. σ), respectively. Since $\mathcal{P} \cong_{\sigma\text{-single}} \mathcal{Q}$ we obtain $[\mathcal{P}]_\sigma \cong [\mathcal{Q}]_\sigma$. Then, by the composition theorem (Theorem 5), we obtain that $!\underline{[\mathcal{P}]_\sigma} \cong !\underline{[\mathcal{Q}]_\sigma}$. Now, from $!\underline{[\mathcal{P}]_\sigma}$ and $!\underline{[\mathcal{Q}]_\sigma}$ we can conclude that $\mathcal{P} \cong \mathcal{Q}$: An environment $\mathcal{E}$ for $\mathcal{P}$ and $\mathcal{Q}$ can be simulated by an environment $\mathcal{E}'$ which simply prefixes every message m output on tape c with $\sigma(m, c)$ if $\sigma(m, c) \neq \bot$.

Following this idea, we first define $[\mathcal{P}]_\sigma$. As in the proof of Lemma 6, we construct $[\mathcal{P}]_\sigma$ as an IITM which is compatible with $\mathcal{P}$ and accepts every message in mode CheckAddress. However, it only simulates a single session of $\mathcal{P}$ w.r.t. σ. More specifically, $[\mathcal{P}]_\sigma$ (in any run with any system) does the following: If $[\mathcal{P}]_\sigma$ receives a message m on tape c with $\sigma(m, c) = \bot$, then $[\mathcal{P}]_\sigma$ ends its current activation with empty output. Let m_0 be the first message that $[\mathcal{P}]_\sigma$ receives on some external tape c_0 with $\sigma(m_0, c_0) \neq \bot$, then $[\mathcal{P}]_\sigma$ simulates $\mathcal{P}$ with input m_0 on c_0 and if $\mathcal{P}$ produces output m' on some external tape c', $[\mathcal{P}]_\sigma$ produces output m' on c'. Now, whenever later $[\mathcal{P}]_\sigma$ receives a message m on tape c with $\sigma(m, c) \neq \sigma(m_0, c_0)$, then $[\mathcal{P}]_\sigma$ ends its current activation with empty output. Otherwise, i.e., if $\sigma(m, c) = \sigma(m_0, c_0)$, then $[\mathcal{P}]_\sigma$ simulates $\mathcal{P}$ with input m on c as above.

By construction of $[\mathcal{P}]_\sigma$ and since $\mathcal{P}$ is a σ-session version, it is easy to see that for every $\mathcal{E} \in \mathrm{Env}_{\sigma\text{-single}}(\mathcal{P})$ we have that $\mathcal{E} \,|\, \mathcal{P} \equiv_0 \mathcal{E} \,|\, [\mathcal{P}]_\sigma$. (This does not necessarily hold for all $\mathcal{E} \in \mathrm{Env}(\mathcal{P})$.) Because $!\underline{\mathcal{P}}$ is environmentally bounded by assumption, and hence, in particular $\mathcal{P}$ is environmentally bounded, it is not hard to see that $[\mathcal{P}]_\sigma$ is environmentally bounded. In fact, to conclude that $[\mathcal{P}]_\sigma$ is environmentally bounded it suffices to assume that $\mathcal{P}$ is σ-environmentally bounded, since $[\mathcal{P}]_\sigma$ simulates only a single session of $\mathcal{P}$.

Just as $[\mathcal{P}]_\sigma$, we define $[\mathcal{Q}]_\sigma$. By assumption, $\mathcal{Q}$ is a σ-session version and it is σ-environmentally bounded. Now, just as in the case of $[\mathcal{P}]_\sigma$, we obtain that (i) $\mathcal{E} \,|\, \mathcal{Q} \equiv_0 \mathcal{E} \,|\, [\mathcal{Q}]_\sigma$ for every $\mathcal{E} \in \mathrm{Env}_{\sigma\text{-single}}(\mathcal{Q}) = \mathrm{Env}_{\sigma\text{-single}}(\mathcal{P})$ and (ii) $[\mathcal{Q}]_\sigma$ is environmentally bounded.

Because $\mathcal{P} \cong_{\sigma\text{-single}} \mathcal{Q}$, it now easily follows that $\mathcal{E} \,|\, [\mathcal{P}]_\sigma \equiv \mathcal{E} \,|\, [\mathcal{Q}]_\sigma$ for all $\mathcal{E} \in \mathrm{Env}(\mathcal{P})$, and thus,

$$[\mathcal{P}]_\sigma \cong [\mathcal{Q}]_\sigma \ . \tag{10}$$

By assumption, $!\underline{\mathcal{P}}$ is environmentally bounded. It is not hard to see that this implies that $!\underline{[\mathcal{P}]_\sigma}$ is environmentally bounded too.[19] By Lemma 9 it follows that $!\underline{[\mathcal{Q}]_\sigma}$ is environmentally bounded. Moreover, by Theorem 5 we obtain:

$$!\underline{[\mathcal{P}]_\sigma} \cong !\underline{[\mathcal{Q}]_\sigma} \ . \tag{11}$$

We use (11) to show that

$$\mathcal{P} \cong \mathcal{Q} \ . \tag{12}$$

Let $\mathcal{E} \in \mathrm{Env}(\mathcal{P})$. By Lemma 7, we may assume that $\mathcal{E}$ is a single IITM which accepts every message in mode CheckAddress. Furthermore, without loss of generality, we may assume that $\mathsf{start}, \mathsf{decision} \in \mathcal{T}(\mathcal{E})$ and that $\mathcal{E}$ only outputs messages m on tape c with $\sigma(m, c) \neq \bot$, for every tape $c \neq \mathsf{decision}$: since $\mathcal{P}$ and $\mathcal{Q}$ are σ-session versions, they would reject any message m on tape c with $\sigma(m, c) = \bot$ anyway. We may also assume that on tape decision, $\mathcal{E}$ only outputs 0 or 1.

Now, to show that $\mathcal{E} \,|\, \mathcal{P} \equiv \mathcal{E} \,|\, \mathcal{Q}$, we define an IITM $\mathcal{E}' \in \mathrm{Env}(!\underline{[\mathcal{P}]_\sigma})$. The external tapes of $\mathcal{E}'$ are the external tapes of $\mathcal{E}$. In mode CheckAddress, $\mathcal{E}'$ accepts all messages. In mode Compute, $\mathcal{E}'$ simulates $\mathcal{E}$ as follows:

- If $\mathcal{E}'$ receives some input m (possibly empty input) on the tape start, then $\mathcal{E}'$ simulates $\mathcal{E}$ on input m on start. Otherwise, if $\mathcal{E}'$ receives some input of shape (s, m) on some tape c with $\sigma(m, c) = s \neq \bot$, then $\mathcal{E}'$ simulates $\mathcal{E}$ on input m on c.
- If the simulated $\mathcal{E}$ outputs some message m on the tape decision, then $\mathcal{E}'$ outputs m on decision. Otherwise, if the simulated $\mathcal{E}$ outputs some message m on some tape $c \neq \mathsf{decision}$ with $\sigma(m, c) \neq \bot$, then $\mathcal{E}'$ outputs $(\sigma(m, c), m)$ on c.

[19] To conclude that $!\underline{[\mathcal{P}]_\sigma}$ is environmentally bounded, the assumption that $!\underline{\mathcal{P}}$ is environmentally bounded is required. Just assuming that $\mathcal{P}$ is environmentally bounded would not be enough.

- Otherwise, i.e., $\mathcal{E}'$ receives some other input or the simulated $\mathcal{E}$ produces different output, $\mathcal{E}'$ produces output "error"$\notin \{0,1\}$ on decision.

Since $\mathcal{E}$ never outputs messages with $\sigma(m,c) = \bot$ (expect maybe on tape decision) and by definition of $\mathcal{E}'$, $!\underline{[\mathcal{P}]_\sigma}$, and $!\underline{[\mathcal{Q}]_\sigma}$, it is easy to see that for all η, a:

$$\text{Prob}\left[(\mathcal{E}' \,|\, !\underline{[\mathcal{P}]_\sigma})(1^\eta, a) = \text{“error”}\right] = \text{Prob}\left[(\mathcal{E}' \,|\, !\underline{[\mathcal{Q}]_\sigma})(1^\eta, a) = \text{“error”}\right] = 0 \ .$$

Therefore it is easy to define a bijection between the runs of $(\mathcal{E}' \,|\, !\underline{[\mathcal{P}]_\sigma})(1^\eta, a)$ and $(\mathcal{E} \,|\, \mathcal{P})(1^\eta, a)$ as well as between the runs of $(\mathcal{E}' \,|\, !\underline{[\mathcal{Q}]_\sigma})(1^\eta, a)$ and $(\mathcal{E} \,|\, \mathcal{Q})(1^\eta, a)$. Hence, we obtain:

$$\mathcal{E}' \,|\, !\underline{[\mathcal{P}]_\sigma} \equiv_0 \mathcal{E} \,|\, \mathcal{P} \quad \text{and} \quad \mathcal{E}' \,|\, !\underline{[\mathcal{Q}]_\sigma} \equiv_0 \mathcal{E} \,|\, \mathcal{Q} \ . \tag{13}$$

Since $\mathcal{E}$ is universally bounded, it follows easily that $\mathcal{E}'$ is universally bounded too. Thus, $\mathcal{E}' \in \text{Env}(!\underline{[\mathcal{P}]_\sigma})$. By (11), we can conclude that $\mathcal{E}' \,|\, !\underline{[\mathcal{P}]_\sigma} \equiv \mathcal{E}' \,|\, !\underline{[\mathcal{Q}]_\sigma}$. It now follows that

$$\mathcal{E} \,|\, \mathcal{P} \overset{(13)}{\equiv_0} \mathcal{E}' \,|\, !\underline{[\mathcal{P}]_\sigma} \equiv \mathcal{E}' \,|\, !\underline{[\mathcal{Q}]_\sigma} \overset{(13)}{\equiv_0} \mathcal{E} \,|\, \mathcal{Q} \ .$$

With the bijection between runs of $\mathcal{E}' \,|\, !\underline{[\mathcal{Q}]_\sigma}(1^\eta, a)$ and $(\mathcal{E} \,|\, \mathcal{Q})(1^\eta, a)$ and the fact that $!\underline{[\mathcal{Q}]_\sigma}$ is environmentally bounded, it also immediately follows that $\mathcal{Q}$ is environmentally bounded. □

We note that under the assumptions of Theorem 6, one can even show that $!\underline{\mathcal{Q}}$ is environmentally bounded, using that $!\underline{[\mathcal{Q}]_\sigma}$ is environmentally bounded.

Clearly, by iteratively applying Theorems 4, 5, and 6, one obtains, in a modular way, environmental indistinguishability for more and more complex systems.

6 Universal Composability Security Notions

In the literature, several universal composability security notions have been proposed: universal composability (UC) [5,36], dummy UC [5], black-box simulatability [36], strong simulatability [24], and reactive simulatability [36].

While intuitively one would expect these notions to be equivalent (where for reactive simulatability this requires non-uniform environments/environments with external input), this is not the case in all models. In particular, it is not true in Canetti's UC model, due to the incompleteness of the dummy adversary (see Section 4.5 and Remark 12).

In this section, the mentioned notions are defined and it is shown that these notions are equivalent in the IITM model. We also state further basic properties. But first we need to introduce some notation and terminology.

6.1 Further Notation and Terminology

To define the universal composability security notions we need the following kinds of systems: environmental, adversarial systems, and protocol systems. While we have already defined environmental and protocol systems (see Definitions 8 and 11), we still need to introduce adversarial systems and the way these systems are connected.

6.1.1 Network and I/O Tapes

To define how the different entities (environments, protocols, adversaries/simulators) connect to each other, we partition the set of external tapes of every system $\mathcal{Q}$ into *network* and *I/O tapes*. We denote the set of (external) network tapes of $\mathcal{Q}$ by $\mathcal{T}^{net}_{ext}(\mathcal{Q})$ and the set of (external) I/O tapes of $\mathcal{Q}$ by $\mathcal{T}^{io}_{ext}(\mathcal{Q})$. Each of the sets $\mathcal{T}^{net}_{ext}(\mathcal{S})$ and $\mathcal{T}^{io}_{ext}(\mathcal{S})$ is further partitioned into input and output tapes. We denote by $\mathcal{T}^{net}_{in}(\mathcal{Q})$, $\mathcal{T}^{net}_{out}(\mathcal{Q})$,

$\mathcal{T}^{io}_{in}(\mathcal{Q})$, and $\mathcal{T}^{io}_{out}(\mathcal{Q})$ the set of network input and output tapes and the set of I/O input and output tapes, respectively.

We assume that network tapes are named differently from I/O tapes also across different systems. This can easily be enforced, for instance, by using different prefixes for the different kind of tapes, such as net for network tapes and io for I/O tapes. By this, we obtain the following:

(i) If $\mathcal{P}$ and $\mathcal{Q}$ are compatible, then they also agree on the type (network or I/O) of their external tapes, i.e., $\mathcal{T}^{net}_{in}(\mathcal{P}) = \mathcal{T}^{net}_{in}(\mathcal{Q})$, $\mathcal{T}^{net}_{out}(\mathcal{P}) = \mathcal{T}^{net}_{out}(\mathcal{Q})$, $\mathcal{T}^{io}_{in}(\mathcal{P}) = \mathcal{T}^{io}_{in}(\mathcal{Q})$, and $\mathcal{T}^{io}_{out}(\mathcal{P}) = \mathcal{T}^{io}_{out}(\mathcal{Q})$.

(ii) If $\mathcal{P}$ and $\mathcal{Q}$ are connectable, then they agree on the type (network or I/O) of their common external tapes, i.e., for all $c \in \mathcal{T}_{ext}(\mathcal{P}) \cap \mathcal{T}_{ext}(\mathcal{Q})$, we have that $c \in \mathcal{T}^{net}_{ext}(\mathcal{P}) \cap \mathcal{T}^{net}_{ext}(\mathcal{Q})$ or $c \in \mathcal{T}^{io}_{ext}(\mathcal{P}) \cap \mathcal{T}^{io}_{ext}(\mathcal{Q})$.

Definition 17. The systems $\mathcal{Q}$ and $\mathcal{P}$ are *I/O-connectable* if $\mathcal{Q}$ and $\mathcal{P}$ are connectable and $\mathcal{T}^{net}_{ext}(\mathcal{Q}) \cap \mathcal{T}^{net}_{ext}(\mathcal{P}) = \emptyset$. In other words, $\mathcal{Q}$ and $\mathcal{P}$ only connect through their I/O tapes.

The systems $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are *I/O-connectable* if $\mathcal{S}_i$ and $\mathcal{S}_j$ are I/O-connectable for every $i, j \leq n$ such that $i \neq j$.

As already explained in Section 4.1 for connectability, if $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are I/O-connectable, then every common external tape of the systems is the external input tape of exactly one system $\mathcal{S}_i$ and the external output tape of exactly one other system $\mathcal{S}_j$. So, there is no ambiguity in how these systems connect to each other in the parallel composition $\mathcal{S}_1 \mid \cdots \mid \mathcal{S}_n$. Note that for I/O-connectibility every common tape between these systems is an I/O tape, and hence, these systems only connect through their I/O tapes.

6.1.2 Adversarial Systems

We are now ready to define adversarial systems and how these connect to protocol systems.

Definition 18. An *adversarial system* $\mathcal{A}$ is a system such that no tape in $\mathcal{A}$ is named start or decision.

For a system $\mathcal{Q}$, we denote by $\mathrm{Adv}(\mathcal{Q})$ the set of all adversarial systems $\mathcal{A}$ such that $\mathcal{A}$ can be connected to $\mathcal{Q}$, $\mathcal{A} \mid \mathcal{Q}$ is environmentally bounded, the set of external tapes of $\mathcal{A}$ is disjoint from the set of I/O tapes of $\mathcal{Q}$, and $\mathcal{A} \mid \mathcal{Q}$ does not have any (external) network tapes (i.e., $\mathcal{A} \in \mathrm{Con}(\mathcal{Q})$, $\mathcal{T}_{ext}(\mathcal{A}) \cap \mathcal{T}^{io}_{ext}(\mathcal{Q}) = \emptyset$, and $\mathcal{T}^{net}_{ext}(\mathcal{A} \mid \mathcal{Q}) = \emptyset$). Thus, an adversary $\mathcal{A}$ can connect to the network tapes of $\mathcal{Q}$ only.

For two systems $\mathcal{Q}$ and $\mathcal{Q}'$, by $\mathrm{Sim}^{\mathcal{Q}'}(\mathcal{Q})$ we denote the set of all adversarial systems $\mathcal{A}$ such that $\mathcal{A}$ can be connected to $\mathcal{Q}$, $\mathcal{A} \mid \mathcal{Q}$ is environmentally bounded, the set of external tapes of $\mathcal{A}$ is disjoint from the set of I/O tapes of $\mathcal{Q}$, and $\mathcal{A} \mid \mathcal{Q}$ is compatible with $\mathcal{Q}'$.

We note that environmental systems, which will run concurrently with a protocol and possibly an adversarial system, may contain tapes named start or decision. In particular, they may contain a master IITM (while protocol and adversarial systems may not). This choice is justified and motivated by results shown in [16, 24].

6.2 Defining the Universal Composability Security Notions

The various security notions for universal composability proposed in the literature—(dummy) UC, black-box simulatability, strong simulatability, reactive simulatability—can be defined in a concise and simple way in the IITM model.

The basic idea behind these security notions is that security properties are specified as an ideal protocol/functionality $\mathcal{F}$. For example, $\mathcal{F}$ might specify a secure channel (see also Section 10.4 for examples). Using such an ideal protocol, parties can carry out their security critical tasks, such as communicating over a secure channel, in a secure way. This functionality is secure by construction, it defines what security means. Now a real protocol $\mathcal{P}$ realizes an ideal protocol $\mathcal{F}$ if for every (real) adversary $\mathcal{A}$ for $\mathcal{P}$ there exists an ideal adversary/simulator $\mathcal{I}$ for $\mathcal{F}$ such that no environment can distinguish whether it interacts with $\mathcal{A} \mid \mathcal{P}$ or with $\mathcal{I} \mid \mathcal{F}$. Since $\mathcal{F}$ is secure by definition and no environment can distinguish whether it interacts with the real or the ideal system, $\mathcal{P}$ is as secure as $\mathcal{F}$, and hence, secure.

The different security notions proposed in the literature differ in the kind of real adversaries considered and in the order of quantification. For dummy UC, the real adversary is just the dummy adversary, i.e., the adversary which forwards all messages between the protocol and the environment. For strong simulatability, the real adversary is dropped altogether and the environment directly connects to the network interface of the real protocol. For black-box simulatability, one requires that there exists a simulator $\mathcal{S}$ such that for every adversary $\mathcal{A}$ the composition $\mathcal{A} \,|\, \mathcal{S}$ is a good ideal adversary. For reactive simulatability, the ideal adversary may depend on the environment that tries to distinguish the real from the ideal protocol.

In order to formally define the notion of dummy UC, we first need to define the dummy adversary for a protocol. Given a system $\mathcal{P}$, by $\mathcal{D}_\mathcal{P}$ we denote the *dummy adversary for* $\mathcal{P}$ which is the dummy IITM $\mathcal{D}_\mathcal{P} = \mathcal{D}(\mathcal{T}^{net}_{in}(\mathcal{P}), \mathcal{T}^{net}_{out}(\mathcal{P}))$ (see Section 4.5) where all tapes $c' \in \mathcal{T}_{ext}(\mathcal{D}) \setminus \mathcal{T}^{net}_{ext}(\mathcal{P})$ are declared to be I/O tapes. The dummy adversary simply forwards all messages received from the protocol on a network tape to the environment on the corresponding I/O tape and all messages it receives from the environment on an I/O tape to the protocol on the corresponding network tape. We note that if $\mathcal{P}$ is environmentally bounded, then $\mathcal{D}_\mathcal{P} \,|\, \mathcal{P}$ is environmentally bounded too, and hence, $\mathcal{D}_\mathcal{P} \in \mathrm{Adv}(\mathcal{P})$.

The security notions defined in what follows are illustrated in Figures 4 to 7. In terms of runtime, the notions UC and dummyUC introduced below conceptually follow the definitions in [21].

Definition 19. Let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems, the *real* and *ideal protocol*, respectively, such that $\mathcal{P}$ is environmentally bounded.

1. *Strong Simulatability (SS):* $\mathcal{P} \leq^{SS} \mathcal{F}$ iff $\exists\, \mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}(\mathcal{F})$:
$$\mathcal{P} \cong \mathcal{S} \,|\, \mathcal{F} \ .$$
2. *Universal Simulatability/Composability (UC):* $\mathcal{P} \leq^{UC} \mathcal{F}$ iff $\forall\, \mathcal{A} \in \mathrm{Adv}(\mathcal{P})\ \exists\, \mathcal{I} \in \mathrm{Sim}^{\mathcal{A} \,|\, \mathcal{P}}(\mathcal{F})$:
$$\mathcal{A} \,|\, \mathcal{P} \cong \mathcal{I} \,|\, \mathcal{F} \ .$$
3. *Dummy Version of UC (dummyUC):* $\mathcal{P} \leq^{dumUC} \mathcal{F}$ iff $\exists\, \mathcal{I} \in \mathrm{Sim}^{\mathcal{D}_\mathcal{P} \,|\, \mathcal{P}}(\mathcal{F})$:
$$\mathcal{D}_\mathcal{P} \,|\, \mathcal{P} \cong \mathcal{I} \,|\, \mathcal{F} \ .$$
4. *Black-box Simulatability (BB):* $\mathcal{P} \leq^{BB} \mathcal{F}$ iff $\exists\, \mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}(\mathcal{F})\ \forall\, \mathcal{A} \in \mathrm{Adv}(\mathcal{P})$:
$$\mathcal{A} \,|\, \mathcal{P} \cong \mathcal{A} \,|\, \mathcal{S} \,|\, \mathcal{F} \ .$$
5. *Reactive Simulatability (RS):* $\mathcal{P} \leq^{RS} \mathcal{F}$ iff $\forall\, \mathcal{A} \in \mathrm{Adv}(\mathcal{P})\ \forall\, \mathcal{E} \in \mathrm{Env}(\mathcal{A} \,|\, \mathcal{P})\ \exists\, \mathcal{I} \in \mathrm{Sim}^{\mathcal{A} \,|\, \mathcal{P}}(\mathcal{F})$:
$$\mathcal{E} \,|\, \mathcal{A} \,|\, \mathcal{P} \equiv \mathcal{E} \,|\, \mathcal{I} \,|\, \mathcal{F} \ .$$

We say that $\mathcal{P}$ *SS-realizes* $\mathcal{F}$ if $\mathcal{P} \leq^{SS} \mathcal{F}$; similarly for the other security notions. We often simply say that $\mathcal{P}$ *realizes* $\mathcal{F}$ without explicitly indicating the underlying security notion. (This is justified by Theorem 7, which states that all security notions are equivalent.)

For strong and black-box simulatability, if $\mathcal{P}$ and $\mathcal{F}$ do not have disjoint network tapes, there does not exist a simulator with $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}(\mathcal{F})$. We therefore always (implicitly) assume that the network tapes of $\mathcal{F}$ are renamed first so that the set of network tapes of $\mathcal{P}$ and $\mathcal{F}$ *are* disjoint.

Remark 7. We emphasize that details such as addressing of machines by party/session IDs, corruption, and the structure of protocols are not, and do not need to be fixed in order to define the security notions. This makes the definitions and the underlying model particularly simple and expressive.

Remark 8. Recall that the notion of environmental indistinguishability that we use in Definition 19 is defined with respect to environments that output one-bit. According to Remark 3, using an alternative definition with a distinguisher and an environment that may output more than one bit would yield equivalent security notions.

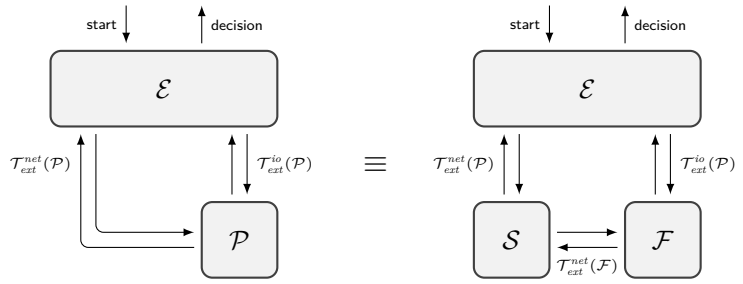

Figure 4: Strong simulatability (SS).

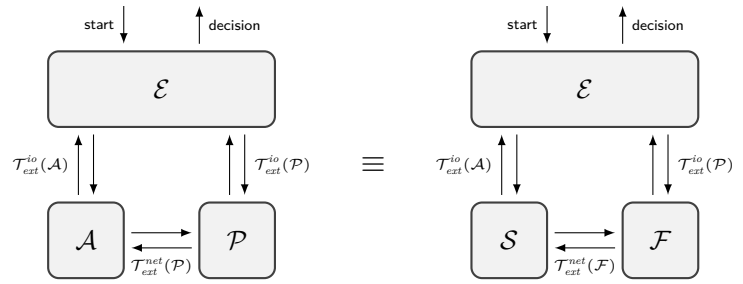

Figure 5: Universal simulatability/composability (UC) and reactive simulatability (RS).

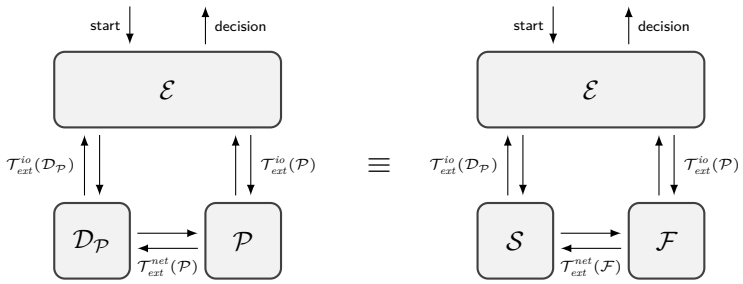

Figure 6: Dummy universal simulatability/composability (dummyUC).

Remark 9. Even though in the above definitions, $\mathcal{F}$ is not required to be environmentally bounded, in applications it will always be environmentally bounded.

Remark 10. Note that all systems considered in Definition 19, 1. to 4. are environmentally bounded and the ones in 5. are almost bounded. For all systems, except for $\mathcal{A} \,|\, \mathcal{S} \,|\, \mathcal{F}$ in the definition of black-box simulatability, this follows immediately from the definitions of the systems. For the system $\mathcal{A} \,|\, \mathcal{S} \,|\, \mathcal{F}$ this follows using Theorem 4 as follows: Since $\mathcal{D}_\mathcal{P} \in \mathrm{Adv}(\mathcal{P})$ (where $\mathcal{D}_\mathcal{P}$ is the dummy adversary defined above), we have that $\mathcal{D}_\mathcal{P} \,|\, \mathcal{P} \cong \mathcal{D}_\mathcal{P} \,|\, \mathcal{S} \,|\, \mathcal{F}$ and, hence, $\mathcal{P} \cong \mathcal{S} \,|\, \mathcal{F}$. By Theorem 4 ($\mathcal{P}$, $\mathcal{S} \,|\, \mathcal{F}$, and $\mathcal{A} \,|\, \mathcal{P}$ are environmentally bounded and $\mathcal{P} \cong \mathcal{S} \,|\, \mathcal{F}$), we obtain that $\mathcal{A} \,|\, \mathcal{S} \,|\, \mathcal{F}$ is environmentally bounded.

Remark 11. We note that all security notions imply that $\mathcal{P}$ and $\mathcal{F}$ have the same set of (external) I/O tapes (i.e., $\mathcal{T}^{io}_{ext}(\mathcal{P}) = \mathcal{T}^{io}_{ext}(\mathcal{F})$).

6.3 Relationships Between the Universal Composability Security Notions

In this section, we show that all security notions introduced in the previous section are equivalent. We note that for the equivalence with reactive simulatability we use that the environment gets external input, and

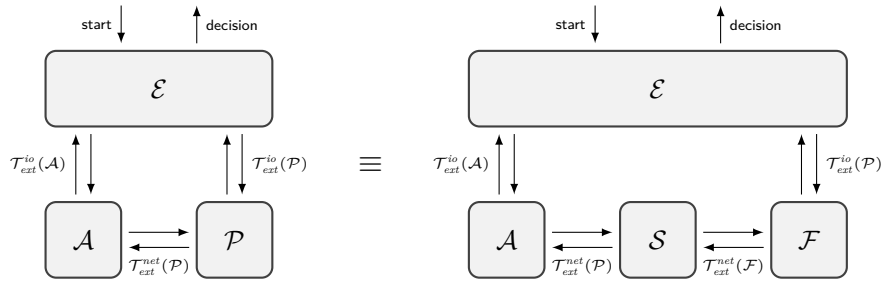

Figure 7: Black-box simulatability (BB).

hence, is non-uniform. As shown in [20], reactive simulatability is not equivalent to universal simulatability in the uniform case; this is also true if, in the case of reactive simulatability, the external input provided to the environment is chosen before the ideal adversary. All other equivalences hold true even for uniform environments, that is, environments that do not receive external input.

Theorem 7. *Let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems such that $\mathcal{P}$ is environmentally bounded. Then:*

$$\mathcal{P} \leq^{SS} \mathcal{F} \text{ iff } \mathcal{P} \leq^{UC} \mathcal{F} \text{ iff } \mathcal{P} \leq^{dumUC} \mathcal{F} \text{ iff } \mathcal{P} \leq^{BB} \mathcal{F} \text{ iff } \mathcal{P} \leq^{RS} \mathcal{F}.$$

Proof. Most of the above equivalences can be proven by equational reasoning using the equational principles established in Section 4.5 and Theorem 4. We first show that UC implies dummyUC implies SS implies UC. We then show that SS and BB are equivalent, which altogether proves equivalence of SS, UC, dummyUC, and BB. As mentioned above, these equivalences also hold true in the case where the environment does not get external input, i.e., is uniform. We then prove that SS is equivalent to RS, which concludes the proof of Theorem 7. As mentioned above, for this equivalence, we (have to) use that the environment gets external input, i.e., is non-uniform.

UC ⇒ dummyUC: It is easy to see that $\mathcal{P} \leq^{UC} \mathcal{F}$ implies $\mathcal{P} \leq^{dumUC} \mathcal{F}$: Since $\mathcal{P}$ is environmentally bounded, the system $\mathcal{D}_{\mathcal{P}} \,|\, \mathcal{P}$ is environmentally bounded as well. Hence, $\mathcal{D}_{\mathcal{P}} \in \mathrm{Adv}(\mathcal{P})$ and, by the definition of $\mathcal{P} \leq^{UC} \mathcal{F}$, there exists $\mathcal{I} \in \mathrm{Sim}^{\mathcal{D}_{\mathcal{P}} \,|\, \mathcal{P}}(\mathcal{F})$ such that $\mathcal{D}_{\mathcal{P}} \,|\, \mathcal{P} \cong \mathcal{I} \,|\, \mathcal{F}$. We conclude that $\mathcal{P} \leq^{dumUC} \mathcal{F}$.

dummyUC ⇒ SS: Intuitively, if $\mathcal{P} \leq^{dumUC} \mathcal{F}$, then we can use the simulator obtained for the dummy adversary to prove $\mathcal{P} \leq^{SS} \mathcal{F}$. More specifically:

By the definition of $\mathcal{P} \leq^{dumUC} \mathcal{F}$, there exists $\mathcal{I} \in \mathrm{Sim}^{\mathcal{D}_{\mathcal{P}} \,|\, \mathcal{P}}(\mathcal{F})$ such that $\mathcal{D}_{\mathcal{P}} \,|\, \mathcal{P} \cong \mathcal{I} \,|\, \mathcal{F}$. We define $\mathcal{S} := \mathcal{I}'$ where $\mathcal{I}'$ is obtained from $\mathcal{I}$ by renaming the (external) I/O tapes c' of $\mathcal{I}$ (i.e., the tapes that do not connect to $\mathcal{F}$ but to an environment) to c and declaring them to be network tapes. Hence, $\mathcal{S} \,|\, \mathcal{F}$ is compatible with $\mathcal{P}$ and $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}(\mathcal{F})$. Let $\mathcal{E} \in \mathrm{Env}(\mathcal{P})$. Let $\mathcal{E}'$ be the system obtained from $\mathcal{E}$ by renaming the (external) network tapes c of $\mathcal{E}$ (i.e., the tapes connecting to the network tapes of $\mathcal{P}$) to c' and declaring them to be I/O tapes. Hence, $\mathcal{E}' \in \mathrm{Env}(\mathcal{D}_{\mathcal{P}} \,|\, \mathcal{P})$. Then, we obtain that:

$$\begin{aligned} \mathcal{E} \,|\, \mathcal{P} &\equiv \mathcal{E}' \,|\, \mathcal{D}_{\mathcal{P}} \,|\, \mathcal{P} && \text{(Lemma 5 and 4)} \\ &\equiv \mathcal{E}' \,|\, \mathcal{I} \,|\, \mathcal{F} && (\mathcal{D}_{\mathcal{P}} \,|\, \mathcal{P} \cong \mathcal{I} \,|\, \mathcal{F}) \\ &\equiv \mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F} && \text{(Lemma 4)} \end{aligned}$$

We conclude that $\mathcal{P} \leq^{SS} \mathcal{F}$.

SS ⇒ UC: The basic idea of proving this implication is as follows: If $\mathcal{P} \leq^{SS} \mathcal{F}$, then there exists a simulator $\mathcal{S}$ such that $\mathcal{P} \cong \mathcal{S} \,|\, \mathcal{F}$. Now for every adversary $\mathcal{A}$ for $\mathcal{P}$, we define $\mathcal{I} := \mathcal{A} \,|\, \mathcal{S}$ to be the ideal adversary (simulator). We obtain that $\mathcal{A} \,|\, \mathcal{P} \cong \mathcal{I} \,|\, \mathcal{F}$. More formally: By definition of $\mathcal{P} \leq^{SS} \mathcal{F}$, there exists $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}(\mathcal{F})$ such that $\mathcal{P} \cong \mathcal{S} \,|\, \mathcal{F}$. Let $\mathcal{A} \in \mathrm{Adv}(\mathcal{P})$. By Lemma 6, we may assume that both $\mathcal{S}$ and $\mathcal{A}$ are

single IITMs that accept every message in mode CheckAddress.[20] We define $\mathcal{I} := \mathcal{A} \,|\, \mathcal{S}$. Since $\mathcal{P} \cong \mathcal{S} \,|\, \mathcal{F}$ and $\mathcal{P}$, $\mathcal{S} \,|\, \mathcal{F}$, and $\mathcal{A} \,|\, \mathcal{P}$ are environmentally bounded, by Theorem 4 (note that $\mathcal{A}$ and $\mathcal{S} \,|\, \mathcal{F}$ are protocol systems because $\mathcal{A}$ and $\mathcal{S}$ accept every message in mode CheckAddress) we obtain that $\mathcal{A} \,|\, \mathcal{P} \cong \mathcal{A} \,|\, \mathcal{S} \,|\, \mathcal{F} = \mathcal{I} \,|\, \mathcal{F}$ and $\mathcal{A} \,|\, \mathcal{S} \,|\, \mathcal{F} = \mathcal{I} \,|\, \mathcal{F}$ is environmentally bounded. Hence, we can also conclude that $\mathcal{I} \in \mathrm{Sim}^{\mathcal{A} \,|\, \mathcal{P}}(\mathcal{F})$, which, altogether, proves that $\mathcal{P} \leq^{UC} \mathcal{F}$.

SS $\Leftrightarrow$ BB: It follows directly from Theorem 4 that $\mathcal{P} \leq^{SS} \mathcal{F}$ implies $\mathcal{P} \leq^{BB} \mathcal{F}$: By definition of $\mathcal{P} \leq^{SS} \mathcal{F}$, there exists $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}(\mathcal{F})$ such that $\mathcal{P} \cong \mathcal{S} \,|\, \mathcal{F}$. Let $\mathcal{A} \in \mathrm{Adv}(\mathcal{P})$. By Lemma 6, we may assume that both $\mathcal{S}$ and $\mathcal{A}$ are single IITMs that accept every message in mode CheckAddress.[21] Since $\mathcal{A} \,|\, \mathcal{P}$, $\mathcal{P}$, and $\mathcal{S} \,|\, \mathcal{F}$ are environmentally bounded and $\mathcal{P} \cong \mathcal{S} \,|\, \mathcal{F}$, by Theorem 4 (note that $\mathcal{A}$ and $\mathcal{S} \,|\, \mathcal{F}$ are protocol systems because $\mathcal{A}$ and $\mathcal{S}$ accept every message in mode CheckAddress), we have that $\mathcal{A} \,|\, \mathcal{P} \cong \mathcal{A} \,|\, \mathcal{S} \,|\, \mathcal{F}$. We conclude that $\mathcal{P} \leq^{BB} \mathcal{F}$.

Next, we show that $\mathcal{P} \leq^{BB} \mathcal{F}$ implies $\mathcal{P} \leq^{SS} \mathcal{F}$: By definition of $\mathcal{P} \leq^{BB} \mathcal{F}$ there exists $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}(\mathcal{F})$ such that $\mathcal{A} \,|\, \mathcal{P} \cong \mathcal{A} \,|\, \mathcal{S} \,|\, \mathcal{F}$ for all $\mathcal{A} \in \mathrm{Adv}(\mathcal{P})$. Since the dummy adversary is a valid adversary for $\mathcal{P}$, i.e., $\mathcal{D}_{\mathcal{P}} \in \mathrm{Adv}(\mathcal{P})$, we immediately obtain that $\mathcal{D}_{\mathcal{P}} \,|\, \mathcal{P} \cong \mathcal{D}_{\mathcal{P}} \,|\, \mathcal{S} \,|\, \mathcal{F}$. By Lemma 5, we conclude $\mathcal{P} \cong \mathcal{S} \,|\, \mathcal{F}$, and hence, $\mathcal{P} \leq^{SS} \mathcal{F}$.

SS $\Leftrightarrow$ RS: It is trivially seen that $\mathcal{P} \leq^{UC} \mathcal{F}$ implies $\mathcal{P} \leq^{RS} \mathcal{F}$. Since, as shown above, UC and SS are equivalent, it follows that $\mathcal{P} \leq^{SS} \mathcal{F}$ implies $\mathcal{P} \leq^{RS} \mathcal{F}$. We now prove that $\mathcal{P} \leq^{RS} \mathcal{F}$ implies $\mathcal{P} \leq^{SS} \mathcal{F}$. The main argument is similar to the one presented in [4]. We therefore only present the proof sketch.

By definition of $\mathcal{P} \leq^{RS} \mathcal{F}$, for all $\mathcal{A} \in \mathrm{Adv}(\mathcal{P})$ and all $\mathcal{E} \in \mathrm{Env}(\mathcal{A} \,|\, \mathcal{P})$ there exists $\mathcal{I} \in \mathrm{Sim}^{\mathcal{A} \,|\, \mathcal{P}}(\mathcal{F})$ such that:

$$\mathcal{E} \,|\, \mathcal{A} \,|\, \mathcal{P} \equiv \mathcal{E} \,|\, \mathcal{I} \,|\, \mathcal{F} \ . \tag{14}$$

We choose $\mathcal{A} = \mathcal{D}_{\mathcal{P}}$ to be the dummy adversary for $\mathcal{P}$. We also choose $\mathcal{E} \in \mathrm{Env}(\mathcal{D}_{\mathcal{P}} \,|\, \mathcal{P})$ to be a "universal" Turing machine (more precisely, a universal IITM) which takes as external input (i.e., input on start) a tuple of the form $(a, e, 1^t)$ where e is an encoding of some IITM (representing an environmental system $\mathcal{E}'$), a is interpreted as an external input to $\mathcal{E}'$, and t is interpreted as runtime. (By Lemma 7, we may assume that e encodes a single IITM which accepts every message in mode CheckAddress.) The universal IITM $\mathcal{E}$ simulates $\mathcal{E}'$ with external input a up to t steps. Clearly, $\mathcal{E}$ is universally bounded because its runtime is polynomial in the security parameter plus the length of the external input. Now, given a security parameter η, external input a, and an environmental system $\mathcal{E}'$, there exists a tuple $(a, e, 1^t)$ of length polynomial in $\eta + |a|$ (it suffices to choose t polynomial in $\eta + |a|$ because $\mathcal{E}'$ is universally bounded) such that $\mathcal{E}$ with external input $(a, e, 1^t)$ precisely simulates $\mathcal{E}'$. Hence, (14) implies $\mathcal{E}' \,|\, \mathcal{D}_{\mathcal{P}} \,|\, \mathcal{P} \equiv \mathcal{E}' \,|\, \mathcal{I} \,|\, \mathcal{F}$, i.e., $\mathcal{P} \leq^{dumUC} \mathcal{F}$. Since we already know that $\mathcal{P} \leq^{dumUC} \mathcal{F}$ implies $\mathcal{P} \leq^{SS} \mathcal{F}$, we obtain $\mathcal{P} \leq^{SS} \mathcal{F}$.

□

Since all security notions are equivalent, it does not matter which notion we use in the following. Since strong simulatability ($\leq^{SS}$) is the conceptually simplest notion, we typically use this notion.

Remark 12. As already mentioned at the beginning of this section, while intuitively one would expect that all security notions are equivalent, except for reactive simulatability (with uniform environments), equivalence does not hold true in all models, in particular this is the case for Canetti's UC model (version from 2005). In the UC model, due to what is sometimes called the *incompleteness of the dummy adversary* UC and dummyUC are not equivalent (see, e.g., [21]). However, completeness is needed in the proof of the composition theorem. As mentioned in [4, page 47], dummyUC and SS are not equivalent in the UC model either.

[20] We note that this assumption is actually not required. Below, where we use it, we could have used the mentioned stronger variant of Theorem 4.

[21] We note that this assumption is actually not required. We could have used the mentioned stronger variant of Theorem 4 instead.

6.4 Reflexivity and Transitivity

The following two lemmas state that strong simulatability is a reflexive and transitive relationship. By Theorem 7, this is true for all security notions.

While again these properties are expected, they are not satisfied in all models. One trivial reason is that the real and ideal protocol are required to be syntactical different in some models: The ideal protocol often has to be a single machine, while this is not the case for the real protocol. More importantly, inadequate definitions of runtime notions can also cause problems (see, for instance, the discussion in [23]).

Lemma 12. *Let $\mathcal{P}$ be an environmentally bounded protocol system. Then, $\mathcal{P} \leq^{SS} \mathcal{P}$.*

Proof. Reflexivity holds for $\leq^{UC}$ because every valid adversary $\mathcal{A} \in \mathrm{Adv}(\mathcal{P})$ is a valid simulator $\mathcal{A} \in \mathrm{Sim}^{\mathcal{A}\,|\,\mathcal{P}}(\mathcal{P})$ and $\mathcal{A}\,|\,\mathcal{P} \cong \mathcal{A}\,|\,\mathcal{P}$. By Theorem 7, reflexivity also holds for $\leq^{SS}$. □

Lemma 13. *Let $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{F}$ be protocol systems such that $\mathcal{P}$ and $\mathcal{Q}$ are environmentally bounded, $\mathcal{P} \leq^{SS} \mathcal{Q}$, and $\mathcal{Q} \leq^{SS} \mathcal{F}$. Then, $\mathcal{P} \leq^{SS} \mathcal{F}$.*

Proof. We will show transitivity for $\leq^{UC}$ and obtain transitivity for $\leq^{SS}$ by Theorem 7. By Theorem 7, we obtain that $\mathcal{P} \leq^{UC} \mathcal{Q}$ and $\mathcal{Q} \leq^{UC} \mathcal{F}$. Let $\mathcal{A} \in \mathrm{Adv}(\mathcal{P})$. By definition of $\mathcal{P} \leq^{UC} \mathcal{Q}$, there exists $\mathcal{I}_1 \in \mathrm{Sim}^{\mathcal{A}\,|\,\mathcal{P}}(\mathcal{Q})$ such that $\mathcal{A}\,|\,\mathcal{P} \cong \mathcal{I}_1\,|\,\mathcal{Q}$. Since $\mathcal{I}_1 \in \mathrm{Adv}(\mathcal{Q})$, by definition of $\mathcal{Q} \leq^{UC} \mathcal{F}$, there exists $\mathcal{I}_2 \in \mathrm{Sim}^{\mathcal{I}_1\,|\,\mathcal{Q}}(\mathcal{F})$ such that $\mathcal{I}_1\,|\,\mathcal{Q} \cong \mathcal{I}_2\,|\,\mathcal{F}$. Clearly, $\mathcal{I}_2 \in \mathrm{Sim}^{\mathcal{A}\,|\,\mathcal{P}}(\mathcal{F})$. By transitivity of $\cong$ (see Lemma 2), we have that $\mathcal{A}\,|\,\mathcal{P} \cong \mathcal{I}_2\,|\,\mathcal{F}$. We conclude that $\mathcal{P} \leq^{UC} \mathcal{F}$. Finally, by Theorem 7, we conclude that $\mathcal{P} \leq^{SS} \mathcal{F}$. □

7 Composition Theorems for the Realization Relations

We now prove general composition theorems for the realization relations. These theorems are at the heart of the universal composability paradigm and are the main motivation for this paradigm.

In a nutshell, these theorems say that if real protocols are secure individually, then their concurrent composition is secure as well. More precisely, if real protocols each realize some ideal protocol, then the concurrent composition of the real protocols realizes the concurrent composition of the corresponding ideal protocols, even if an unbounded number of sessions of the real/ideal protocols run concurrently. Therefore, these theorems can be used to analyze and design systems in a modular way: it suffices to show security of every individual real protocol in a single session, i.e., show that a single session of a real protocol realizes a single session of the corresponding ideal protocol, in order to conclude security of the concurrent composition of multiple sessions of these protocols.

The composition theorems presented in this section follow quite easily from the composition theorems for environmental indistinguishability presented in Section 5. Analogously to Section 5, we first, in Section 7.1, present a composition theorem for the composition of a constant number of, possibly different, protocols (more precisely, protocol systems). We then state a theorem which captures the security of a protocol when run in an unbounded number of sessions/copies (unbounded self-composition). We, similarly to Section 5, prove two versions of this theorem: one in which protocol participants are not aware of the session identifiers that are used to address the different protocol sessions (see Section 7.2) and one in which they are aware of the session identifiers (see Section 7.3). The theorems for a constant number of protocols and for unbounded self-composition can freely be combined to establish, in a modular way, the security of more and more complex systems, as illustrated in Section 7.4.

All composition theorems are stated for strong simulatability, but by Theorem 7 they also hold for every other security notion introduced in Section 6.2, e.g., universal simulatability (UC).

Our composition theorems are, in many ways, more expressive than those proven in other models (see also the discussion in Sections 10 and 11). The fact that the IITM model does not a priori fix a specific addressing mechanism or a specific form of corruption and does not impose a specific structure on protocols allows us to prove general composition theorems which hold true no matter how these details are chosen. Moreover, unlike other models, our composition theorems do not restrict the environment to only access top-level protocols. This flexibility and generality of the theorems is also reflected in the fact that the general

joint state theorem is an immediate consequence of our composition theorem. In other models, even stating the joint state theorem requires to change the model and/or introduce new concepts. Similarly, composition theorems with global setup can also be formalized without changing the model and the main such theorems again follow immediately from the general composition theorem, as discussed in Section 10.3.

7.1 Composition Theorem for a Constant Number of Protocol Systems

We now present the composition theorem for the composition of a constant number of (possibly different) protocol systems. The theorem directly follows from Theorem 4.

Theorem 8. *Let $k \geq 1$ and $\mathcal{Q}, \mathcal{P}_1, \ldots, \mathcal{P}_k, \mathcal{F}_1, \ldots, \mathcal{F}_k$ be protocol systems with pairwise disjoint sets of network tapes such that the following conditions are satisfied:*

1. *For all $j \leq k$: $\mathcal{P}_j$ is environmentally bounded and $\mathcal{P}_j \leq^{SS} \mathcal{F}_j$*
2. *$\mathcal{Q}, \mathcal{P}_1, \ldots, \mathcal{P}_k$ are I/O-connectable and $\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \cdots \,|\, \mathcal{P}_k$ is environmentally bounded.*

Then, $\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \cdots \,|\, \mathcal{P}_k \leq^{SS} \mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \cdots \,|\, \mathcal{F}_k$.

Clearly, the theorem also hold true if the system $\mathcal{Q}$ is dropped. We note that in the above theorem it is not required that, for all $j \leq k$, $\mathcal{F}_j$ and $\mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \cdots \,|\, \mathcal{F}_k$ are environmentally bounded (although in applications this will typically be the case). Just as for Theorem 4, we finally remark that Condition 2. in the above theorem is easy to check. I/O-connectability is a simple syntactic condition. Moreover, for typical applications, it is easy to check whether a system is environmentally bounded (see also the discussion in Section 8).

Proof of Theorem 8. Since $\mathcal{P}_j \leq^{SS} \mathcal{F}_j$, for all $j \leq k$, there exists $\mathcal{S}_j \in \mathrm{Sim}^{\mathcal{P}_j}(\mathcal{F}_j)$ such that $\mathcal{P}_j \cong \mathcal{S}_j \,|\, \mathcal{F}_j$. By Lemma 6, we may assume that all $\mathcal{S}_j$ are single IITMs that accept all messages in mode CheckAddress.[22] By definition of $\mathrm{Sim}^{\mathcal{P}_j}(\mathcal{F}_j)$, we know that $\mathcal{S}_j \,|\, \mathcal{F}_j$ is environmentally bounded and $\mathcal{P}_j$ and $\mathcal{S}_j \,|\, \mathcal{F}_j$ are compatible. Since $\mathcal{Q}, \mathcal{P}_1, \ldots, \mathcal{P}_k$ are I/O-connectable, it follows that $\mathcal{S}_1 \,|\, \mathcal{F}_1, \ldots, \mathcal{S}_k \,|\, \mathcal{F}_k$ are connectable. With this, by Theorem 4 (with $\mathcal{S} = \mathcal{Q}$ and $\mathcal{Q}_j = \mathcal{S}_j \,|\, \mathcal{F}_j$ for all $j \leq k$; note that $\mathcal{Q}_j$ is a protocol system because $\mathcal{S}_j$ accepts every message in mode CheckAddress), we obtain that $\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \cdots \,|\, \mathcal{P}_k \cong \mathcal{Q} \,|\, \mathcal{S}_1 \,|\, \mathcal{F}_1 \,|\, \cdots \,|\, \mathcal{S}_k \,|\, \mathcal{F}_k$ and that $\mathcal{Q} \,|\, \mathcal{S}_1 \,|\, \mathcal{F}_1 \,|\, \cdots \,|\, \mathcal{S}_k \,|\, \mathcal{F}_k$ is environmentally bounded. In particular, we obtain that $\mathcal{S}_1 \,|\, \cdots \,|\, \mathcal{S}_k \in \mathrm{Sim}^{\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \cdots \,|\, \mathcal{P}_k}(\mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \cdots \,|\, \mathcal{F}_k)$. Altogether, this proves that $\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \cdots \,|\, \mathcal{P}_k \leq^{SS} \mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \cdots \,|\, \mathcal{F}_k$. □

7.2 Composition Theorem for Unbounded Self-Composition

We now present a composition theorem for the unbounded self-composition of a protocol system. It basically says that if a real protocol securely realizes an ideal protocol, then an unbounded number of sessions (copies) of the real protocol securely realize an unbounded number of sessions of the ideal protocol, where both for the real and the ideal protocol we consider the session versions of these protocols to address different sessions. In particular, SIDs are merely used as a means to address certain (IITMs belonging to) sessions of protocols. Protocol participants are not and do not need to be aware of the SID used to address their protocol sessions, and the specific addressing mechanism used. The theorem follows from Theorem 5.

As mentioned before, below we will present another composition theorem, which is a corollary of Theorem 6, where protocol participants are aware of their own SID.

Theorem 9. *Let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems with disjoint sets of network tapes such that $!\underline{\mathcal{P}}$ is environmentally bounded and $\mathcal{P} \leq^{SS} \mathcal{F}$. Then, $!\underline{\mathcal{P}} \leq^{SS} !\underline{\mathcal{F}}$.*

We mention, similarly to the remark following Theorem 5, that $\mathcal{P}$ will typically be environmentally *strictly* bounded and that this is easy to check. By Lemma 17 it then follows immediately that $!\underline{\mathcal{P}}$ is environmentally strictly bounded as well, and hence, environmentally bounded. So Theorem 9 can be applied directly to typical protocol systems without additional effort.

[22] We note that this assumption is not actually required. We could use the mentioned stronger variant of Theorem 4 when we use this theorem in the proof.

Proof of Theorem 9. Since $\mathcal{P} \leq^{SS} \mathcal{F}$, there exists $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}(\mathcal{F})$ such that $\mathcal{P} \cong \mathcal{S} \,|\, \mathcal{F}$. By Lemma 6, we may assume that $\mathcal{S}$ is a single IITM which, in mode CheckAddress, accepts all messages. Hence, $\mathcal{S} \,|\, \mathcal{F}$ is a protocol system (in particular, Condition (ii) of Definition 11 is satisfied). By definition of $\mathrm{Sim}^{\mathcal{P}}(\mathcal{F})$, $\mathcal{S} \,|\, \mathcal{F}$ is environmentally bounded and $\mathcal{P}$ and $\mathcal{S} \,|\, \mathcal{F}$ are compatible. With this, by Theorem 5 (with $\mathcal{Q} = \mathcal{S} \,|\, \mathcal{F}$), we obtain that $!\underline{\mathcal{P}} \cong !\underline{\mathcal{S}} \,|\, !\underline{\mathcal{F}}$ and $!\underline{\mathcal{S}} \,|\, !\underline{\mathcal{F}}$ is environmentally bounded. The latter implies that $!\underline{\mathcal{S}} \in \mathrm{Sim}^{!\underline{\mathcal{P}}}(!\underline{\mathcal{F}})$. This proves $!\underline{\mathcal{P}} \leq^{SS} !\underline{\mathcal{F}}$. □

7.3 Composition Theorem for Unbounded Self-Composition of SID Dependent Protocols

As explained before, in Theorem 9, similarly to Theorem 5, SIDs are merely used as a means to address certain (IITMs belonging to) sessions of protocols. In particular, protocol participants are not aware of the SIDs used to address their protocol sessions, and the specific addressing mechanism that is employed. We now present a composition theorem for the unbounded self-composition of a protocol where the protocol participants *are* aware of their SIDs. This theorem is a corollary of Theorem 6.

Before we can state the composition theorem, we need to introduce the notion of single-session realizability, using the notion of σ-environmental indistinguishability ($\cong_{\sigma\text{-single}}$) introduced in Section 5.3.1: For two systems $\mathcal{P}$ and $\mathcal{F}$ and an SID function σ, we denote by $\mathrm{Sim}^{\mathcal{P}}_{\sigma\text{-single}}(\mathcal{F})$ the set of all adversarial systems $\mathcal{S}$ such that $\mathcal{S}$ can be connected to $\mathcal{F}$, the set of external tapes of $\mathcal{S}$ is disjoint from the set of I/O tapes of $\mathcal{F}$ (i.e., $\mathcal{T}_{ext}(\mathcal{S}) \cap \mathcal{T}^{io}_{ext}(\mathcal{F}) = \emptyset$), $\mathcal{S} \,|\, \mathcal{F}$ and $\mathcal{P}$ are compatible, and $\mathcal{S} \,|\, \mathcal{F}$ is σ-environmentally bounded. We note that $\mathrm{Sim}^{\mathcal{P}}(\mathcal{F}) \subseteq \mathrm{Sim}^{\mathcal{P}}_{\sigma\text{-single}}(\mathcal{F})$; the only difference between these two sets is that $\mathcal{S} \,|\, \mathcal{F}$ has to be environmentally bounded in one case and σ-environmentally bounded in the other case.

Definition 20. Let σ be an SID function and let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems (the real and ideal protocol, respectively) such that $\mathcal{P}$ is σ-environmentally bounded. We say that $\mathcal{P}$ *single-session SS-realizes* $\mathcal{F}$ *w.r.t.* σ or $\mathcal{P}$ *SS-realizes* $\mathcal{F}$ *w.r.t.* σ*-single session environments*, denoted by $\mathcal{P} \leq^{SS}_{\sigma\text{-single}} \mathcal{F}$, iff there exists $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{\sigma\text{-single}}(\mathcal{F})$ such that $\mathcal{P} \cong_{\sigma\text{-single}} \mathcal{S} \,|\, \mathcal{F}$.[23]

Now, we are able to formulate the composition theorem for unbounded self-composition of protocol systems that may depend on their SID:

Theorem 10. *Let σ be an SID function and let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems such that $\mathcal{P}$ and $\mathcal{F}$ are σ-session versions with disjoint sets of network tapes, $!\underline{\mathcal{P}}$ is environmentally bounded, and $\mathcal{P} \leq^{SS}_{\sigma\text{-single}} \mathcal{F}$. Then, $\mathcal{P} \leq^{SS} \mathcal{F}$.*

We note that a stronger variant of the above theorem, where it is assumed that only $\mathcal{P}$ (instead of $!\underline{\mathcal{P}}$) is environmentally bounded, also holds. Such a theorem can be obtained from the stronger variant of Theorem 6 which is sketched following Theorem 6 in Section 5.3.2. However, as discussed above, typically $\mathcal{P}$ is environmentally strictly bounded and in this case, by Lemma 17, it follows immediately that $!\underline{\mathcal{P}}$ is environmentally bounded. So in concrete applications, Theorem 10 is just as easy to apply as the stronger variant.

Before we prove the above theorem, we show the following lemma, which basically states that if there exists a good single-session simulator w.r.t. some SID function σ, then there also exists a good single-session simulator which is a σ-session version. To state the lemma, we use the following terminology. We say that an IITM M is *σ-complete* if it satisfies all conditions stated in Definition 13 (σ-session version) but with Condition 2. replaced by the following stronger condition: If the first input message that M accepted in ρ in mode CheckAddress is m_0 on tape c_0 and (later) M is activated in mode CheckAddress in ρ with an input message m on tape c, then M accepts m iff $\sigma(m, c) = \sigma(m_0, c_0)$. In other words, σ determines exactly those messages accepted by M in mode CheckAddress.

Lemma 14. *Let σ be an SID function and let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems such that $\mathcal{P}$ and $\mathcal{F}$ are σ-session versions, $\mathcal{P}$ and $\mathcal{F}$ have disjoint sets of network tapes, $\mathcal{P}$ is σ-environmentally bounded, and $\mathcal{P} \leq^{SS}_{\sigma\text{-single}} \mathcal{F}$. Then, there exists $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{\sigma\text{-single}}(\mathcal{F})$ such that $\mathcal{P} \cong_{\sigma\text{-single}} \mathcal{S} \,|\, \mathcal{F}$ and $\mathcal{S}$ is a single IITM which is σ-complete.*

[23] As usual, if the sets of network tapes of $\mathcal{P}$ and $\mathcal{F}$ are not disjoint, we first rename the network tapes of $\mathcal{F}$ accordingly.

Proof. Since $\mathcal{P} \leq^{SS}_{\sigma\text{-single}} \mathcal{F}$, there exists $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{\sigma\text{-single}}(\mathcal{F})$ such that:

$$\forall \mathcal{E} \in \mathrm{Env}_{\sigma\text{-single}}(\mathcal{P}) \colon \mathcal{E} \,|\, \mathcal{P} \equiv \mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F} \ . \tag{15}$$

In what follows, we construct a simulator $\mathcal{S}'$ which is a single IITM that basically simulates $\mathcal{S}$ and is σ-complete.

By Lemma 6, we may assume that $\mathcal{S} = M$ is a single IITM which, in mode CheckAddress, accepts all messages. We now define the system $\mathcal{S}' := M'$ where M' is σ-complete. The basic idea behind the construction of M' is as follows: Let us consider a run of $\mathcal{E} \,|\, M \,|\, \mathcal{F}$ for some $\mathcal{E} \in \mathrm{Env}_{\sigma\text{-single}}(\mathcal{P})$. Since $\mathcal{E}$ is σ-single session, there exists an SID sid such that every message m output by $\mathcal{E}$ in this run has SID sid, i.e., for every such message m and tape c we have that $\sigma(m, c) = sid$. Nevertheless, M might output messages with $\sigma(m, c) \neq sid$.

If M sent such a message to $\mathcal{E}$, then $\mathcal{E}$ could easily distinguish $\mathcal{P}$ from $M \,|\, \mathcal{F}$: Because $\mathcal{E}$ is σ-single session and $\mathcal{P}$ is an σ-session version, $\mathcal{P}$ will only output messages to $\mathcal{E}$ with SID sid. So, by (15), it can only happen with negligible probability that M sends a message m on tape c to $\mathcal{E}$ with $\sigma(m, c) \neq sid$.

However, M might send a message m on tape c to $\mathcal{F}$ with $\sigma(m, c) \neq sid$ (without $\mathcal{E}$ noticing this). Since $\mathcal{F}$ is a σ-session version, m will not be accepted by (a copy of) a machine in $\mathcal{F}$ which has accepted messages with SID sid. Also, whenever (a copy of) a machine of $\mathcal{F}$ receives a message, it outputs a message with the same SID. So, the copy of the machine of $\mathcal{F}$ which accepts m must not (except with negligible probability) produce output to $\mathcal{E}$. In addition, with $\mathcal{E}$ being σ-single session, $\mathcal{E}$ cannot send messages to such a machine. Consequently, since such a machine cannot interact with $\mathcal{E}$, M' can simply simulate this machine internally.

From the above discussion the definition of M' suggests itself: M' is compatible with M and its CheckAddress mode is defined in such a way that M' is σ-complete, i.e., M' rejects all messages m on any tape c with $\sigma(m, c) = \bot$ and it accepts the first message m_0 on some tape c_0 with $\sigma(m_0, c_0) \neq \bot$. From then on, M' accepts a message m on tape c iff $\sigma(m, c) = \sigma(m_0, c_0)$. In mode Compute, M' internally simulates a copy of M and possibly machines of $\mathcal{F}$ as follows.

- Whenever M' receives input m on some of its external tapes, M' internally simulates M with input m (on the same tape).
- Whenever (the internally simulated) M produces empty output, M' produces empty output.
- Whenever M outputs m on tape c with $\sigma(m, c) = sid$, M' outputs m (on the same tape).
- Whenever M outputs m on tape c to $\mathcal{E}$ with $\sigma(m, c) \neq sid$, M' produces empty output; as discussed above, this will happen with only negligible probability.
- Whenever M outputs m on tape c to (a copy of) a machine of $\mathcal{F}$ with $\sigma(m, c) \neq sid$, then M' internally simulates this copy with input m.
- Whenever an internally simulated machine of $\mathcal{F}$ produces empty output, M' produces empty output.
- Whenever an internally simulated machine of $\mathcal{F}$ produces output on its I/O interface to $\mathcal{E}$, M' produces empty output; as discussed above, this will happen with only negligible probability.
- Whenever an internally simulated machine of $\mathcal{F}$ produces output m on an internal tape to another machine in $\mathcal{F}$, M' internally simulates the receiving machine with input m.
- Whenever an internally simulated machine of $\mathcal{F}$ outputs m on its network interface (to M'), M' internally simulates M with input m.

From the above discussion and by (15), it easily follows that

$$\forall \mathcal{E} \in \mathrm{Env}_{\sigma\text{-single}}(\mathcal{P}) \colon \mathcal{E} \,|\, \mathcal{P} \equiv \mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F} \equiv \mathcal{E} \,|\, \mathcal{S}' \,|\, \mathcal{F} \ . \tag{16}$$

Now, since $\mathcal{S} \,|\, \mathcal{F}$ is σ-environmentally bounded, it is easy to see that $\mathcal{S}' \,|\, \mathcal{F}$ is σ-environmentally bounded too. With $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{\sigma\text{-single}}(\mathcal{F})$ we thus obtain $\mathcal{S}' \in \mathrm{Sim}^{\mathcal{P}}_{\sigma\text{-single}}(\mathcal{F})$. Also, by construction we have that $\mathcal{S}'$ is σ-complete. □

Theorem 10 now follows directly from the above lemma and Theorem 6:

Proof of Theorem 10. By Lemma 14, there exists $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{\sigma\text{-single}}(\mathcal{F})$ such that $\mathcal{S}$ is a σ-complete IITM and $\mathcal{P} \cong_{\sigma\text{-single}} \mathcal{S} \,|\, \mathcal{F}$. Since $\mathcal{S}$ is σ-complete, we can conclude that $\mathcal{P} \cong_{\sigma\text{-single}} !\mathcal{S} \,|\, \mathcal{F}$: since $\mathcal{E} \in \mathrm{Env}_{\sigma\text{-single}}(\mathcal{P})$ invokes only a single session, i.e., sends only messages to $\mathcal{S}$ and $\mathcal{F}$ with the same SID (w.r.t. σ) and $\mathcal{F}$ is a σ-session version, $\mathcal{S}$ receives only messages with the same SID (w.r.t. σ) from $\mathcal{F}$ and $\mathcal{E}$. So, even with $!\mathcal{S}$ only one instance of $\mathcal{S}$ will be invoked in every run of $\mathcal{E} \,|\, !\mathcal{S} \,|\, \mathcal{F}$ and we have $\mathcal{P} \cong_{\sigma\text{-single}} !\mathcal{S} \,|\, \mathcal{F}$. Now, it is easy to see that $!\mathcal{S} \,|\, \mathcal{F}$ is a protocol system (in particular, Condition (ii) of Definition 11 is satisfied) and a σ-session version. Furthermore, $!\mathcal{S} \,|\, \mathcal{F}$ is σ-environmentally bounded because $\mathcal{S} \,|\, \mathcal{F}$ is σ-environmentally bounded by definition of $\mathrm{Sim}^{\mathcal{P}}_{\sigma\text{-single}}(\mathcal{F})$. From this, by Theorem 6 (with $\mathcal{Q} = !\mathcal{S} \,|\, \mathcal{F}$), we obtain that $\mathcal{P} \cong !\mathcal{S} \,|\, \mathcal{F}$ and that $!\mathcal{S} \,|\, \mathcal{F}$ is environmentally bounded. The latter implies $!\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}(\mathcal{F})$. Hence, we conclude that $\mathcal{P} \leq^{SS} \mathcal{F}$. □

7.4 Composition Theorem for More Complex Systems

By iteratively applying Theorems 8, 9, and 10 and using transitivity of the $\leq^{SS}$ relation (cf. Lemma 13), one can construct more and more complex systems. For example, as an immediate consequence of Theorem 8 and 9 we obtain that if (an unbounded number of sessions of) an ideal protocol $\mathcal{F}$ is used as a component in a more complex system $\mathcal{Q}$, then it can be replaced by its realization $\mathcal{P}$:

Corollary 1. *Let $\mathcal{Q}$, $\mathcal{P}$, and $\mathcal{F}$ be protocol systems such that the following conditions are satisfied:*

1. *$!\underline{\mathcal{P}}$ is environmentally bounded and $\mathcal{P} \leq^{SS} \mathcal{F}$.*
2. *$\mathcal{Q}$ and $\mathcal{P}$ are I/O-connectable and $\mathcal{Q} \,|\, !\underline{\mathcal{P}}$ is environmentally bounded.*

Then, $\mathcal{Q} \,|\, !\underline{\mathcal{P}} \leq^{SS} \mathcal{Q} \,|\, !\underline{\mathcal{F}}$.

If we want $\mathcal{P}$ and $\mathcal{F}$ to be aware of their SIDs, we can use Theorem 10 instead of Theorem 9 and obtain the following corollary.

Corollary 2. *Let $\mathcal{Q}$, $\mathcal{P}$, and $\mathcal{F}$ be protocol systems such that $\mathcal{P}$ and $\mathcal{F}$ are σ-session versions for some SID function σ and the following conditions are satisfied:*

1. *$!\underline{\mathcal{P}}$ is environmentally bounded and $\mathcal{P} \leq^{SS}_{\sigma\text{-single}} \mathcal{F}$.*
2. *$\mathcal{Q}$ and $\mathcal{P}$ are I/O-connectable and $\mathcal{Q} \,|\, \mathcal{P}$ is environmentally bounded.*

Then, $\mathcal{Q} \,|\, \mathcal{P} \leq^{SS} \mathcal{Q} \,|\, \mathcal{F}$.

We can also use the composition theorems to analyze protocols based on ideal subroutines which can later be implemented by their realizations. For example, we immediately obtain the following corollary of Theorem 8:

Corollary 3. *Let $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{F}_1$, $\mathcal{F}_2$ be protocol systems such that the following holds true::*

1. *$\mathcal{P}_1 \leq^{SS} \mathcal{F}_1$.*
2. *$\mathcal{P}_2$ and $\mathcal{F}_1$ are I/O-connectable and $\mathcal{P}_2 \,|\, \mathcal{F}_1 \leq^{SS} \mathcal{F}_2$.*
3. *$\mathcal{P}_2 \,|\, \mathcal{P}_1$ is environmentally bounded.*

Then, $\mathcal{P}_2 \,|\, \mathcal{P}_1 \leq^{SS} \mathcal{F}_2$.

This corollary allows us to show $\mathcal{P}_1 \leq^{SS} \mathcal{F}_1$ in isolation, then prove the security of $\mathcal{P}_2$ using $\mathcal{F}_1$ as a subroutine, and then conclude that $\mathcal{P}_2$ using $\mathcal{P}_1$ as a subroutine is still secure. This is usually much easier then directly proving security of the protocol $\mathcal{P}_2 \,|\, \mathcal{P}_1$ because $\mathcal{P}_1$ might be a quite complex subroutine, employing various cryptographic primitives to ensure certain properties, whereas the ideal functionality $\mathcal{F}_1$ is usually quite simple and provides absolute security guarantees. Note that this process can be iterated arbitrarily often.

In particular, we can now build a protocol $\mathcal{P}_3$ on top of $\mathcal{F}_2$ to realize $\mathcal{F}_3$, say, and then, after performing a security proof, use Corollary 3 to replace $\mathcal{F}_2$ with $\mathcal{P}_2 \mid \mathcal{P}_1$. This process is one of the main features of universal composability frameworks as it allows for modular protocols and proofs where one can analyze several small (and relatively simple) protocol parts in isolation to obtain security of complex combined protocols.

These are just a few examples of what can be obtained by iteratively applying Theorems 8, 9, and 10. Further examples are provided in Section 10.

8 On the Composability of Runtime Notions

In this section, we discuss the composability of environmentally (almost/strictly) bounded systems. The first simple observation is that, in general, the composition of two environmentally (almost/strictly) bounded systems does not need to be environmentally bounded: For example, consider two environmentally (almost/strictly) bounded protocol systems $\mathcal{Q}_1$ and $\mathcal{Q}_2$ which connect via some external tapes. (The simplest example is that both $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are single IITMs.) Then, $\mathcal{Q}_1$ and $\mathcal{Q}_2$ could "play ping-pong" with each other, i.e., they could send messages back and forth forever, and hence, in such a case they are not environmentally bounded.

However, in applications the composition of environmentally almost/strictly bounded systems is basically always environmentally almost/strictly bounded (see Section 10.4 for examples). Clearly, one can construct examples, such as the above, where this is not the case. However, these examples typically do not occur in applications. Moreover, in applications it is typically easy to see whether a system, including the composition of two environmentally almost/strictly bounded systems, is environmentally almost/strictly bounded.

We also observe that in applications protocol systems are typically *strictly* bounded, and for such systems we obtain useful general composability statements, as presented in the following subsection. In Section 8.2 we show that some of these general statements do not hold true for environmentally almost bounded systems.

8.1 On the Composability of Environmentally Strictly Bounded Systems

The following lemma shows that the composition of two environmentally strictly bounded systems that have disjoint tapes, and hence, do not communicate directly, is environmentally strictly bounded.

Lemma 15. *Let $\mathcal{P}$ and $\mathcal{Q}$ be two environmentally strictly bounded protocol systems such that the sets of external tapes of $\mathcal{P}$ and $\mathcal{Q}$ are disjoint. Then, $\mathcal{P} \mid \mathcal{Q}$ is environmentally strictly bounded.*

Proof. Let $\mathcal{E}$ be a universally bounded system which can be connected to $\mathcal{P} \mid \mathcal{Q}$. We need to show that $\mathcal{E} \mid \mathcal{P} \mid \mathcal{Q}$ is strictly bounded.

First, we observe that because $\mathcal{E}$ is universally bounded there exists a polynomial p_0 such that for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$:

$$\mathrm{Time}_{\mathcal{E}}((\mathcal{E} \mid \mathcal{P} \mid \mathcal{Q})(1^\eta, a)) \leq p_0(\eta + |a|) \ .^{24} \tag{17}$$

In particular, the length of the overall output produced by $\mathcal{E}$ in a run of $(\mathcal{E} \mid \mathcal{P} \mid \mathcal{Q})(1^\eta, a)$ is bounded from above by $p_0(\eta + |a|)$. We thus can construct an environmental system $\mathcal{E}' \in \mathrm{Env}(\mathcal{P})$ that upon every activation produces random output such that for every sequence of messages sent from $\mathcal{E}$ to $\mathcal{P}$ in a run of $(\mathcal{E} \mid \mathcal{P} \mid \mathcal{Q})(1^\eta, a)$ the probability that this sequence of messages is sent from $\mathcal{E}'$ to $\mathcal{P}$ in a run of $(\mathcal{E}' \mid \mathcal{P})(1^\eta, a)$ is non-zero. (Here we need that $\mathcal{E}'$ knows the length of a. This is true since $\mathcal{P}$ and $\mathcal{Q}$ are protocol systems, and hence, $\mathsf{start} \notin \mathcal{T}(\mathcal{P} \mid \mathcal{Q})$ and $\mathcal{E}'$ may use start.).

Clearly, $\mathcal{E}'$ is universally bounded. Now, since $\mathcal{P}$ is environmentally strictly bounded, there exists a polynomial p_1 such that for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$:

$$\mathrm{Time}_{\mathcal{P}}((\mathcal{E}' \mid \mathcal{P})(1^\eta, a)) \leq p_1(\eta + |a|) \ .$$

[24] That is, $\mathrm{Time}_{\mathcal{E}}((\mathcal{E} \mid \mathcal{P} \mid \mathcal{Q})(1^\eta, a))(\alpha) \leq p_0(\eta + |a|)$ for all random coins $\alpha \in \mathsf{Rand}$.

Since with non-zero probability $\mathcal{E}'$ (running with $\mathcal{P}$) sends the same sequence of messages to $\mathcal{P}$ as $\mathcal{E}$ (running with $\mathcal{P} \,|\, \mathcal{Q}$), we deduce that for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$:

$$\mathrm{Time}_{\mathcal{P}}((\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q})(1^\eta, a)) \leq p_1(\eta + |a|) \ . \tag{18}$$

Analogously, there exists a polynomial p_2 such that for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$:

$$\mathrm{Time}_{\mathcal{Q}}((\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q})(1^\eta, a)) \leq p_2(\eta + |a|) \ . \tag{19}$$

Combining (17), (18), and (19), we obtain that for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$:

$$\mathrm{Time}((\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q})(1^\eta, a)) \leq p_0(\eta + |a|) + p_1(\eta + |a|) + p_2(\eta + |a|) \ .$$

Thus, $\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q}$ is strictly bounded. □

The above lemma can be generalized to protocol systems $\mathcal{P}$ and $\mathcal{Q}$ whose sets of external tapes are not disjoint, provided that the length of the messages that, say, $\mathcal{P}$ sends to $\mathcal{Q}$ is bounded by a polynomial in the security parameter plus the external input (of the environment) plus the input that $\mathcal{P}$ gets from the environment. To state this precisely, let $\mathcal{R}$, $\mathcal{R}'$, and $\mathcal{R}''$ be connectable systems. Then, we define $\mathrm{Flow}_{\mathcal{R}\to\mathcal{R}'}((\mathcal{R} \,|\, \mathcal{R}' \,|\, \mathcal{R}'')(1^\eta, a))$ to be the random variable over Rand which assigns to every $\alpha \in \mathsf{Rand}$ the overall length of all messages that were sent from $\mathcal{R}$ to $\mathcal{R}'$ (i.e., the output by $\mathcal{R}$ on input tapes of $\mathcal{R}'$) in the run of $\mathcal{R} \,|\, \mathcal{R}' \,|\, \mathcal{R}''$ with security parameter η, external input a, and random coins α. Here we count all messages even messages that are rejected in mode $\mathsf{CheckAddress}$. (Note that if the set of external tapes of $\mathcal{R}$ and $\mathcal{R}'$ are disjoint, then the flow from $\mathcal{R}$ and $\mathcal{R}'$ is empty.) Now we can prove the following generalization of Lemma 15.

Lemma 16. *Let $\mathcal{P}$ and $\mathcal{Q}$ be two environmentally strictly bounded protocol systems that are connectable and such that there exists a polynomial p such that for all $\mathcal{E} \in \mathrm{Env}(\mathcal{P} \,|\, \mathcal{Q})$:*

$$\mathrm{Prob}\left[\mathrm{Flow}_{\mathcal{P}\to\mathcal{Q}}((\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q})(1^\eta, a)) \leq p\big(\eta + |a| + \mathrm{Flow}_{\mathcal{E}\to\mathcal{P}}((\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q})(1^\eta, a))\big)\right] = 1$$

(i.e., the output from $\mathcal{P}$ to $\mathcal{Q}$ is bounded in length by a polynomial in the security parameter plus the length of the external input plus the length of the input $\mathcal{P}$ received from $\mathcal{E}$). Then, $\mathcal{P} \,|\, \mathcal{Q}$ is environmentally strictly bounded.

Proof. Let $\mathcal{E}$ and p be as in the assumption of the lemma. We may assume that p has only non-negative coefficients. Hence, $p(n) \leq p(n')$ for all $n, n' \in \mathbf{N}$ such that $n \leq n'$. Since $\mathcal{E}$ is universally bounded, we know that there exists a polynomial p' such that $\mathrm{Flow}_{\mathcal{E}\to\mathcal{P}}((\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q})(1^\eta, a)) \leq p'(\eta + |a|)$ (i.e., the probability for this is 1) and such that $\mathrm{Flow}_{\mathcal{E}\to\mathcal{Q}}((\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q})(1^\eta, a)) \leq p'(\eta + |a|)$. Hence, by assumption, $\mathrm{Flow}_{\mathcal{P}\to\mathcal{Q}}((\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q})(1^\eta, a)) \leq p(\eta + |a| + p'(\eta + |a|))$. So, $\mathrm{Flow}_{\mathcal{E}|\mathcal{P}\to\mathcal{Q}}((\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q})(1^\eta, a)) \leq p'(\eta + |a|) + p(\eta + |a| + p'(\eta + |a|))$. As in the proof of Lemma 15 and since $\mathcal{Q}$ is environmentally strictly bounded, we can deduce that there exits a polynomial q such that $\mathrm{Time}_{\mathcal{Q}}((\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q})(1^\eta, a)) \leq q(\eta + |a|)$. Hence, $\mathrm{Flow}_{\mathcal{E}|\mathcal{Q}\to\mathcal{P}}((\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q})(1^\eta, a)) \leq p'(\eta + |a|) + q(\eta + |a|)$. Now, again since $\mathcal{P}$ is environmentally strictly bounded and as in the proof of Lemma 15, we can conclude that there exists a polynomial q' such that $\mathrm{Time}_{\mathcal{P}}((\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q})(1^\eta, a)) \leq q'(\eta + |a|)$. So, $\mathrm{Time}_{\mathcal{Q}|\mathcal{P}}((\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q})(1^\eta, a)) \leq q(\eta + |a|) + q'(\eta + |a|)$. It follows that $\mathcal{P} \,|\, \mathcal{Q}$ is environmentally strictly bounded. □

The condition stated in Lemma 16 is often satisfied. For instance, if $\mathcal{P}$ and $\mathcal{Q}$ do not have common external tapes or in many settings where $\mathcal{Q}$ is a subprotocol of $\mathcal{P}$. However, we emphasize that even if this condition is not satisfied, it is generally easy to see whether the composition of two environmentally strictly bounded systems is environmentally strictly bounded. In fact, as already mentioned, we would argue that for natural protocol systems it basically never happens that the composition is not environmentally strictly bounded.

The next lemma shows that the multi-session version of an environmentally strictly bounded system is environmentally strictly bounded.

Lemma 17. *Let $\mathcal{Q}$ be an environmentally strictly bounded protocol system. Then, $!\underline{\mathcal{Q}}$ is environmentally strictly bounded.*

Proof. First, let $\mathcal{E} \in \mathrm{Env}(!\underline{\mathcal{Q}})$ be σ_{prefix}-single session, i.e., the environment $\mathcal{E}$ invokes only a single session of $\underline{\mathcal{Q}}$. Let $\mathcal{E}' \in \mathrm{Env}(\mathcal{Q})$ simulate $\mathcal{E}$ except that it strips off the SID with which $\mathcal{E}$ prefixes messages. Since $\mathcal{E}$ is universally bounded, $\mathcal{E}'$ is universally bounded too. Then it is easy to construct a bijection between runs of $\mathcal{E} \,|\, !\underline{\mathcal{Q}}$ and $\mathcal{E}' \,|\, \mathcal{Q}$ such that corresponding runs produce the same output and have the same probability of occurring.[25] Since, by assumption, $\mathcal{Q}$ is environmentally strictly bounded, there exists a polynomial p in the security parameter plus the length of the external input such that for all runs of $\mathcal{E}' \,|\, \mathcal{Q}$ the number of steps taken by $\mathcal{Q}$ is bounded by p. It follows that there exists a polynomial p' such that for every run of $\mathcal{E} \,|\, !\underline{\mathcal{Q}}$ the number of steps taken by the session invoked by $\mathcal{E}$ (i.e., the overall number of steps taken by machines in mode Compute belonging to that session) in this run is bounded p'.

Now, let $\mathcal{E} \in \mathrm{Env}(!\underline{\mathcal{Q}})$, where we do not require $\mathcal{E}$ to be σ_{prefix}-single session. Since $\mathcal{E}$ is universally bounded, there exists a polynomial q (in the security parameter plus the length of the external input) such that the overall number of steps taken by $\mathcal{E}$ (in any run with any system) is bounded by q. In particularly, the length of messages sent by $\mathcal{E}$ to $!\underline{\mathcal{Q}}$ is bounded by $q(\eta + |a|)$ (in any run of $(\mathcal{E} \,|\, !\underline{\mathcal{Q}})(1^\eta, a)$ for any $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$). Hence, similarly to the proof of Lemma 15, we can construct a universally bounded environment $\mathcal{E}' \in \mathrm{Env}(!\underline{\mathcal{Q}})$ which randomly picks an SID *sid* (one bounded in length by $q(\eta + |a|)$) and randomly picks at most $q(\eta + |a|)$ messages of length at most $q(\eta + |a|)$ and sends these messages prefixed by *sid* to $!\underline{\mathcal{Q}}$. By construction, $\mathcal{E}'$ is a σ_{prefix}-single session environment. Hence, by the above, we know that there exists a polynomial p such that for every run of $\mathcal{E}' \,|\, !\underline{\mathcal{Q}}$ the number of steps taken by the session invoked by $\mathcal{E}'$ in this run is bounded by p. Now, consider a run of $\mathcal{E} \,|\, !\underline{\mathcal{Q}}$ and, for some SID, all machines with this SID in this run. The probability that this SID is picked by $\mathcal{E}'$ and that $\mathcal{E}'$ produces exactly the same input for the machines with that SID is non-zero. Hence, it follows that the number of steps taken by these machines in the considered run of $\mathcal{E} \,|\, !\underline{\mathcal{Q}}$ is bounded by p.

Overall, we obtain that for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$:

$$\mathrm{Time}((\mathcal{E} \,|\, !\underline{\mathcal{Q}})(1^\eta, a)) \leq q(\eta + |a|) + q(\eta + |a|) \cdot p(\eta + |a|) \ .$$

Hence, $\mathcal{E} \,|\, !\underline{\mathcal{Q}}$ is strictly bounded. □

Clearly, the converse of Lemma 17 is also true, as stated in the next lemma. Furthermore, the multi-session version of the multi-session version of $\mathcal{Q}$, i.e., $!\underline{\underline{\mathcal{Q}}}$—in this system all messages are prefixed by two SIDs, i.e., they are of the form $m = (s_1, (s_2, m'))$—is environmentally bounded if $!\underline{\mathcal{Q}}$ is:

Lemma 18. *Let $\mathcal{Q}$ be a protocol system such that $!\underline{\mathcal{Q}}$ is environmentally strictly bounded. Then, $\mathcal{Q}$ and $!\underline{\underline{\mathcal{Q}}}$ are environmentally strictly bounded.*

The proof of this lemma is straightforward and therefore omitted.

8.2 On the Composability of Environmentally Almost Bounded Systems

We now show that Lemmas 15 and 17 (and hence, Lemma 16 because it is a generalization of Lemma 15) do not hold true for environmentally almost bounded protocol systems in general. However, as mentioned, for applications this is not really relevant since protocol systems are typically strictly bounded and the composition of such systems typically is strictly bounded as well.

Intuitively, Lemma 15 and Lemma 17 need that the runtime of a protocol is bounded by a polynomial no matter which sequence of messages (of polynomial length) is received from the environment. Environmentally almost bounded protocols do not have this property: There might be (sequences of) messages that trigger exponential or worse runtime. As long as every environment has only a negligible chance of finding such

[25] Here we use that $\mathcal{Q}$ is a protocol system, in particular Condition (ii) of Definition 11. Without this condition, there could be several copies of a machine in $!\underline{\mathcal{Q}}$, whereas in $\mathcal{Q}$ there is only one such copy. We also use that $\mathcal{Q}$ does not contain the tape start since the content of this tape would be interpreted differently by $\underline{\mathcal{Q}}$ and $\mathcal{Q}$.

"bad" messages the protocol would still be environmentally almost bounded. In the following, we use this to construct two protocol systems that are environmentally almost bounded but can be used to generate "bad" messages for each other. When running concurrently, an environment can then simply forward these messages to break polynomial runtime of the combined protocols.

For the construction of these protocols, we need the existence of so-called time-lock puzzles, a complexity assumption introduced in [37]. We define time-lock puzzles following [21]:[26]

Definition 21. A *time-lock puzzle* consists of an ITM V (the *verifier*) and an ITM P (the *prover*) such that the following conditions are satisfied, where by $\langle P, V\rangle$ we denote the distribution of the output of V after an interaction with P:

1. Given an argument of the form $(1^\eta, s)$, V runs in polynomial time in η. Given an argument of the form $(1^\eta, s)$, P runs in polynomial time in $\eta + s$.

2. *Easiness.* For every polynomial p we have that

$$\min_{s\leq p(\eta)} \text{Prob}\left[\langle P(1^\eta, s), V(1^\eta, s)\rangle = 1\right]$$

   is overwhelming (as a function in η). (We call s the *hardness* of the puzzle.)

3. *Hardness.* For any ITM B running in polynomial time in the length of its first two arguments (i.e., in $\eta + |a|$) there exists a polynomial p such that

$$\sup_{s\geq p(\eta+|a|)} \text{Prob}\left[\langle B(1^\eta, a, s), V(1^\eta, s)\rangle = 1\right]$$

   is negligible (as a function in η and a).

Lemma 19. *If time-lock puzzles exist, then the following statements are true.*

1. *There exist protocol systems $\mathcal{P}$ and $\mathcal{Q}$ with disjoint sets of external tapes such that $\mathcal{P}$ and $\mathcal{Q}$ are environmentally bounded but their composition $\mathcal{P} \mid \mathcal{Q}$ is not environmentally bounded.*

2. *There exists a protocol system $\mathcal{R}$ such that $\mathcal{R}$ is environmentally bounded but $!\underline{\mathcal{R}}$ is not environmentally bounded.*

Proof. Hofheinz et al. [21, footnote 21 on page 34] sketched that if time-lock puzzles exist, then there exist environmentally bounded protocol systems $\mathcal{P}$ and $\mathcal{Q}$ with disjoint sets of external tapes such that $\mathcal{P}$ and $\mathcal{Q}$ are environmentally bounded but $!\underline{\mathcal{P}} \mid !\underline{\mathcal{Q}}$ is not environmentally bounded. Their idea can be used as follows to show that there even exist environmentally bounded systems $\mathcal{P}$ and $\mathcal{Q}$ such that $\mathcal{P} \mid \mathcal{Q}$ is not environmentally bounded.

Let (V_A, P_A) and (V_B, P_B) be two time-lock puzzles (with verifiers V_A/V_B and provers P_A/P_B). We define $\mathcal{P}$ and $\mathcal{Q}$ to have disjoint tapes. Also both systems are defined to have one input and one output tape and they accept every message in mode CheckAddress. Upon input s (this is the hardness of the puzzle), $\mathcal{P}$ simulates $V_A(1^\eta, s)$ (if V_A sends a message to P_A, $\mathcal{P}$ outputs this message on its output tape and if V_A wants to receive a message, $\mathcal{P}$ waits for input on its input tape and then gives this input to V_A) until V_A halts and outputs a bit b. If $b = 0$ (i.e., the verifier does not accept), then $\mathcal{P}$ halts. Otherwise, $\mathcal{P}$ continues and simulates $P_B(1^\eta, 2s)$ (here $2s$ means "2 times s" where s is interpreted as a non-negative integer). When P_B halts, $\mathcal{P}$ goes back to the start and again waits for input s, to simulate $V_A(1^\eta, s)$ and so on. $\mathcal{Q}$ is defined as $\mathcal{P}$ except that the puzzles A and B are swapped, i.e., $\mathcal{Q}$ simulates V_B and P_A instead of V_A and P_B. So, $\mathcal{P}$ ($\mathcal{Q}$) verifies a puzzle of type A (B) of hardness s and then solves a puzzle of type B (A) of hardness $2s$.

[26] We note that in [21], for the hardness assumption, the runtime of B does not depend on the length of the external input a, but only on the security parameter η. However, since here the runtime of environments depends on the length of a and, in the proof of the following lemma, we need that environments play the role of B, we adapted this notion. For uniform environments one could drop the external input altogether.

We now show that $\mathcal{P}$ and $\mathcal{Q}$ (in separation) are environmentally bounded. For this purpose, let $\mathcal{E}$ be an environment of $\mathcal{P}$. Since $\mathcal{E}$ is universally bounded, by the hardness assumption of the puzzles, there exists a polynomial p_{hard} such that $\mathcal{E}$ can only solve puzzles of hardness $s \leq p_{hard}(\eta + |a|)$ (except with negligible probability), where η is the security parameter and a is the external input to $\mathcal{E}$. Let $s_1, \ldots, s_n$ be the hardnesses that $\mathcal{E}$ sends to $\mathcal{P}$ in a run of $(\mathcal{E} \,|\, \mathcal{P})(1^\eta, a)$. Note that in every run $n \leq p_{\mathcal{E}}(\eta + |a|)$, where $p_{\mathcal{E}}$ is a polynomial that bounds the number of steps taken by $\mathcal{E}$. By definition of time-lock puzzles, there exist polynomials p_V and p_P such that, for all $i \leq n$, the runtime of (the simulated) $V_A(1^\eta, s_i)$ is bounded by $p_V(\eta)$ and the runtime of $P_B(1^\eta, s_i)$ is bounded by $p_P(\eta + s_i)$. We may assume that p_P has only non-negative coefficients. Hence, $p_P(l) \leq p_P(l')$ for all $l, l' \in \mathbf{N}$ with $l \leq l'$. As argued above, the simulated $V_A(1^\eta, s_i)$, $i \leq n$, accepts (i.e., outputs 1) only if $s_i \leq p_{hard}(\eta + |a|)$ (except with negligible probability). Hence, $P_B(1^\eta, 2s_i)$, $i \leq n$, is only simulated if $s_i \leq p_{hard}(\eta + |a|)$. We obtain that the overall runtime of $\mathcal{P}$ is bounded by

$$\sum_{i=1}^{n} p_V(\eta) + \sum_{i=1}^{n} p_P(\eta + 2s_i) \leq p_{\mathcal{E}}(\eta + |a|) \cdot p_V(\eta) + p_{\mathcal{E}}(\eta + |a|) \cdot p_P(\eta + 2p_{hard}(\eta + |a|))$$

(except with negligible probability). Hence, $\mathcal{P}$ is environmentally bounded.[27] Analogously, $\mathcal{Q}$ is environmentally bounded.

Now, we show that the system $\mathcal{P} \,|\, \mathcal{Q}$ is not environmentally bounded. Therefore, we construct an environment $\mathcal{E}$ as follows. Basically $\mathcal{E}$ will start with a puzzle of hardness 1 for $\mathcal{P}$, solves it itself and then forwards all messages between $\mathcal{P}$ and $\mathcal{Q}$ for η many runs of the puzzles. At the end, a puzzle of hardness 2^η is solved. Now we describe $\mathcal{E}$ in more detail. First, $\mathcal{E}$ sends hardness 1 to $\mathcal{P}$ and simulates $P_A(1^\eta, 1)$ with $\mathcal{P}$. So, $V_A(1^\eta, 1)$, simulated by $\mathcal{P}$, accepts (except with negligible probability), and hence, $\mathcal{P}$ starts the simulation of $P_B(1^\eta, 2)$. Then, $\mathcal{E}$ sends hardness 2 to $\mathcal{Q}$, which then simulates $V_B(1^\eta, 2)$, where $\mathcal{E}$ forwards all messages between $\mathcal{P}$ (and hence, $P_B(1^\eta, 2)$) and $\mathcal{Q}$ (and hence, $V_B(1^\eta, 2)$). As a result, $V_B(1^\eta, 2)$, simulated by $\mathcal{Q}$, accepts, and hence, $\mathcal{Q}$ starts the simulation of $P_A(1^\eta, 4)$. $\mathcal{E}$ sends hardness 4 to $\mathcal{P}$ and again forwards all messages between $\mathcal{P}$ and $\mathcal{Q}$. This is iterated by $\mathcal{E}$ η times, always doubling the hardness. At the end, a puzzle of hardness 2^η is solved, so, except with negligible probability, the overall runtime of $\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q}$ can not be polynomially bounded in $\eta + |a|$ (by the hardness assumption of the puzzles). Thus, $\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q}$ is not almost bounded. Since clearly $\mathcal{E}$ is universally bounded, this means that $\mathcal{P} \,|\, \mathcal{Q}$ is not environmentally bounded. This proves the first statement of the lemma.

We now prove the second statement using the first one. Let $\mathcal{P}$ and $\mathcal{Q}$ be protocol systems as above. That is, $\mathcal{P}$ and $\mathcal{Q}$ are environmentally bounded but there exists an environment $\mathcal{E}$ of $\mathcal{P} \,|\, \mathcal{Q}$ such that $\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q}$ is not almost bounded. We define the protocol system $\mathcal{R}$ as follows: $\mathcal{R}$ chooses a bit b uniformly at random. If $b = 0$, $\mathcal{R}$ behaves exactly like $\mathcal{P}$ and otherwise it behaves exactly like $\mathcal{Q}$. (The tape names have to be adjusted appropriately.) Now, $\mathcal{R}$ is environmentally bounded because $\mathcal{P}$ and $\mathcal{Q}$ are environmentally bounded, and $\mathcal{R}$ runs either only $\mathcal{P}$ or only $\mathcal{Q}$. However, $!\underline{\mathcal{R}}$ is not environmentally bounded: We construct an environment $\mathcal{E}'$ of $!\underline{\mathcal{R}}$ as follows: $\mathcal{E}'$ uses two sessions of $\underline{\mathcal{R}}$ and simulates $\mathcal{E}$ where messages exchanged with $\mathcal{P}/\mathcal{Q}$ are now exchanged with the first/second session of $\underline{\mathcal{R}}$, where now SIDs are used to address these sessions. Clearly, $\mathcal{E}'$ is universally bounded because $\mathcal{E}$ is universally bounded. Now, with probability $\frac{1}{4}$, the system $\mathcal{E}' \,|\, !\underline{\mathcal{R}}$ behaves exactly as $\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{Q}$, and hence, $\mathcal{E}' \,|\, !\underline{\mathcal{R}}$ is not almost bounded. □

We remark that the lemma holds both for uniform and non-uniform environments. Note that the environments constructed in the above proof are uniform, i.e., they ignore the external input.

The following lemma states that Lemma 18 also holds for environmentally almost bounded systems. Again, this lemma is easy to prove.

Lemma 20. *Let $\mathcal{R}$ be a protocol system such that $!\underline{\mathcal{R}}$ is environmentally almost bounded. Then, $\mathcal{R}$ and $!\underline{\underline{\mathcal{R}}}$ are environmentally almost bounded.*

[27] Note that $\mathcal{P}$ is not environmentally strictly bounded because there is a negligible chance for the environment to solve a puzzle of hardness $s > p_{hard}(\eta + |a|)$. This is why Lemma 19 does not hold true for environmentally strictly bounded systems.

9 On Basing Universal Composability on Environmentally Strictly Bounded Systems

As already mentioned before, in applications the systems one has to deal with are typically environmentally strictly bounded. Also, as shown in Section 8, environmentally strictly bounded systems enjoy useful composability properties. Therefore, a question that suggests itself is whether we obtain useful universal composability security notions if we formulate these notions based on environmentally strictly bounded systems, instead of environmentally almost bounded systems: we would assume the protocol system $\mathcal{P}$ to be environmentally strictly bounded and, for strong simulatability, we would require $\mathcal{S} \,|\, \mathcal{F}$ to be environmentally strictly bounded; similarly, the other security notions would be adapted.

In this section, we show that basing the security notions on environmentally *strictly* bounded systems, rather than environmentally almost bounded systems, yields unsuitable security notions. More specifically, we show the following properties:

1. The composition theorem for a constant number of protocol systems (the analog of Theorem 8) would not hold for any of the new security notions.
2. Strong simulatability and dummy UC would not be transitive relationships anymore.
3. Strong simulatability and dummy UC would not imply UC anymore.

We note, however, that the composition theorem for the unbounded self-composition of systems (the analog of Theorem 9) would still hold. The proof would even be simpler: due to Lemma 17 it would follow immediately that $!(\mathcal{S} \,|\, \mathcal{F})$ and all hybrid systems considered in the proof of this theorem are environmentally strictly bounded. In the case of environmentally almost bounded systems this required a more tailored and involved proof, namely, the proof of Lemma 9.

We further remark that Hofheinz et al. [21] considered a security definition conceptually similar to our strict variant of UC (they do not consider strict variants of SS or dummy UC) and that they showed that, for their notion, the universal composition theorem in the UC model does not hold. The counterexample used in their proof could also be used to show that the composition theorem does not hold for our strict variant of UC. However, the counterexample we use below is simpler and more general. In particular, it can be used to show that strict dummy UC and strict strong simulatability are unsuitable security notions too, a fact that could not have been proven with their example.

In what follows, we first introduce the security notions based on environmentally strictly bounded systems and then prove the negative results mentioned above.

9.1 Strict Simulatability

In order to define the security notions based on environmentally strictly bounded systems, we first need to introduce some notation.

In Section 6.2, we required that the system composed of an adversary/simulator and a protocol system is environmentally almost bounded. Now, we require that this composition is environmentally *strictly* bounded. Therefore, we introduce the sets $\mathrm{Adv}_{\mathrm{strict}}(\mathcal{S})$ and $\mathrm{Sim}^{\mathcal{S}'}_{\mathrm{strict}}(\mathcal{S})$, which are defined just as $\mathrm{Adv}(\mathcal{S})$ and $\mathrm{Sim}^{\mathcal{S}'}(\mathcal{S})$ (see Definition 18), respectively, except that for $\mathcal{A} \in \mathrm{Adv}_{\mathrm{strict}}(\mathcal{S})$ or $\mathcal{A} \in \mathrm{Sim}^{\mathcal{S}'}_{\mathrm{strict}}(\mathcal{S})$ we now require that $\mathcal{A} \,|\, \mathcal{S}$ is environmentally strictly bounded. We emphasize that, as before, $\mathcal{A}$ itself is not required to be environmentally (strictly) bounded.

Definition 22. Let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems such that $\mathcal{P}$ is environmentally strictly bounded.

1. *Strict Strong Simulatability (strict SS):* $\mathcal{P} \leq^{SS}_{\mathrm{strict}} \mathcal{F}$ iff $\exists\, \mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{\mathrm{strict}}(\mathcal{F})$: $\mathcal{P} \cong \mathcal{S} \,|\, \mathcal{F}$.
2. *Strict Universal Simulatability/Composability (strict UC):* $\mathcal{P} \leq^{UC}_{\mathrm{strict}} \mathcal{F}$ iff $\forall\, \mathcal{A} \in \mathrm{Adv}_{\mathrm{strict}}(\mathcal{P})\ \exists\, \mathcal{I} \in \mathrm{Sim}^{\mathcal{A} \,|\, \mathcal{P}}_{\mathrm{strict}}(\mathcal{F})$: $\mathcal{A} \,|\, \mathcal{P} \cong \mathcal{I} \,|\, \mathcal{F}$.

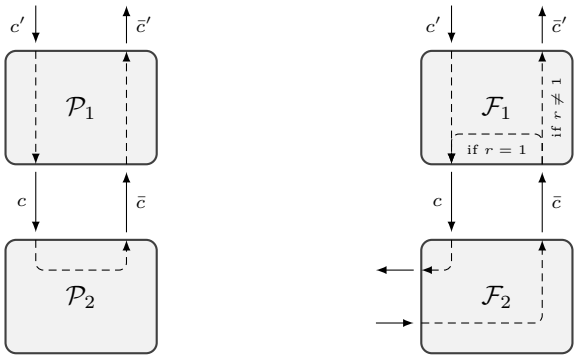

Figure 8: IITMs constructed in the proof of Lemma 22.

3. *Dummy Version of strict UC (strict dummyUC):* $\mathcal{P} \leq^{dumUC}_{\text{strict}} \mathcal{F}$ iff $\exists\, \mathcal{I} \in \text{Sim}^{\mathcal{D}\,|\,\mathcal{P}}_{\text{strict}}(\mathcal{F})$: $\mathcal{D}\,|\,\mathcal{P} \cong \mathcal{I}\,|\,\mathcal{F}$ where $\mathcal{D} = \mathcal{D}(\mathcal{T}^{net}_{in}(\mathcal{P}), \mathcal{T}^{net}_{out}(\mathcal{P}))$.

Analogously, we could define strict variants of black-box and reactive simulatability. However, from the results presented for above notions it can be easily seen that the strict variants of black-box and reactive simulatability yield unsuitable security notions as well.

Using Lemmas 4 and 5, it is easy to see that strict SS is equivalent to strict dummyUC and that strict UC implies strict SS (and hence, strict dummyUC):

Lemma 21. *Let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems such that $\mathcal{P}$ is environmentally strictly bounded. Then, $\mathcal{P} \leq^{SS}_{\text{strict}} \mathcal{F}$ if and only if $\mathcal{P} \leq^{dumUC}_{\text{strict}} \mathcal{F}$ and $\mathcal{P} \leq^{SS}_{\text{strict}} \mathcal{F}$ if $\mathcal{P} \leq^{UC}_{\text{strict}} \mathcal{F}$.*

Since strict dummyUC and strict SS are equivalent, in what follows, we will discuss only strict SS and strict UC.

9.2 No Universal Composability for a Constant Number of Protocol Systems

We now show that the analog of Theorem 8, the composition theorem for a constant number of protocol systems, does not hold for strict SS and strict UC:

Lemma 22. *There exist protocol systems $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{F}_1$, and $\mathcal{F}_2$ that satisfy the following conditions:*

1. *$\mathcal{P}_1$ and $\mathcal{P}_2$ as well as $\mathcal{F}_1$ and $\mathcal{F}_2$ are I/O-connectable, respectively.*
2. *$\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_1\,|\,\mathcal{P}_2$, $\mathcal{F}_1$, $\mathcal{F}_2$, and $\mathcal{F}_1\,|\,\mathcal{F}_2$ are environmentally strictly bounded.*
3. *For all $i \in \{1, 2\}$: $\mathcal{P}_i \leq^{SS}_{\text{strict}} \mathcal{F}_i$ and $\mathcal{P}_i \leq^{UC}_{\text{strict}} \mathcal{F}_i$.*
4. *$\mathcal{P}_1\,|\,\mathcal{P}_2 \not\leq^{SS}_{\text{strict}} \mathcal{F}_1\,|\,\mathcal{F}_2$ and $\mathcal{P}_1\,|\,\mathcal{P}_2 \not\leq^{UC}_{\text{strict}} \mathcal{F}_1\,|\,\mathcal{F}_2$.*

Proof. We define the systems $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{F}_1$, and $\mathcal{F}_2$ as follows; see Figure 8 for an illustration of the systems. Every system is a single IITM. The systems $\mathcal{P}_2$ and $\mathcal{F}_2$ have the same I/O interface which consists of an I/O input tape c and an I/O output tape $\bar{c}$. The system $\mathcal{F}_2$ additionally has a network input tape and a network output tape but $\mathcal{P}_2$ does not have any network tapes. The systems $\mathcal{P}_1$ and $\mathcal{F}_1$ have no network tapes and they have the same I/O interface which consists of two I/O input tapes $\bar{c}$ and c' and two I/O output tapes c and $\bar{c}'$. That is, with the tapes c and $\bar{c}$, they connect to $\mathcal{P}_2/\mathcal{F}_2$. Clearly, $\mathcal{P}_1$ and $\mathcal{P}_2$ as well as $\mathcal{F}_1$ and $\mathcal{F}_2$ are I/O-connectable, respectively. These systems are defined to accept all messages in mode CheckAddress.

In mode Compute the machines act as follows: When receiving a message m on c' (resp. $\bar{c}$) the machine $\mathcal{P}_1$ outputs m on c (resp. $\bar{c}'$). When receiving a message m on c the machine $\mathcal{P}_2$ outputs m on $\bar{c}$. When receiving a message m on the I/O tape c, the machine $\mathcal{F}_2$ outputs m on its network tape. When receiving a

message m on the network tape, m is output on $\bar{c}$. Note that $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{F}_2$ are simple dummy IITMs which merely forward all messages they receive. The machine $\mathcal{F}_1$ also only forwards messages but to which tape a message is forwarded depends on a random choice. More specifically, $\mathcal{F}_1$, upon its first activation in mode Compute, first chooses $r \in \{1, \ldots, 2^\eta\}$ uniformly at random (where η is the security parameter). If $r \neq 1$, then $\mathcal{F}_1$ forwards messages from c' to c and from $\bar{c}$ to $\bar{c}'$ just like $\mathcal{P}_1$. Otherwise (i.e., $r = 1$), $\mathcal{F}_1$ still forwards messages from c' to c but it forwards messages from $\bar{c}$ to c. That is, with the overwhelming probability $1 - 2^{-\eta}$, $\mathcal{F}_1$ behaves exactly as $\mathcal{P}_1$. But with the negligible probability $2^{-\eta}$, $\mathcal{F}_1$ forwards all messages to c.

It is easy to see that $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_1 \,|\, \mathcal{P}_2$, $\mathcal{F}_1$, $\mathcal{F}_2$, and $\mathcal{F}_1 \,|\, \mathcal{F}_2$ are environmentally strictly bounded (both for uniform and non-uniform environments). Furthermore, it is easy to see that $\mathcal{P}_1 \leq^{SS}_{\text{strict}} \mathcal{F}_1$ and $\mathcal{P}_1 \leq^{UC}_{\text{strict}} \mathcal{F}_1$, because, except with negligible probability, $\mathcal{F}_1$ behaves exactly like $\mathcal{P}_1$ (no simulator is needed). It is also easy to see that $\mathcal{P}_2 \leq^{SS}_{\text{strict}} \mathcal{F}_2$ and $\mathcal{P}_2 \leq^{UC}_{\text{strict}} \mathcal{F}_2$: the simulator can simply replay messages output by $\mathcal{F}_2$ back to $\mathcal{F}_2$.

Next, we show that $\mathcal{P}_1 \,|\, \mathcal{P}_2 \not\leq^{SS}_{\text{strict}} \mathcal{F}_1 \,|\, \mathcal{F}_2$. Note that this implies $\mathcal{P}_1 \,|\, \mathcal{P}_2 \not\leq^{UC}_{\text{strict}} \mathcal{F}_1 \,|\, \mathcal{F}_2$ because strict UC implies strict SS (Lemma 21). As a warming up, we note that the simulator $\mathcal{S}$ that simply replays messages output by $\mathcal{F}_2$ back to $\mathcal{F}_2$ (as it can be used to prove $\mathcal{P}_2 \leq^{SS}_{\text{strict}} \mathcal{F}_2$) is not a "good" simulator to prove $\mathcal{P}_1 \,|\, \mathcal{P}_2 \leq^{SS}_{\text{strict}} \mathcal{F}_1 \,|\, \mathcal{F}_2$ because the system $\mathcal{S} \,|\, \mathcal{F}_1 \,|\, \mathcal{F}_2$ is not environmentally *strictly* bounded: If $r = 1$ (in $\mathcal{F}_1$), which happens with probability $2^{-\eta}$, and an environment sends a message to $\mathcal{S} \,|\, \mathcal{F}_1 \,|\, \mathcal{F}_2$ via c', then this message circles between $\mathcal{F}_1$, $\mathcal{F}_2$, and $\mathcal{S}$ forever, and hence, the run does not terminate. (Note, however, that $\mathcal{S} \,|\, \mathcal{F}_1 \,|\, \mathcal{F}_2$ is environmentally almost bounded and with the simulator $\mathcal{S}$ we obtain $\mathcal{P}_1 \,|\, \mathcal{P}_2 \leq^{SS} \mathcal{F}_1 \,|\, \mathcal{F}_2$).

We now show that $\mathcal{P}_1 \,|\, \mathcal{P}_2 \not\leq^{SS}_{\text{strict}} \mathcal{F}_1 \,|\, \mathcal{F}_2$ by contradiction. Assume that $\mathcal{P}_1 \,|\, \mathcal{P}_2 \leq^{SS}_{\text{strict}} \mathcal{F}_1 \,|\, \mathcal{F}_2$. Then, there exists $\mathcal{S} \in \text{Sim}^{\mathcal{P}_1 \,|\, \mathcal{P}_2}_{\text{strict}}(\mathcal{F}_1 \,|\, \mathcal{F}_2)$ such that $\mathcal{P}_1 \,|\, \mathcal{P}_2 \cong \mathcal{S} \,|\, \mathcal{F}_1 \,|\, \mathcal{F}_2$.

First, we show that $\mathcal{S}$ has basically no choice but to forward all messages it receives back to $\mathcal{F}_2$. Note that $\mathcal{S}$ only has two network tapes (one input and one output tape) which connect to $\mathcal{F}_2$. For all $\eta, i \in \mathbf{N}$, let $p_{\eta,i}$ denote the probability that $\mathcal{S}$ (with security parameter η) outputs 1 in the first i consecutive activations with input 1. (Here, the probability is taken over the random coins of $\mathcal{S}$.) We now show that the probability $p_{\eta,i}$ is overwhelming for all polynomially bounded i (in η). Let q be a polynomial in η. We construct $\mathcal{E} \in \text{Env}(\mathcal{P}_1 \,|\, \mathcal{P}_2)$ as follows: $\mathcal{E}$ sends the message 1 on tape c' (i.e., to $\mathcal{P}_1$ or $\mathcal{F}_1$) and waits to receive 1 on $\bar{c}'$. If $\mathcal{E}$ does not receive 1 (either $\mathcal{E}$ is activated with empty input on tape start or it receives some other message on $\bar{c}'$), $\mathcal{E}$ outputs 0 on decision. If $\mathcal{E}$ receives 1 on $\bar{c}'$, it again sends the message 1 on tape c' and behaves just as before. The system $\mathcal{E}$ does this $q(\eta)$ times, unless it outputs 0 on tape decision. If in all these iterations, $\mathcal{E}$ received 1 back from $\mathcal{F}_2$, $\mathcal{E}$ outputs 1 on decision. Clearly, if $\mathcal{E}$ interacts with $\mathcal{P}_1 \,|\, \mathcal{P}_2$, it will always output 1 on decision, i.e., $\text{Prob}[(\mathcal{E} \,|\, \mathcal{P}_1 \,|\, \mathcal{P}_2)(1^\eta, a) = 1] = 1$ for all η, a. If $\mathcal{E}$ interacts with $\mathcal{S} \,|\, \mathcal{F}_1 \,|\, \mathcal{F}_2$, then $\mathcal{E}$ will output 1 only if $\mathcal{S}$ forwarded 1 $q(\eta)$ times (and $r \neq 1$; where r is the number chosen by $\mathcal{F}_1$). That is, $\text{Prob}[(\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}_1 \,|\, \mathcal{F}_2)(1^\eta, a) = 1] \leq p_{\eta,q(\eta)}$, for all η, a. Since $|\text{Prob}[(\mathcal{E} \,|\, \mathcal{P}_1 \,|\, \mathcal{P}_2)(1^\eta, a) = 1] - \text{Prob}[(\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}_1 \,|\, \mathcal{F}_2)(1^\eta, a) = 1]|$ is negligible (as a function in η and a), $1 - p_{\eta,q(\eta)}$ is negligible, i.e., $p_{\eta,q(\eta)}$ is overwhelming. In particular, there exists $\eta_0 \in \mathbf{N}$ such that $p_{\eta,q(\eta)} > 0$ for all $\eta \geq \eta_0$.

Now, we use the above to show that $\mathcal{S} \,|\, \mathcal{F}_1 \,|\, \mathcal{F}_2$ is not environmentally strictly bounded, which contradicts the assumption that $\mathcal{S} \in \text{Sim}^{\mathcal{P}_1 \,|\, \mathcal{P}_2}_{\text{strict}}(\mathcal{F}_1 \,|\, \mathcal{F}_2)$. For this purpose, let $\mathcal{E} \in \text{Env}(\mathcal{S} \,|\, \mathcal{F}_1 \,|\, \mathcal{F}_2)$ be the following system: $\mathcal{E}$ sends the message 1 on tape c' (i.e., to $\mathcal{P}_1$ or $\mathcal{F}_1$) and then halts. Since, by assumption, $\mathcal{S} \,|\, \mathcal{F}_1 \,|\, \mathcal{F}_2$ is environmentally strictly bounded, there exists a polynomial p such that for every security parameter $\eta \in \mathbf{N}$ and external input a the runtime of the system $\mathcal{Q} := \mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}_1 \,|\, \mathcal{F}_2$ is bounded by $p(\eta)$ for all runs of this system, i.e., $\text{Time}(\mathcal{Q}(1^\eta, a)) \leq p(\eta)$ (note that $\mathcal{E}$ ignores the external input a). Let $q(\eta) := p(\eta) + 1$ for all $\eta \in \mathbf{N}$. If $r = 1$ in $\mathcal{F}_2$, which happens with probability $2^{-\eta}$, then in such a run $\mathcal{S}$ is activated (at least) $q(\eta)$ times with 1 and we know from what we have shown above that there exists $\eta_0 \in \mathbf{N}$ such that $p_{\eta,q(\eta)} > 0$ for all $\eta \geq \eta_0$, i.e., the probably that $\mathcal{S}$ will return 1 at least $q(\eta)$ times is non-zero. In particular, there exists a run of $\mathcal{Q}$ with security parameter η_0 (the external input a is ignore by $\mathcal{E}$, and hence, does not matter) that has non-zero probability such that the runtime of this run is bigger than $p(\eta_0)$, a contradiction. This shows that $\mathcal{S} \,|\, \mathcal{F}_1 \,|\, \mathcal{F}_2$ is not environmentally strictly bounded. We conclude that $\mathcal{P}_1 \,|\, \mathcal{P}_2 \not\leq^{SS}_{\text{strict}} \mathcal{F}_1 \,|\, \mathcal{F}_2$. □

We note that the above proof works for both uniform and non-uniform environments. Hence, the statement

of the lemma also holds in a uniform setting. We also note that the protocol systems constructed in the above proof cannot be used to break the composition theorem for the unbounded self-composition of systems, which, as mentioned earlier, holds true even for environmentally strictly bounded protocol systems. This is because one cannot encode the systems $\mathcal{P}_1 \,|\, \mathcal{P}_2$ and $\mathcal{F}_1 \,|\, \mathcal{F}_2$ into appropriate protocol systems $!\,\mathcal{P}$ and $!\,\mathcal{F}$. If one encodes the whole system $\mathcal{P}_1 \,|\, \mathcal{P}_2$ in one session of $!\,\mathcal{P}$ and the whole system $\mathcal{F}_1 \,|\, \mathcal{F}_2$ in one session of $!\,\mathcal{F}$, then one cannot show that $\mathcal{P}$ single session realizes $\mathcal{F}$ by the same argument as in the above proof. This, however, is the main requirement for the composition theorem for unbounded self-composition. If one encodes $\mathcal{P}_1$ and $\mathcal{P}_2$ (and also $\mathcal{F}_1$ and $\mathcal{F}_2$) in different sessions of $!\,\mathcal{P}$ (and $!\,\mathcal{F}$, respectively), then their behavior differs from the the combined system $\mathcal{P}_1 \,|\, \mathcal{P}_2$ as both protocols cannot communicate with each other directly (recall that different sessions cannot interact with each other directly, only via the environment/adversary).

We emphasize that the protocol systems $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{F}_2$ defined in the above proof are simple dummy IITMs which only forward/replay messages. The IITM $\mathcal{F}_1$ is simple too: It only forwards/replays messages and the "switch" r determines where to forward/replay the messages. Moreover, the simulators that are used in the above proof to show that $\mathcal{P}_1$ realizes $\mathcal{F}_1$ and $\mathcal{P}_2$ realizes $\mathcal{F}_2$ are also very simple: In fact, for $\mathcal{P}_1$ and $\mathcal{F}_1$ there is no simulator because these systems do not have any network tapes. The simulator $\mathcal{S}$ which is used to show that $\mathcal{P}_2$ realizes $\mathcal{F}_2$ is a simple dummy IITM which replays all messages back to $\mathcal{F}_2$. In particular, $\mathcal{S}$ is environmentally strictly bounded. Altogether, since the protocols and simulators considered are all very simple, it is quite hard to restrict the class of protocols and/or simulators in order to obtain reasonable security notions based on environmental strict boundedness. Possible approaches are to enforce some acyclicity conditions on the flow of messages between protocols and/or simulators such that the cycles vanish. Such approaches have been taken in related work but they are unsatisfactory as discussed in Section 11.

9.3 No Transitivity

The proof of the following lemma, which shows that $\leq^{SS}_{\text{strict}}$ is not transitive, is very similar to the proof of Lemma 22.

Lemma 23. *There exist protocol systems $\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{Q}_3$ with pairwise disjoint sets of network tapes such that the following conditions are satisfied:*

1. *$\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{Q}_3$ are environmentally strictly bounded.*
2. *$\mathcal{Q}_1 \leq^{SS}_{\text{strict}} \mathcal{Q}_2$.*
3. *$\mathcal{Q}_2 \leq^{SS}_{\text{strict}} \mathcal{Q}_3$.*
4. *$\mathcal{Q}_1 \not\leq^{SS}_{\text{strict}} \mathcal{Q}_3$.*

Proof. Let $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{F}_1$, and $\mathcal{F}_2$ be the protocol systems defined in the proof of Lemma 22. Furthermore, let $\mathcal{F}_2'$ be obtained from $\mathcal{F}_2$ by renaming the network tapes such that the set of network tapes of $\mathcal{F}_2$ and $\mathcal{F}_2'$ are disjoint. We define:

$$\begin{aligned} \mathcal{Q}_1 &:= \mathcal{P}_1 \,|\, \mathcal{P}_2 \ , \\ \mathcal{Q}_2 &:= \mathcal{P}_1 \,|\, \mathcal{F}_2 \ , \\ \mathcal{Q}_3 &:= \mathcal{F}_1 \,|\, \mathcal{F}_2' \ . \end{aligned}$$

From the proof of Lemma 22, we already know that $\mathcal{Q}_1$ and $\mathcal{Q}_3$ are environmentally strictly bounded and that $\mathcal{Q}_1 \not\leq^{SS}_{\text{strict}} \mathcal{Q}_3$. Moreover, it is easy to see that $\mathcal{Q}_2 = \mathcal{P}_1 \,|\, \mathcal{F}_2$ is environmentally strictly bounded. It remains to show that $\mathcal{Q}_1 \leq^{SS}_{\text{strict}} \mathcal{Q}_2$ and $\mathcal{Q}_2 \leq^{SS}_{\text{strict}} \mathcal{Q}_3$.

In order to show $\mathcal{Q}_1 \leq^{SS}_{\text{strict}} \mathcal{Q}_2$ it suffices to consider a simulator $\mathcal{S}_1$ which simply replays all messages from $\mathcal{Q}_2$ back to $\mathcal{Q}_2$. It is easy to see that $\mathcal{S}_1 \,|\, \mathcal{Q}_2$ is environmentally strictly bounded. With this, $\mathcal{Q}_1 \leq^{SS}_{\text{strict}} \mathcal{Q}_2$ follows immediately.

In order to show $\mathcal{Q}_2 \leq^{SS}_{\text{strict}} \mathcal{Q}_3$ it clearly suffices to consider a simulator $\mathcal{S}_3$ which simply forwards all messages from $\mathcal{Q}_3$ to the environment and vice versa. Again, it is easy to see that $\mathcal{S}_2 \mid \mathcal{Q}_3$ is environmentally strictly bounded. □

We note that the above lemma does not carry over to strict UC (with the notation in the proof of Lemma 22 we have that $\mathcal{Q}_2 \not\leq^{UC}_{\text{strict}} \mathcal{Q}_3$; this can be shown analogously to the proof of Lemma 22). In fact, strict UC can easily been seen to be transitive:

Lemma 24. *Let $\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{Q}_3$ be environmentally strictly bounded protocol systems. If $\mathcal{Q}_1 \leq^{UC}_{\text{strict}} \mathcal{Q}_2$ and $\mathcal{Q}_2 \leq^{UC}_{\text{strict}} \mathcal{Q}_3$, then $\mathcal{Q}_1 \leq^{UC}_{\text{strict}} \mathcal{Q}_3$.*

Proof. The proof is analogous to the proof of Lemma 13. □

9.4 Strict SS does not Imply Strict UC (Incompleteness of the Dummy Adversary)

The following lemma states that strict SS does not imply strict UC.

Lemma 25. *There exist environmentally strictly bounded protocol systems $\mathcal{P}$ and $\mathcal{F}$ such that $\mathcal{P} \leq^{SS}_{\text{strict}} \mathcal{F}$ and $\mathcal{P} \not\leq^{UC}_{\text{strict}} \mathcal{F}$.*

Proof. Let $\mathcal{Q}_2$ and $\mathcal{Q}_3$ be defined as in the proof of Lemma 23. With $\mathcal{P} = \mathcal{Q}_2$ and $\mathcal{F} = \mathcal{Q}_3$, the lemma follows. □

Note that this lemma means that the dummy adversary is not complete (see Remark 12).

10 Instantiation of the IITM Model

The IITM model is a very flexible and expressive model which allows us to model cryptographic protocols in various ways. This section provides several examples of how various types of protocols can be modeled and analyzed in the IITM model. More specifically, in Section 10.1, we suggest one possible way of modeling (multi-party) protocols in the IITM model, including the addressing of multiple sessions as well as the modeling of subprotocols and corruption; depending on the kind of setting, other approaches might be favorable. In order to illustrate the flexibility of the model, we also briefly discuss composition with joint state in Section 10.2 and composition with global setup in Section 10.3. These aspects required major changes in other models (or entirely new models) and new composition theorems. In the IITM model, joint and global state can be dealt with seamlessly. In Section 10.4, we present concrete protocols and functionalities and illustrate how they can be modeled in the IITM model (see also, e.g., [3, 25–30] for other examples). In Section 10.5, we briefly discuss another possible instantiation of the IITM model based on the SUC model from [8]. We emphasize that these sections focus on a small selection of examples and mainly serve as an illustrative starting point for working with the IITM model. Due to the generality of the IITM model, many other instantiations, protocol types, and modeling choices are conceivable and fully supported as well, going beyond the capabilities of other models. In fact, the iUC model [3] is one possible instantiation of the IITM model (and its extension with responsive environments proposed in [2]) that focuses on ease of use for protocol designers by providing a simple yet flexible and expressive set of conventions for modeling protocols.

10.1 Modeling of Real Protocols and Ideal Functionalities

Structure of real protocols. A real protocol with n roles can be modeled in the IITM model as the following protocol system:

$$\mathcal{P} := !\, M_1 \mid \cdots \mid !\, M_n \tag{20}$$

where M_i, for $i \in \{1, \ldots, n\}$, is an IITM which models the ith role. Every machine is in the scope of a bang operator to model multiple sessions of the protocol (see below). Moreover, every machine M_i has I/O

and network tapes. The network tapes are used to communicate with other machines over the (untrusted) network. As usual, the network is controlled by the adversary, and hence, all messages sent on a network tape go directly to the adversary and all messages received on a network tape come from the adversary. Network tapes are also used to model corruption (see below). The I/O tapes are not controlled by the adversary. They are used by a machine M_i to communicate with the environment, such as (honest) users of the protocol or higher level protocols. For example, if $\mathcal{P}$ is a key exchange protocol, then an I/O tape would be used to output a successfully established session key (to some user or, for example, to an instance of a secure channel protocol).

Potentially, the I/O tapes can also be used by a machine M_i to communicate with other roles directly, rather than through the adversary or environment. In this case, M_i would directly be connected with another machine, say M_j, via I/O tapes. This is useful in many settings, for example, for modeling joint, shared, and global state as well as components global parameters to other protocol components (see the following subsection for more details).

Note that this definition of $\mathcal{P}$ models a finite number of roles. Of course, the IITM model can also represent an unbounded number of roles. For example, one might use the above system for $n = 1$ and encode different roles in machine M_1. That is, each instance of M_1 not only models a single session but rather models a specific role in a session. Because M_1 is in the scope of a bang, we can thus obtain an unbounded number of roles during a protocol run (see also below for how such roles can be addressed).

Structure of ideal functionalities. One way of modeling an ideal functionality (ideal protocol) $\mathcal{F}$ is simply by a single IITM in the scope of a bang, i.e.,

$$\mathcal{F} := !\, M_{\mathcal{F}} \tag{21}$$

where $M_{\mathcal{F}}$ has the same external I/O tapes as $\mathcal{P}$ and different network tapes. Note that $M_{\mathcal{F}}$ comprises the external I/O tapes of all machines $M_1, \ldots, M_n$ of $\mathcal{P}$. Usually, every instance of $M_{\mathcal{F}}$ models a single session, e.g., a single session of a key exchange protocol. A single instance of $M_{\mathcal{F}}$ then handles all inputs and outputs from the environment for *all* instances from the real protocol *in the same session*; in particular, this usually includes instances of different roles M_i. This allows $M_{\mathcal{F}}$ to maintain a single internal state for the whole session, unlike the real protocol where the state of one session is scattered among instances of different roles. For example, an instance of $M_{\mathcal{F}}$ modeling an ideal key exchange protocol would generate the session key just once and then output the same key to all participants. In order to realize such an ideal functionality, the real protocol would then have to make use of network communication and suitable cryptographic primitives to ensure that all participants in the same session end up with the same key.

Just as for real protocols, the IITM model also allows for various other modelings of ideal functionalities. For example, in the literature one often adds so-called dummy machines/parties to the ideal protocol, serving as forwarders for inputs and outputs to/from the environment. This can be modeled as

$$\mathcal{F} := !\, M'_1 \mid \cdots \mid !\, M'_n \mid !\, M_{\mathcal{F}} \tag{22}$$

where M'_i, for $i \in \{1, \ldots, n\}$, is an IITM with the same external I/O tapes as M_i and two additional input and output tapes connecting to $M_{\mathcal{F}}$. Every instances of M'_i corresponds to exactly one instance of M_i in the real protocol, but acts only as a forwarder for inputs/outputs between $M_{\mathcal{F}}$ and the environment. The machine $M_{\mathcal{F}}$ has the same purpose as above, i.e., it usually handles a whole session of a protocol in an ideal way, but now receives and sends inputs/outputs from/to the environment via instances of M'_i.

In the following, we assume that $\mathcal{F} := !\, M_{\mathcal{F}}$, i.e., we use the modeling without dummy machines, as this modeling is the conceptually simpler one.

Addressing of multiple sessions of real protocols. Due to the general concept of the mode CheckAddress there are many possible ways of how multiple sessions of a protocol can be addressed. We now describe one such approach, which is based on pre-established globally unique session identifiers (SIDs). This approach is the standard way of addressing protocol instances in universal composable models and it is what the

composition theorems expect. That is, protocols are modeled as session versions (see Section 5.2.1) or σ_{prefix}-session versions[28] (see Section 5.3.1); see also below. As argued in [28], pre-established SIDs are not always appropriate and desired. We refer the reader to [28] for an addressing mechanism formulated in the IITM model where SIDs are not pre-established, locally chosen and managed (see also below).

More specifically, to address multiple instances of a machine (role) M_i, and hence, multiple sessions of $\mathcal{P}$, by using (global) SIDs, M_i can be defined (as a σ_{prefix}-session version) as follows: In mode CheckAddress, the machine M_i accepts an incoming message only if it is of the form (sid, m) for some SID sid and some message m. In mode Compute, the machine M_i records the SID contained in the first message it accepted. Later it will only accept messages in mode CheckAddress which are prefixed with the recorded SID, say sid. Moreover, in mode Compute the machine M_i will only output messages that are prefixed with sid.

This guarantees that in a run of $\mathcal{P}$ (with some environment) there is at most one instance of every M_i for every SID sid; we denote such an instance by $M_i[sid]$ and say that this instance is *addressed by sid*. The instances $M_1[sid], \ldots, M_n[sid]$ (or a subset thereof if not all instances are present in a run) form a session of the protocol $\mathcal{P}$, the *session with SID sid*. We say that $M_i[sid]$ *belongs to session sid*. Note that all instances within one session share the same SID. This SID is globally unique, is given to an instance from outside the protocol, and is pre-established by the parties participating in one session, in the sense that the SID is established before the actual protocol starts to run.

Addressing of multiple sessions of ideal functionalities. The machine $M_{\mathcal{F}}$ in the ideal functionality $\mathcal{F}$ handles sessions in the same way as the machines M_i. Thus, in a run there will be at most one instance of $M_{\mathcal{F}}$ per session sid, denoted by $M_{\mathcal{F}}[sid]$. As mentioned above, this single instance $M_{\mathcal{F}}[sid]$ handles the inputs/outputs to/from the environment for the session sid of $\mathcal{P}$, i.e., for the instances $M_1[sid], \ldots, M_n[sid]$ (or a subset thereof).

As mentioned before, $\mathcal{P}$ and $\mathcal{F}$ are σ_{prefix}-session versions. In particular, the composition theorems can be applied, and hence, it suffices to reason about $\mathcal{P}$ in the single-session setting (see below). Note that, by modeling M_i to be a σ_{prefix}-session version, an instance $M_i[sid]$ of M_i addressed by sid is aware of its SID sid, i.e., it can use sid in its computation, and, for example, include it in messages to be signed/encrypted. If this is not necessary, one could model M_i as a session version of some IITM M_i', i.e., $M_i = \underline{M_i'}$ (see Section 5.2.1). In this case, M_i would be completely oblivious to its SID.

Typically, an SID is structured and contains, in addition to the actual SID, the names of the parties involved in the session. For example, to model that in one session s the ith role is played by party pid_i the SID would be of the form $sid = (s, pid_1, \ldots, pid_n)$ and a machine M_i would be defined in such a way that an instance $M_i[sid]$ would run the ith role as party pid_i. Clearly, this can be generalized to let several parties run one role in one session of the protocol. Moreover, if multiple roles are encoded in a single machine M_i, then the SID might also include an identifier for the role that is to be activated. Such an identifier could even contain machine code that is then executed by M_i.

The modeling of globally unique and pre-established SIDs as described above is used in all other models for universal composability and it is hard-wired into these models (see Section 11). The IITM model is flexible enough to allow for other forms of addressing multiple protocol sessions. In particular, in [28] an alternative way of addressing multiple protocol session without pre-established and globally unique SIDs is presented within the IITM model. In this formulation, parties merely use locally chosen and managed SIDs. As further discussed in [28], this allows for a *faithful* analysis of protocols that do not use pre-established SIDs, which is the case for most real-world authentication, key exchange, and secure channel protocols. This approach has been successfully used in [25] to faithfully model and analyze several key exchange protocols from practice that do not pre-establish global SIDs.

Security proofs using composition theorems. In the universal composability paradigm, security of a protocol $\mathcal{P}$, as modeled above, typically means that it realizes some appropriate ideal protocol $\mathcal{F}$ (e.g., in the

[28] Recall that, for all messages m and tapes c, $\sigma_{\text{prefix}}(m, c) := sid$ if $m = (sid, m')$ for some sid, m' and $\sigma_{\text{prefix}}(m, c) := \bot$ otherwise.

case of a key exchange protocol, an ideal key exchange functionality), i.e., $\mathcal{P}$ is considered secure (w.r.t. $\mathcal{F}$) if $\mathcal{P} \leq^{SS} \mathcal{F}$. Of course, one can attempt to prove $\mathcal{P} \leq^{SS} \mathcal{F}$ directly. But this would require a proof which has to consider multiple concurrent sessions. Using the composition theorems for unbounded self-composition (Theorem 9, if the protocol does not depend on the SIDs, i.e., $M_i = \underline{M_i'}$ for some M_i'; or Theorem 10 if the protocol depends on the SIDs) simplifies this proof because one has to consider only a single session of the protocol: For example, by Theorem 10, except for some (typically simple) checks concerning the runtime of the system, one has to show only that $\mathcal{P} \leq^{SS}_{\sigma_{\text{prefix-single}}} \mathcal{F}$ to obtain that $\mathcal{P} \leq^{SS} \mathcal{F}$. Hence, roughly speaking, it suffices to show that $M_1[sid] \mid \cdots \mid M_n[sid]$ realizes $M_{\mathcal{F}}[sid]$ for one session sid.

Subprotocols and ideal functionalities. Complex protocols can often be/are often structured in a hierarchy of higher- and lower-level protocols. For example, a secure channel protocol might use a key exchange protocol or an authenticated channel as a subprotocol, and cryptographic primitives (such as encryption or digital signatures) could be modeled as subprotocols (see Section 10.4 and [25, 28, 29]).

For the sake of the discussion here, let $\mathcal{P}' = !\, M_1' \mid \cdots \mid !\, M_n'$ be a subprotocol of $\mathcal{P}$ with the same structure as $\mathcal{P}$ and where the addressing of machines is defined just as for $\mathcal{P}$. Since $\mathcal{P}'$ is supposed to be a subprotocol of $\mathcal{P}$, the machines in $\mathcal{P}'$ typically connect via I/O tapes to the corresponding machines in $\mathcal{P}$, i.e., M_i' and M_i are connected via I/O tapes. By the addressing defined above for $\mathcal{P}$ and $\mathcal{P}'$, every instance $M_i[sid]$ will, via the I/O tapes, only interact with $M_i'[sid]$ (since these instances output messages only of the form (sid, m) for some message m and such messages are not accepted by other instances). Instead of the subprotocol $\mathcal{P}'$, $\mathcal{P}$ might be connected to an ideal protocol (or ideal functionality) $\mathcal{F}'$ that provides the same I/O interface as $\mathcal{P}'$ but provides the functionality of $\mathcal{P}'$ in an ideal way (e.g., an ideal key exchange or an ideal cryptographic primitive).

Structuring a protocol like this again simplifies the proof of security of a (complex) protocol because the subprotocol can be analyzed in separation and then $\mathcal{P}$ can be analyzed based on the ideal protocol as follows: To prove that $\mathcal{P} \mid \mathcal{P}'$ is secure, i.e., $\mathcal{P} \mid \mathcal{P}' \leq^{SS} \mathcal{F}$ for some ideal protocol/functionality $\mathcal{F}$, it suffice to show that

$$\mathcal{P}' \leq^{SS}_{\sigma_{\text{prefix-single}}} \mathcal{F}'$$

for some appropriate ideal protocol/functionality $\mathcal{F}'$ and that

$$\mathcal{P} \mid \mathcal{F}' \leq^{SS}_{\sigma_{\text{prefix-single}}} \mathcal{F} \ .$$

From this, using the composition theorems (Theorem 8 and 10) and transitivity of $\leq^{SS}$ (Lemma 13), it follows immediately that $\mathcal{P} \mid \mathcal{P}' \leq^{SS} \mathcal{F}$, which means that multiple sessions of $\mathcal{P}$, where every such session may use a session of $\mathcal{P}'$, realizes multiple sessions of $\mathcal{F}$. We emphasize that both proof steps require merely single-session reasoning and that the second proof step is further simplified because the subprotocol/functionality used by $\mathcal{P}$ is idealized.

We note that the IITM model and the composition theorems are flexible enough to deal with much more complex scenarios than the one described above. For instance:

1. The reasoning can be iterated: $\mathcal{P}'$ itself could use subprotocols and the composition theorems can be used to simplify the proof of $\mathcal{P}' \leq^{SS}_{\sigma_{\text{prefix-single}}} \mathcal{F}'$, just as in the case of $\mathcal{P}$ above.
2. $\mathcal{P}$ might use more than just one subprotocol in parallel.
3. $\mathcal{P}$ could use multiple sessions of $\mathcal{P}'$ per session. In this case, the sessions of $\mathcal{P}'$ could, for instance, be addressed by a hierarchical SID (sid, sid', m) where sid is the SID that $\mathcal{P}$ uses and sid' is an extra SID that is used to address the sessions of $\mathcal{P}'$ within a session of $\mathcal{P}$. However, again single-session reasoning would suffice for both $\mathcal{P}$ and $\mathcal{P}'$ to establish security properties for $\mathcal{P}$ (in composition with $\mathcal{P}'$) for multiple sessions.
4. Obviously, $\mathcal{P}'$ is not restricted to have the same structure as $\mathcal{P}$. For example, $\mathcal{P}'$ could be a two-party/two-role protocol and $\mathcal{P}$ could be an n-party/n-role protocol where every two parties of one session in $\mathcal{P}$ use one session of $\mathcal{P}'$. This way, for example, an n-party key exchange protocol could be built from a two-party key exchange protocol.

Modeling corruption. In the IITM model, the way corruption is modeled is not fixed and hard-wired into the model but is part of the specification of protocols, with the advantage that (i) the IITM model is simple, (ii) general theorems proven in the IITM model, such as composition theorems, hold true independently of how corruption is modeled, and (iii) corruption can be modeled in a very flexible way. We now describe one possible way of modeling corruption in the IITM model. Clearly, other ways are possible and in some cases might be desirable.

Modeling corruption of real protocols. The adversary (or environment) who connects to the network interface of a real protocol $\mathcal{P}$ may send a special `Corrupt` message to a network tape of (some instance of) a machine M_i in $\mathcal{P}$. When M_i receives such a message it considers itself corrupted and outputs its complete configuration to the adversary. From then on M_i forwards all messages between the I/O and network interface, i.e., the adversary is in full control of the corrupted instance.[29] (clearly other options for defining the behavior of corrupted instances are conceivable and useful as well). If $\mathcal{P}$ uses subprotocols/functionalities, as described above, by this the adversary would also gain full access to the I/O interface of the subprotocols the corrupted instance of M_i has access to. This models fully adaptive, active corruption of single instances. We note that, as always in universal composability settings, the distinguishing environment should have the possibility to know which instances are corrupted because, otherwise, a simulator could always corrupt instances in the ideal world and then perfectly simulate the real world, i.e., every protocol system would realize every other protocol system (with the same I/O interface). Therefore, a machine M_i is defined in such a way that, on the I/O interface, it accepts requests of the form `CorrStatus`? and answers *true* if it is has been corrupted, i.e., if it has received a `Corrupt` message on the network interface before. As a result, an environment can ask whether an instance is corrupted.

If $\mathcal{P}$ uses subprotocols/functionalities, then, typically, the corruption status of a machine in $\mathcal{P}$ also depends on the corruption status of the subprotocols it uses. That is, an instance of a machine M_i might consider itself corrupted also if one of its subprotocol instances is corrupted (a fact that M_i can check by sending a `CorrStatus`? request). Note, however, that even if M_i returns to the environment that it is corrupted (because some part of the subprotocol is corrupted), then this does not necessarily mean that M_i has to consider itself completely controlled by the adversary. (Clearly, if desired, M_i could be modeled in such a way that in this case it considers itself to be fully controlled by the adversary.)

For example, if M_i models an instance of a key exchange protocol and uses a functionality for public-key encryption, then M_i, if asked whether it is corrupted, would return *yes* if it has been corrupted directly or if its public-key functionality has been corrupted, because in this case it could not provide security guarantees. It makes sense to model M_i in such a way that, even though the public-key functionality that M_i uses is corrupted, it still follows its prescribed protocol: the fact that the private key has been stolen by the adversary does not necessarily mean that every instance that uses the key is completely controlled by the adversary. However, if desired, one could also model M_i in such a way that if one of its subprotocols is corrupted, then M_i considers itself controlled by the adversary. This depends on the kind of corruption one would like to consider. Conversely, the adversary could corrupt only M_i but not the public-key functionality that M_i uses, which would model that the private key of M_i is still not known by the adversary (e.g., because it is stored on a smart card), but the process (the instance of M_i) that uses the private key is corrupted.

It should be clear that the way of corruption sketched above allows for a very fine-grained and flexible modeling of corruption, ranging from the corruption of single instances to the corruption of complete parties. In order to corrupt a complete party, the adversary can corrupt every instance of M_i (and subprotocol instances used by this instance) that belong to the party an adversary wants to control. For such a form of corruption, instances would typically check whether one of their subprotocols are corrupted and then consider themselves completely controlled by the adversary as well. If one wants to make sure that if an instance is corrupted, then also all its subinstances are, an instance could check that if it has been explicitly corrupted by the adversary that then all its subinstances have been corrupted as well (and if this is not the case it could wait until the adversary has explicitly corrupted all subinstances).

So far we have not put a restriction on when corruption can occur, and hence, we modeled adaptive

[29] However, this instance of M_i would still make sure addressing conventions are followed. For example, if it is a σ_{prefix}-session version, it would still behave like a σ_{prefix}-session version with the SID it already has.

corruption. Clearly, static corruption can be modeled as well: For this purpose, upon its first activation (an instance of a) machine could first ask the adversary whether he wants to corrupt the machine. Subsequent corrupt messages would then be ignored by the machine.

While, as introduced above, explicit corruption meant that a machine provides its complete configuration to the adversary, other forms of corruption where a machine gives away, for example, merely its long-term or ephemeral keys are conceivable as well.

Modeling corruption of ideal functionalities. So far we have discussed corruption only for real protocols. Corruption of an ideal protocol $\mathcal{F}$ is similar, however, less details of the internal behavior can be fixed a priori as those depend on the specific function that is modeled. More precisely, recall that every instance of $M_{\mathcal{F}}$ in $\mathcal{F}$ corresponds to potentially several instances of different roles M_i of $\mathcal{P}$. An instance of $M_{\mathcal{F}}$ keeps track of the corruption status of all corresponding instances in the real protocol, where they are initially uncorrupted. The adversary (or environment) on the network can send a special $(\texttt{Corrupt}, m)$ message to an instance of $M_{\mathcal{F}}$ in $\mathcal{F}$, where m is an arbitrary bit string that is used to specify which corresponding instance of $\mathcal{P}$ gets corrupted. The exact behavior of $M_{\mathcal{F}}$ upon corruption of some (or all) of the corresponding instances is different depending on the ideal functionality. It has to be specified by the protocol designer as this strongly depends on the task that the ideal functionality is supposed to model. For example, an ideal signature functionality might enable forgery of signatures upon corruption of the owner of a key, but not care about corruption of other parties. Another example is an ideal key exchange functionality which might allow the adversary to determine the session key as soon as at least one party in a session is corrupted. Just as the real protocol $\mathcal{P}$, the ideal protocol $\mathcal{F}$ also allows the environment/a higher level protocol to send a special `CorrStatus`? input on any of the external I/O tapes. The machine $M_{\mathcal{F}}$ checks for which of the corresponding instances of $\mathcal{P}$ the request was issued (based on the tape the request was received on and potentially other information), and then returns its current corruption status.

10.2 Composition with Joint State and Shared State

In many protocols, parties/sessions/machine instances are required to share some kind of state. A common example for such shared state are long-term keys, such as public/private key pairs or shared long-term symmetric keys, that are re-used across several protocol runs. It is, of course, straightforward to model a (practical) protocol with shared state in the IITM model. In particular, there are no restrictions imposed on how machines and instances interact with each other, and the flexible addressing mechanism allows for handling multiple sessions and parties in the same instance. For example, a protocol $\mathcal{P}$ of the form $!M_1 \mid \cdots \mid !M_n$ as introduced above could be extended by a subprotocol, say an ideal functionality $\mathcal{F}$ (or its realization), such that in every run all instances of M_i access the same instance of $\mathcal{F}$.[30] We note that instances which share state cannot be analyzed in isolation as they do not constitute a σ-session version, i.e., the composition theorems for unbounded self-composition are not applicable. This it not a problem in the IITM model as it also provides a general composition theorem that works for arbitrary protocols.

In contrast to the IITM model, which allows for directly modeling protocols with shared state and then composing these protocols in arbitrary ways, many other models are built such that protocols cannot directly share any state between sessions. To overcome this limitation, Canetti and Rabin [14] were the first to propose composition theorems that allow for joint state—so-called composition theorems with joint state, or simply joint state (composition) theorems. These theorems essentially allow for analyzing protocols that share state in the single-session setting while obtaining security in the multi-session setting, as long as state is shared only in specific ways that are indistinguishable from a setting without shared state. (See Section 11 and [26] for a discussion of [14].) We note that the joint state theorems of [14] (and similar theorems), which we discuss more in more detail in the following, allow only for sharing state between multiple sessions of the same protocol. The IITM model also supports other types of joint state theorems, such as sharing state between multiple different protocols, which have not been considered in the literature so far. We briefly discuss an example at the end of this section.

[30] For this purpose, the CheckAddress mode of $\mathcal{F}$ could be defined in such a way that all message are accepted.

In [14], Canetti and Rabin first proposed a general joint state theorem (in the UC framework). In [26], this theorem was stated in the IITM model and it was shown that, unlike in the UC framework, in the IITM model it is a direct consequence of the composition theorem for a constant number of systems (Theorem 8) and the formulation of this theorem does not require to introduce a joint state operator. Formulated in the IITM model, the general joint state theorem states that if $\mathcal{P}' \leq^{SS} \mathcal{F}'$, then $\mathcal{P} \,|\, \mathcal{P}' \leq^{SS} \mathcal{P} \,|\, \mathcal{F}'$, where $\mathcal{F}'$ models some multi-session version (with disjoint state) of some ideal functionality and $\mathcal{P}'$ is supposed to be a realization of $\mathcal{F}'$ that utilizes some joint state across the sessions, e.g., $\mathcal{P}'$ uses only a single instance of $\mathcal{F}'$ for all sessions. Clearly, in the IITM model, this general joint state theorem is a direct consequence of Theorem 8. In the UC framework, it requires to extend the model and the notation and it requires a proof.

The general joint state theorem by itself does not say what a joint state realization looks like. The main challenge is always to find suitable joint state realizations for concrete ideal functionalities. As an example, we consider the joint state realization for public-key encryption, following [26]. In [26], an ideal functionality $\mathcal{F}_{\text{pke}}$ for public-key encryption is defined; the details of this functionality are not important for this discussion.[31] This functionality is the "encryption/decryption-box" of one party. In particular, it encapsulates the public/private key pair of that party, where the private key stays in the functionality (except if the functionality is corrupted) and the public key is given out, and hence, can be distributed. The system $!\underline{\mathcal{F}_{\text{pke}}}$ describes the multi-party version of $\mathcal{F}_{\text{pke}}$, i.e., in every run there is at most one instance $\underline{\mathcal{F}_{\text{pke}}}[pid]$ of $\underline{\mathcal{F}_{\text{pke}}}$ per party pid. The system $!\underline{\underline{\mathcal{F}_{\text{pke}}}}$ is the multi-session and multi-party version of $\mathcal{F}_{\text{pke}}$, i.e, there may be multiple sessions per party and instances of $\mathcal{F}_{\text{pke}}$ are addresses by identifiers of the form (sid, pid), denoting the session sid of party pid. In [26], a joint state realization $\mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\underline{\mathcal{F}_{\text{pke}}}$ is proposed, where $\mathcal{P}^{\text{js}}_{\text{pke}}$ is a kind of multiplexer which handles multiple sessions per party but where encryption and decryption of all sessions of one party pid are handled by the instance $\underline{\mathcal{F}_{\text{pke}}}[pid]$ of that party, i.e., if $\mathcal{P}^{\text{js}}_{\text{pke}}$ is asked to encrypt/decrypt a message for pid in some session (sid, pid'), $\mathcal{P}^{\text{js}}_{\text{pke}}$ uses the instance $\underline{\mathcal{F}_{\text{pke}}}[pid]$ for that purpose. The basic idea is that SIDs sid are added to messages to be encrypted. Upon decryption in session sid, it is checked whether a message contains sid. By this, it is prevented that ciphertexts created in one session can be used in other sessions. In [26], it has been proven that the proposed joint state realization in fact realizes the multi-session and multi-party version of $\mathcal{F}_{\text{pke}}$, i.e.,

$$\mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\underline{\mathcal{F}_{\text{pke}}} \leq^{SS} !\underline{\underline{\mathcal{F}_{\text{pke}}}} \ . \tag{23}$$

With such a joint state realization it is possible to prove that a protocol $\mathcal{P}$ that uses public-key encryption is secure in a multi-session setting where a party uses the same public/private key across all sessions by reasoning about just a single session of $\mathcal{P}$. To illustrate this, let $\mathcal{P}$ be a (multi-session and multi-party) protocol of the form (20) that uses $\mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\underline{\mathcal{F}_{\text{pke}}}$ for encryption. Recall that an instance of M_i is addressed by an SID of the form $sid = (s, pid_1, \ldots, pid_n)$. If such an instance wants to encrypt a message m for party pid_j it would send an encryption request containing the message m to $\mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\underline{\mathcal{F}_{\text{pke}}}$. For addressing purposes, the request also contains the SID $(s, pid_1, \ldots, pid_n)$ and the PID pid_j. With this, the message m is encrypted using the instance $\underline{\mathcal{F}_{\text{pke}}}[pid_j]$. (More precisely, by the definition of $\mathcal{P}^{\text{js}}_{\text{pke}}$, $\mathcal{P}^{\text{js}}_{\text{pke}}$ asks $\underline{\mathcal{F}_{\text{pke}}}[pid_j]$ to encrypt the message $((s, pid_1, \ldots, pid_n), m)$.)

Now, assume that we want to prove that $\mathcal{P} \,|\, \mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\underline{\mathcal{F}_{\text{pke}}}$ realizes $\mathcal{F}$, where $\mathcal{F}$ is a multi-session and multi-party formulation of some ideal functionality. Then it suffices to show that:

$$\mathcal{P} \,|\, !\underline{\underline{\mathcal{F}_{\text{pke}}}} \leq^{SS}_{\sigma_{\text{prefix}}\text{-single}} \mathcal{F} \ . \tag{24}$$

Note that to prove (24) only a single session $(s, pid_1, \ldots, pid_n)$ of $\mathcal{P}/\mathcal{F}$ needs to be analyzed. Such a session might contain n instances of $\mathcal{F}_{\text{pke}}$, one for each pid_i. The analysis of such a session is further simplified due to the use of $\mathcal{F}_{\text{pke}}$, i.e., *ideal* public-key encryption.

[31] The main idea of this functionality is that when given a message m to be encrypted, it encrypts a random message of the length of m instead of encrypting m. The resulting ciphertext c is stored in $\mathcal{F}_{\text{pke}}$ along with m. This ciphertext, by construction, does not contain any information about m, except for the length of m. If later, $\mathcal{F}_{\text{pke}}$ is asked to decrypt c, it looks up the corresponding plaintext in the table and returns this plaintext. We refer the reader to [26] for details.

From (24), using the joint state composition theorem for public-key encryption (23), the composition theorems (Theorem 8 and 10), and transitivity of $\leq^{SS}$ (Lemma 13), it immediately follows that

$$\mathcal{P} \,|\, \mathcal{P}_{\text{pke}}^{\text{js}} \,|\, !\,\underline{\mathcal{F}_{\text{pke}}} \leq^{SS} \mathcal{F} \ . \tag{25}$$

Moreover, if $\mathcal{P}_{\text{pke}}$ is a realization of $\mathcal{F}_{\text{pke}}$, i.e., $\mathcal{P}_{\text{pke}} \leq^{SS} \mathcal{F}_{\text{pke}}$ ($\mathcal{P}_{\text{pke}}$ could, for example, be an IND-CCA2-secure public-key encryption scheme), we obtain by Theorem 8 and 9 that

$$\mathcal{P} \,|\, \mathcal{P}_{\text{pke}}^{\text{js}} \,|\, !\,\underline{\mathcal{P}_{\text{pke}}} \leq^{SS} \mathcal{F} \ . \tag{26}$$

This says that $\mathcal{P}$, modeling a multi-session, multi-party version of some protocol which uses public-key encryption, realizes $\mathcal{F}$, modeling a multi-session, multi-party version of an ideal functionality. By the joint state realization $\mathcal{P}_{\text{pke}}^{\text{js}} \,|\, !\,\underline{\mathcal{P}_{\text{pke}}}$, $\mathcal{P}$ uses only one public/private key pair for every party across all sessions of $\mathcal{P}$. We emphasize that to prove (26), we needed to show only (24). (Note that proving (23) is a once and for all proof, which does not depend on the context in which public-key encryption is used.)

However, as already mentioned before, in the realization $\mathcal{P} \,|\, \mathcal{P}_{\text{pke}}^{\text{js}} \,|\, !\,\underline{\mathcal{P}_{\text{pke}}}$ the SIDs $(s, pid_1, \ldots, pid_n)$ are added to all encrypted messages (this is what $\mathcal{P}_{\text{pke}}^{\text{js}}$ does). While this is a good design principle, existing, in particular real-world protocols, typically do not follow this pattern. Hence, such protocols cannot be analyzed with the joint state theorem sketched above without severely modifying the protocols: adding SIDs to plaintexts is a severe modification of a protocol, which can turn an insecure protocol into a secure one. In order to analyze a protocol without modifying it, one can resort to multi-session analysis which is fully supported by the IITM model. Alternatively, in [28] composition and joint state theorems were proposed (within the IITM model) which allow for establishing the security of a protocol w.r.t. multiple sessions by analyzing only a single session of the protocol, but without requiring to change the protocol by adding SIDs in messages or in any other way, as illustrated for several real-world protocols in [28].

Sharing state between different protocols. So far we have focused on joint state as defined by Canetti and Rabin [14], which allows for replacing a single functionality $\mathcal{F}$ that has disjoint sessions with a realization $\mathcal{P}$ that can share state between sessions in arbitrary ways. Note that this is quite specific: Only a single protocol $\mathcal{F}/\mathcal{P}$ is considered, and state is shared only between sessions of that protocol. However, other types of joint state are often also desirable, such as sharing some state between several different protocols. Consider the following example: suppose we have shown that $\mathcal{P}_1 \,|\, \mathcal{F}_{\text{sig}_1} \leq^{SS} \mathcal{F}_1$, i.e., some protocol $\mathcal{P}_1$ using an ideal signature functionality $\mathcal{F}_{\text{sig}_1}$ realizes some ideal functionality $\mathcal{F}_1$ (the exact details of these protocols do not matter here). Suppose we have also shown that $\mathcal{P}_2 \,|\, \mathcal{F}_{\text{sig}_2} \leq^{SS} \mathcal{F}_2$, i.e., some other protocol $\mathcal{P}_2$ using ideal signatures is secure. By the composition theorem for a constant number of systems, we know that $\mathcal{P}_1 \,|\, \mathcal{F}_{\text{sig}_1} \,|\, \mathcal{P}_2 \,|\, \mathcal{F}_{\text{sig}_2} \leq^{SS} \mathcal{F}_1 \,|\, \mathcal{F}_2$. However, in this situation the protocols $\mathcal{P}_1$ and $\mathcal{P}_2$ use different subroutines $\mathcal{F}_{\text{sig}_1}$ and $\mathcal{F}_{\text{sig}_2}$, and hence, different public and private keys even if the same parties take part in both protocols. It might sometimes be desirable to use the same keys for each party across both protocols, i.e., share state between these two protocols.

This situation, which is not covered by the joint state theorem from Canetti and Rabin [14] (and similar ones from the literature), can easily be dealt with in the IITM model. More specifically, we obtain the following joint state realization as yet another straightforward application of the composition theorem for a constant number of systems: we can realize $\mathcal{F}_{\text{sig}_1} \,|\, \mathcal{F}_{\text{sig}_2}$ via a new protocol $\mathcal{P}_{\text{sig}}^{\text{js}} \,|\, \mathcal{F}_{\text{sig}}$ that uses a single instance of $\mathcal{F}_{\text{sig}}$ per party for all signing and verification operations, thus using the same keys for both $\mathcal{F}_{\text{sig}_1}$ and $\mathcal{F}_{\text{sig}_2}$. The protocol $\mathcal{P}_{\text{sig}}^{\text{js}}$ uses the same trick as $\mathcal{P}_{\text{pke}}^{\text{js}}$, however, instead of prefixing inputs with an SID it prefixes inputs with a protocol identifier (e.g., "1" or "2"). By the composition theorem (and transitivity of $\leq^{SS}$), we have that $\mathcal{P}_1 \,|\, \mathcal{P}_2 \,|\, \mathcal{P}_{\text{sig}}^{\text{js}} \,|\, \mathcal{F}_{\text{sig}} \leq^{SS} \mathcal{F}_1 \,|\, \mathcal{F}_2$, i.e., both protocols are secure even if they use the same keys (as long as they keep their respective message spaces disjoint by prefixing messages with a protocol ID).

10.3 Composition with Global State / Global Setup

Sometimes it is desirable to model a protocol that uses some kind of globally accessible resource such as a public-key infrastructure (PKI) or a common reference string (CRS). "Globally available" means that *every*

component, including arbitrary other protocols and the environment, have access to this resource. This concept is called *global state* or *global setup*. Just as for shared state, modeling global state is straightforward in the IITM model: a resource that should be globally available simply offers sufficiently many I/O tapes for the environment/other protocols to connect to. Composition theorems for protocols with global state then directly follow from the standard composition theorems, analogously to the composition theorems for joint state. In particular, as discussed in the following, the IITM model provides composition theorems for global state similar to those proposed by Canetti et al. [9] for the GUC model. We stress that these are merely some examples; the IITM model is able to express a large variety of types of global state, some of which have not even been considered in the literature so far, such as combinations of global state and joint state.

To enable global setup in the UC model, Canetti et al. [9] had to extend the UC model to a generalized UC (GUC) framework and define a new security notion, which Canetti et al. called *GUC-emulation/realizability*. The extension lets the environment invoke and interact directly with the global setup functionality (i.e., the globally available functionality); this is not possible in the UC model without extension because the environment can only interact directly with the highest-level protocol and the adversary. GUC-emulation is defined similarly to UC-emulation except that the environment is now allowed to access the global setup functionality. To formulate the composition theorem with global setup, Canetti et al. had to also introduce a notion called *EUC-emulation/realizability*, which is defined like GUC-emulation but restricts the environment to invoke only a single session of the protocol. The composition theorem with global setup that Canetti et al. prove is basically the following, where we use IITM-style notation to state this theorem (the actual IITM version is presented later):

Theorem 11 (global setup composition theorem in the GUC model [9]; informal)**.** *Let $\mathcal{P}$ be a protocol that uses a global setup functionality $\mathcal{G}$. If $\mathcal{P} \mid \mathcal{G}$ EUC-emulates $\mathcal{F} \mid \mathcal{G}$ for some ideal functionality $\mathcal{F}$, then $\mathcal{Q} \mid \mathcal{P} \mid \mathcal{G}$ GUC-emulates $\mathcal{Q} \mid \mathcal{F} \mid \mathcal{G}$, where $\mathcal{Q}$ is a protocol that uses (possibly multiple instances of) $\mathcal{P}$ (or $\mathcal{F}$) as a subprotocol.*

We note that, in the above theorem, $\mathcal{F}$ might or might not use $\mathcal{G}$ but it still has to be present on the right-hand side because the environment has to be able to interact with it.

As mentioned above, in the IITM model we can express the specific type of global setup used in GUC without extending the model. Not only is the IITM model sufficiently flexible and general for this purpose, composition with global setup is, similar to composition with joint state, essentially a mere special case of our standard composition theorems. More specifically, global setup (as used in GUC) can be expressed in the IITM model with a *global setup (ideal) functionality* $\mathcal{G}$ that can be any ideal functionality but it has a parametric number of I/O tape pairs, consisting of an input and output tape each. When we combine $\mathcal{G}$ with environmental, protocol, and adversarial systems (e.g., $\mathcal{E} \mid \mathcal{S} \mid \mathcal{F} \mid \mathcal{G}$ for an environment $\mathcal{E}$, a simulator $\mathcal{S}$, and an ideal functionality $\mathcal{F}$), then the parameter of $\mathcal{G}$ is chosen arbitrarily but sufficiently large such that every IITM in the protocol system can connect to one pair of I/O tapes of $\mathcal{G}$. The remaining I/O tape pairs (of which there are, as said, arbitrarily many) are connected to the environmental system which can then internally simulate arbitrary other protocols that access the same $\mathcal{G}$. Network tapes of $\mathcal{G}$ are connected, as usual, either to the simulator or the environment. In the following, we will keep the set of tapes $\mathcal{G}$ has implicit.

Now let $\mathcal{P}$ be a protocol using a global functionality $\mathcal{G}$. In a security proof, we have to show that $\mathcal{P} \mid \mathcal{G} \leq^{SS} \mathcal{F} \mid \mathcal{G}$ as usual, however, this should hold true no matter which specific I/O tape pairs of $\mathcal{G}$ are used by $\mathcal{P}$ and $\mathcal{F}$ to connect to this functionality (see also Figure 9). That is, the security proof holds true no matter which I/O tape names are used in $\mathcal{P}/\mathcal{F}$ for connecting to $\mathcal{G}$. Note that this is a natural property for global setup, which is supposed to provide the same globally available information to multiple protocols, independently of how those protocols connect to $\mathcal{G}$. We assume that the above is fulfilled when we write $\leq^{SS}$ in the following.

Within the IITM model, the composition theorems with global setup are now stated as follows. We start with the theorem for composing a constant number of protocols with shared global setup, which easily follows from Theorem 8 (general composition theorem for a constant number of protocols).

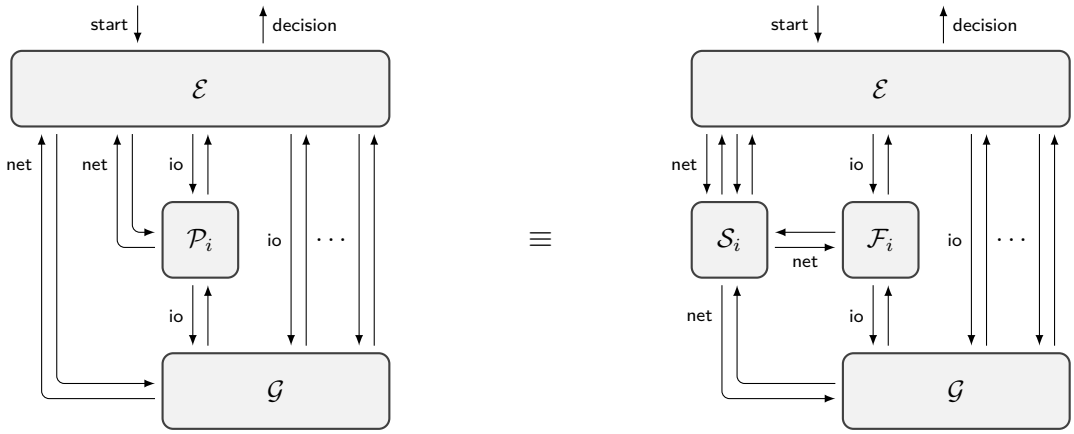

Figure 9: The systems used in the realization relation $\leq^{SS}$ when applied to protocol systems with global setup. Network tapes are labeled by net and I/O tapes are labeled by io. We require that $\mathcal{P}_i \,|\, \mathcal{G} \leq^{SS} \mathcal{F}_i \,|\, \mathcal{G}$, i.e., there exists a simulator $\mathcal{S}_i$ such that $\mathcal{E} \,|\, \mathcal{P}_i \,|\, \mathcal{G} \equiv \mathcal{E} \,|\, \mathcal{S}_i \,|\, \mathcal{F}_i \,|\, \mathcal{G}$ no matter which I/O tape pair $\mathcal{P}_i$ uses. Notice that $\mathcal{G}$ provides the set of tapes needed in the specific system it runs.

Corollary 4. *Let $k \in \mathbb{N}$ and let $\mathcal{P}_1, \ldots, \mathcal{P}_k$ be protocol systems that use a global setup functionality $\mathcal{G}$. Furthermore, let $\mathcal{Q}$ be an arbitrary protocol which might also access $\mathcal{G}$, with runtime bounds analogously to Theorem 8.*
If $\mathcal{P}_i \,|\, \mathcal{G} \leq^{SS} \mathcal{F}_i \,|\, \mathcal{G}$ (in the above sense, see also Figure 9) for some ideal functionalities $\mathcal{F}_i$ and $1 \leq i \leq k$, then:

$$\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k \,|\, \mathcal{G} \leq^{SS} \mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{F}_k \,|\, \mathcal{G}.$$

Proof. First, observe that because the realizations $\mathcal{P}_i \,|\, \mathcal{G}$ work independently of which specific I/O tape pairs they use to connect to $\mathcal{G}$, we can actually connect all protocols $\mathcal{P}_1, \ldots, \mathcal{P}_k$ to *the same* $\mathcal{G}$ using some arbitrarily chosen I/O tape pairs. For the specific I/O tape pairs that have been chosen for connections in the combined system $\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k \,|\, \mathcal{G}$, we then still have $\mathcal{P}_i \,|\, \mathcal{G} \leq^{SS} \mathcal{F}_i \,|\, \mathcal{G}$.

We can thus apply Theorem 8 k times in the following way: We start with the system $\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k \,|\, \mathcal{G}$. Since $\mathcal{P}_1 \,|\, \mathcal{G} \leq^{SS} \mathcal{F}_1 \,|\, \mathcal{G}$, we obtain from Theorem 8 that $\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k \,|\, \mathcal{G} \leq^{SS} \mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \mathcal{P}_2 \,|\, \ldots \,|\, \mathcal{P}_k \,|\, \mathcal{G}$. We can iterate this process $k-1$ more times to iteratively replace every $\mathcal{P}_i$ with $\mathcal{F}_i$ while keeping $\mathcal{G}$ in place. In the last step, we obtain $\mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{F}_{k-1} \,|\, \mathcal{P}_k \,|\, \mathcal{G} \leq^{SS} \mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{F}_k \,|\, \mathcal{G}$. Using transitivity of $\leq^{SS}$ we can combine all intermediate steps to obtain $\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k \,|\, \mathcal{G} \leq^{SS} \mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{F}_k \,|\, \mathcal{G}$, which concludes the proof. □

We emphasize that unlike in the UC model, this theorem (and the following ones) are stated in the IITM model without changing the model. Furthermore, this theorem is a mere corollary of Theorem 8; there is no need to prove any new composition theorems to handle composition of a constant number of protocols sharing the same global state. Similarly, a theorem for unbounded self-composition of protocol systems can be obtained as a mere special case of Theorem 10:

Corollary 5. *Let $\mathcal{P}$ be a protocol system that uses a global setup functionality $\mathcal{G}$ such that $\mathcal{P} \,|\, \mathcal{G}$ is a σ-session version, with runtime bounds defined analogously to Theorem 10. If $\mathcal{P} \,|\, \mathcal{G} \leq^{SS}_{\sigma\text{-single}} \mathcal{F} \,|\, \mathcal{G}$ for some ideal functionality $\mathcal{F}$, then $\mathcal{P} \,|\, \mathcal{G} \leq^{SS} \mathcal{F} \,|\, \mathcal{G}$.*

Proof. This is a special case of Theorem 10. □

This theorem requires $\mathcal{G}$ to be a σ-session version. This typically means that $\mathcal{G}$ consists of multiple independent instances where every session of $\mathcal{P}$ uses a different instance of $\mathcal{G}$. This covers many interesting cases: Consider, for example, a protocol $\mathcal{P}$ that uses one CRS per protocol session. In this case, every instance

of $\mathcal{G}$ models an independent CRS which is used by one session of $\mathcal{P}$. Corollary 5 implies that it is sufficient to analyze a single session of $\mathcal{P}$ using one CRS in isolation to obtain security for an unbounded number of sessions using an unbounded number of CRSs. Importantly, as $\mathcal{G}$ is global, other protocols can access and rely on the same instances of $\mathcal{G}$ and thus the same CRSs without impacting the security of $\mathcal{P}$. Another example is a protocol $\mathcal{P}$ that uses a random oracle, where the inputs to the random oracle are disjoint for different sessions of $\mathcal{P}$ (e.g., because all inputs are prefixed with a session identifier). In this case, $\mathcal{G}$ consists of multiple instances, each modeling an independent random oracle used by one session of $\mathcal{P}$. Again, Corollary 5 implies that it is sufficient to analyze a single session of $\mathcal{P}$ using one global random oracle to obtain security for an unbounded number of sessions using an unbounded number of random oracles. Afterwards, one can realize $\mathcal{G}$ (see also below) with a joint state realization where only a single global random oracle is used for all sessions (with inputs being prefixed with an SID). Thus, one obtains security of an unbounded number of sessions of $\mathcal{P}$ relying on a single random oracle that is globally accessible also for arbitrary other protocols.

Note that Corollary 5 is not precisely the same as the EUC/GUC composition theorem (cf. Theorem 11). The EUC/GUC composition theorem allows arbitrary sessions of $\mathcal{P}$ to access *the same* instance of $\mathcal{G}$, i.e., $\mathcal{G}$ is not necessarily a σ-session version. We can show an analogous composition theorem in the IITM model for global functionalities $\mathcal{G}$ that handle a request independently of the tape the request has been received on, except for sending a response on the corresponding output tape.[32] Again, observe that this property is usually satisfied for functionalities that are supposed to provide some globally available information. For such global functionalities $\mathcal{G}$, we can obtain the following theorem that precisely captures the EUC/GUC setting:

Theorem 12. *Let $\mathcal{P}$ be a protocol system that uses a global setup functionality $\mathcal{G}$ such that $\mathcal{P}$ is a σ-session version, but where multiple sessions of $\mathcal{P}$ can access the same instance of $\mathcal{G}$, with runtime bounds analogously to Theorem 10. If $\mathcal{P}\,|\,\mathcal{G} \leq^{SS}_{\sigma\text{-single}} \mathcal{F}\,|\,\mathcal{G}$ for some ideal functionality $\mathcal{F}$, then $\mathcal{P}\,|\,\mathcal{G} \leq^{SS} \mathcal{F}\,|\,\mathcal{G}$.*

Proof. The proof is analogous to the corresponding theorem without global setup (Theorem 10): one simply adds $\mathcal{G}$ to all systems considered. □

The above composition theorems focus on replacing functionalities $\mathcal{F}$ with realizations $\mathcal{P}$ where both rely on the same global functionality $\mathcal{G}$. However, sometimes it might be desirable to also replace a global functionality $\mathcal{G}$ with a different, indistinguishable protocol system $\mathcal{R}$. One example is given above, where multiple random oracles are realized by a single random oracle via a joint state realization. Another example is given in [15]. Yet another one could be that several CRSs are replaced by one random oracle. Due to the generality of the IITM model and its composition theorems, we also obtain a composition theorem for this purpose as a mere special case of Theorem 8:

Corollary 6. *Let $\mathcal{Q}$ be a protocol system using a global functionality $\mathcal{G}$ and let $\mathcal{R}$ be a protocol system such that $\mathcal{R} \leq^{SS} \mathcal{G}$, with runtime bounds analogously to Theorem 8. Then $\mathcal{Q}\,|\,\mathcal{R} \leq^{SS} \mathcal{Q}\,|\,\mathcal{G}$.*

Proof. This is a special case of Theorem 8. □

The original GUC model did not provide a similar composition theorem (in particular, this case is not covered by Theorem 11). Canetti et al. had to propose and prove yet another composition theorem in [15] to be able to also realize global functionalities in GUC. This is in stark contrast to the IITM model, where Corollary 6 is a simple application of the standard composition theorem.

We note that the above composition theorems for global state (Corollaries 4, 5, 6 and Theorem 12) can be combined to obtain corollaries analogously to those in Section 7.4.

10.4 A Concrete Example

We now present a concrete example of a protocol in order to illustrate the modeling sketched in Section 10.1. More specifically, we show how to model, according to the general description presented above, a simple key exchange protocol based on public-key encryption that is similar to the Needham-Schroeder Public-Key

[32] More precisely, upon receiving a request on a tape t, $\mathcal{G}$ runs some non-interactive algorithm that does not depend on t, obtains some output from that algorithm, and returns the output on the tape corresponding to t. No other actions are performed.

protocol [35] but that uses pre-established SIDs. We also explain how the security of this protocol can be proven in the IITM model along the lines sketched above using the (joint state) composition theorems (see Section 10.2) and an ideal functionality for public-key encryption, where security for this protocol means that it realizes a standard ideal key exchange functionality that we describe below. Finally, we illustrate how to build a secure channel from an ideal key exchange functionality (or its realization).

Notational conventions. To formally define the example protocol in the IITM model, we need some basic conventions. Each IITM is defined by five properties: *Tapes*, *State*, *CheckAddress*, *Initialization*, and *Compute*. The *Tapes* property describes all tapes of an IITM, the *State* property describes global variables which keep state across several activations (of the same instance), the *CheckAddress* property defines the CheckAddress mode in pseudo code, the (optional) *Initialization* property defines pseudo code which is executed exactly once when a fresh instance is activated for the first time in mode Compute, and the *Compute* property defines the behavior in mode Compute (potentially after the code from *Initialization* was executed).

We define the *Compute* property using several blocks of pseudo code. One block specifies how incoming messages, which might have to satisfy certain conditions, are processed, and which output is produced (if any). More specifically, a block starts with a receive (**recv**) command that specifies the message format, tape, and conditions that are expected by that block. When a new message is received, the IITM instance checks whether a pseudo code block accepts that message and executes the first one that does (later blocks that would also accept that message are ignored). Within each block, one can use the send (**send**) command to send a message on a tape and end the activation, or end the block without sending a message and thus activate the environment. One can also combine a send with a receive command within a block, by letting the receive command follow the send command immediately. In such a case, if the instance is activated with a new message, it continues where it left the computation after sending a message (if the new message is not accepted by the receive command, it is dropped and the instance waits for another message until one is accepted).

For better readability, in the pseudo code blocks we use sans-serif font to denote global variables, *italic* font to denote local variables, and `typewriter` font to denote fixed bit strings. We write $a \xleftarrow{\$} A$ to say that some value a is sampled uniformly at random from the set A.

In what follows, we first describe the example protocol informally. Before we formalize this protocol in the IITM model, we specify an ideal functionality for key exchange.

Description of our example protocol. Our example protocol is informally defined in Alice-Bob-Notation in Figure 10. There are two roles A and B with public keys k_A and k_B. We assume that the parties know the public keys of each other (i.e., we assume some form of a public-key infrastructure). We also assume that the two parties that play role A and B have already established a unique SID $sid = (sid', pid_A, pid_B)$ for a session of the protocol where pid_A and pid_B are the party names of the parties in role A and B, respectively. The SID sid is supposed to be unique in the sense that it is used in no other session.[33]

As shown in Figure 10, role A performs the following actions. First, A generates a nonce N_A, encrypts it together with the SID sid under B's public key k_B, and sends the obtained ciphertext to B. Then, A waits to receive a message that is encrypted under A's public key k_A and contains sid, A's nonce N_A, and some nonce N_B. Finally, A encrypts sid and the received nonce N_B under k_B and sends the obtained ciphertext to B. If something goes wrong in one of these steps, A aborts immediately, and otherwise, if the protocol runs through successfully, A outputs N_A as the session key.

Role B performs the following actions according to Figure 10. First, B waits to receive a message that is encrypted under k_B and contains sid and some nonce N_A. Then, B generates a nonce N_B, encrypts it together with sid and the received nonce N_A under k_A, and sends the obtained ciphertext to A. Finally, B waits to receive a message that is encrypted under k_B and contains sid and B's nonce N_B. If the protocol runs through successfully, B outputs N_A as the session key, and otherwise aborts immediately as soon as an error occurs.

[33] This SID can, for example, be established by exchanging nonces as explained in [1]. Both parties send nonces N'_A, N'_B in plain to each other. The SID then is the concatenation of the nonces and the party names: $sid = (N'_A, N'_B, pid_A, pid_B)$.

1. $A \rightarrow B$: $\{sid, N_A\}_{k_B}$
2. $B \rightarrow A$: $\{sid, N_A, N_B\}_{k_A}$
3. $A \rightarrow B$: $\{sid, N_B\}_{k_B}$

sid	–	unique pre-established SID that contains the party names of A and B
k_A, k_B	–	public key of A and B, respectively
N_A, N_B	–	nonces, generated by A and B, respectively
$\{m\}_k$	–	encryption of the message m with the public key k using an IND-CCA2 secure public-key encryption scheme

Figure 10: An example key exchange protocol based on public-key encryption.

The above protocol differs from the original Needham-Schroeder Public-Key protocol in two aspects: First, our protocol uses pre-established and unique SIDs. Second, since the party names are already included in the SID, we do not need to include them in the messages otherwise.

Ideal key exchange functionality. We use a standard ideal functionality $\mathcal{F}_{\text{ke}}$ for (authenticated) key exchange (see, e.g., [10, 13, 25, 28]). A formal definition of $\mathcal{F}_{\text{ke}}$ in the IITM model is given in Figure 11; in the following, we provide an informal description. This functionality describes one session of an ideal key exchange between two parties/roles. It has two sets of I/O tapes, one for each role. It first waits to receive a key exchange request from a role. The simulator (ideal adversary) is informed about such requests. If the simulator sends a message for one role to finish and both roles have sent their key exchange requests to $\mathcal{F}_{\text{ke}}$ before,[34] $\mathcal{F}_{\text{ke}}$ outputs the session key to that role, where the session key is chosen uniformly at random from $\{0,1\}^\eta$ by $\mathcal{F}_{\text{ke}}$ (η is the security parameter). (Of course, other distributions for the session key could be used.) The simulator has the ability to corrupt $\mathcal{F}_{\text{ke}}$, i.e., send a corrupt message to $\mathcal{F}_{\text{ke}}$, before any of the two roles have output a key. More precisely, in one request the simulator can corrupt one role. In this case, no matter which role was corrupted, if the simulator instructs a role to output a key, this key will be output by that role; the simulator can choose different keys for each of the two roles. In addition, $\mathcal{F}_{\text{ke}}$ forwards all messages from/to the corrupted party to/from the simulator. Altogether, an uncorrupted $\mathcal{F}_{\text{ke}}$ guarantees that the key a role receives after having sent a key exchange request to $\mathcal{F}_{\text{ke}}$ is a freshly generated key that is given only to the roles involved in the key exchange. The key is indistinguishable from random for an adversary even if the key is output by one role before the end of the protocol. Also, if both roles receive a key, the two keys are guaranteed to coincide. Conversely, a corrupted $\mathcal{F}_{\text{ke}}$, i.e., one or both roles are corrupted, does not provide security guarantees; the keys the roles obtain (if any) are determined by the simulator and they do not need to coincide. As usual, the environment may ask whether a role of $\mathcal{F}_{\text{ke}}$ has been corrupted; such a request is only allowed on the tapes of the respective role. Note that $\mathcal{F}_{\text{ke}}$ is environmentally bounded as in every activation it performs at most a polynomial number of steps in the current input and the security parameter.

As mentioned, $\mathcal{F}_{\text{ke}}$ captures only a single key exchange between two roles (played by arbitrary parties). Key exchange for an unbounded number of sessions and between an unbounded number of pairs of parties can be described by the multi-session version $!\underline{\mathcal{F}_{\text{ke}}}$ of $\mathcal{F}_{\text{ke}}$.[35] We define the domain of SIDs that $\underline{\mathcal{F}_{\text{ke}}}$ accepts to be SIDs of the form $sid = (sid', pid_A, pid_B)$.[36] Intuitively, an instance of $\underline{\mathcal{F}_{\text{ke}}}$ addressed by such an SID is an ideal functionality for the key exchange between the parties pid_A and pid_B in session sid'. (Note that by this, there can be multiple key exchange sessions between the same two parties.)

[34] The condition that both roles have sent their key exchange requests models *authenticated* key exchange and could be dropped, but is convenient.

[35] Note that an instance of $\mathcal{F}_{\text{ke}}$ does not need to be aware of its SID, and hence, it is not necessary to provide a multi-session and multi-party version $\mathcal{F}'_{\text{ke}}$ of $\mathcal{F}_{\text{ke}}$ as a σ_{prefix}-session version.

[36] See Section 5.2.1 for a discussion on domains of SIDs. Domains were also briefly mentioned in Section 2.

Tapes: from/to io_{role} ($role \in \{A, B\}$): $(\mathsf{io}^{\text{in}}_{role}, \mathsf{io}^{\text{out}}_{role})$; from/to net: $(\mathsf{net}^{\text{in}}, \mathsf{net}^{\text{out}})$ {*two pairs of I/O tapes for initiator A and responder B, one pair of network tapes to the adversary. See also Figure 16*

State:

- $\mathsf{state}_{\text{A}}, \mathsf{state}_{\text{B}} \in \{\bot, \texttt{started}, \texttt{finished}, \texttt{corrupt}\}$ {*Current status of initiator and responder; initially* $\bot$
- $\mathsf{k} \in \{0,1\}^\eta \cup \{\bot\}$ {*session key of an honest session; initially* $\bot$

CheckAddress: Accept every input on every tape.

Compute:

Process messages on I/O tapes:

recv `StartExchange` **from** io_{role} **s.t.** $role \in \{A, B\} \wedge \mathsf{state}_{role} = \bot$**:** {*Start the key exchange*
  $\mathsf{state}_{role} := \texttt{started}$
  send $(\texttt{StartExchange}, role)$ **to** net {*Notify adversary about the start of the key exchange*

recv `CorrStatus`? **from** io_{role} **s.t.** $role \in \{A, B\}$**:** {*Ask for corruption status*
  if $\mathsf{state}_{role} = \texttt{corrupt}$**:**
    send $(\texttt{CorrStatus}, \mathsf{true})$ **to** io_{role}
  else:
    send $(\texttt{CorrStatus}, \mathsf{false})$ **to** io_{role}

recv m **from** io_{role} **s.t.** $role \in \{A, B\} \wedge \mathsf{state}_{role} = \texttt{corrupt}$**:** {*Forwarding for corrupted roles*
  send $(\texttt{Forward}, role, m)$ **to** net

Process messages on the network tape:

recv $(\texttt{OutputKey}, role, k_{sim})$ **from** net **s.t.** $role \in \{A, B\} \wedge \mathsf{state}_{role} = \texttt{started}$**:** {*Output a session key. The key* k_{sim} *is ignored unless at least one role is corrupted, in which case the key* k_{sim} *is output.*
  $\mathsf{state}_{role} := \texttt{finished}$
  if $\mathsf{state}_{\text{A}} = \texttt{corrupt} \vee \mathsf{state}_{\text{B}} = \texttt{corrupt}$**:**
    send $(\texttt{SessionKey}, k_{sim})$ **to** io_{role}
  else if $\mathsf{state}_{\text{A}} \neq \bot \wedge \mathsf{state}_{\text{B}} \neq \bot$**:** {*If no role is corrupted: Ensure that both roles have actually started the key exchange*
    if $\mathsf{k} = \bot$**:**
      $\mathsf{k} \xleftarrow{\$} \{0,1\}^\eta$ {*Choose a fresh session key uniformly at random and store it*
    send $(\texttt{SessionKey}, \mathsf{k})$ **to** io_{role}

recv $(\texttt{Corrupt}, role)$ **from** net **s.t.** $role \in \{A, B\} \wedge \mathsf{state}_{\text{A}} \neq \texttt{finished} \wedge \mathsf{state}_{\text{B}} \neq \texttt{finished}$**:** {*Allow corruption*
  $\mathsf{state}_{role} := \texttt{corrupt}$
  send `Ok` **to** net

recv $(\texttt{Forward}, role, m)$ **from** net **s.t.** $role \in \{A, B\} \wedge \mathsf{state}_{role} = \texttt{corrupt}$**:** {*Forwarding for corrupted roles*
  send m **to** io_{role}

Figure 11: Formal definition of the ideal key exchange functionality $\mathcal{F}_{\text{ke}}$ in the IITM model. Note that the session version $\underline{\mathcal{F}_{\text{ke}}}$ of this functionality accepts SIDs of the form (sid', pid_I, pid_R) where pid_I and pid_R are the party IDs of the initiator and responder role, respectively, in that session. Thus, $\underline{\mathcal{F}_{\text{ke}}}$ guarantees that, if the session is uncorrupted, only those two parties in the same session will receive the session key. In particular, both parties agree on their session partners.

Modeling our example protocol in the IITM model. We now specify the example protocol as a protocol system $\mathcal{P} = !\,M_A \,|\, !\,M_B$ following the general approach outlined in Section 10.1. We define M_A and M_B to be IITMs that are addressed by structured SIDs of the form $sid = (sid', pid_A, pid_B)$. Every message received and output by M_A and M_B is prefixed by such an SID. In other words, M_A and M_B are σ_{prefix}-session versions with SIDs of the described form.

Ultimately, we are interested in the protocol system $\mathcal{P} \,|\, \mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\,\underline{\mathcal{P}_{\text{pke}}}$, where the protocol $\mathcal{P}$ uses (the joint state realization for) public-key encryption, and for this system we want to show that it realizes the ideal multi-session key exchange functionality $!\,\underline{\mathcal{F}_{\text{ke}}}$ (see above). The system $\mathcal{P} \,|\, \mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\,\underline{\mathcal{P}_{\text{pke}}}$ will precisely model our example protocol. Note that in a run of this system (in the context of some environment) there can be multiple sessions (with multiple parties) of our example protocol running concurrently and a party will use the same public/private key pair across all sessions, due to the joint state realization.

However, we do not want to show $\mathcal{P} \,|\, \mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\,\underline{\mathcal{P}_{\text{pke}}} \leq^{SS} !\,\underline{\mathcal{F}_{\text{ke}}}$ directly, but rather make full use of the composition and joint state theorems as described in Section 10.2. Therefore, we first replace $\mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\,\underline{\mathcal{P}_{\text{pke}}}$ by the ideal functionality $!\,\underline{\underline{\mathcal{F}_{\text{pke}}}}$ (the multi-session version of the multi-party version of $\mathcal{F}_{\text{pke}}$) and show that $\mathcal{P} \,|\, !\,\underline{\underline{\mathcal{F}_{\text{pke}}}} \leq^{SS} !\,\underline{\mathcal{F}_{\text{ke}}}$. This needs to be shown even only for a single session of $\mathcal{P}/\mathcal{F}_{\text{ke}}$, since by the composition theorem we can later replace $!\,\underline{\underline{\mathcal{F}_{\text{pke}}}}$ by its (joint state) realization $\mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\,\underline{\mathcal{P}_{\text{pke}}}$.

Now, let us describe $\mathcal{P}$ in more detail; a formal definition of M_A and M_B in the IITM model can be found in Figures 12 to 15. We define M_A and M_B such that they use $!\,\underline{\underline{\mathcal{F}_{\text{pke}}}}$, where every instance of $\underline{\underline{\mathcal{F}_{\text{pke}}}}$, denoted by $\underline{\underline{\mathcal{F}_{\text{pke}}}}[sid, pid]$, is addressed by an SID sid of the form (sid', pid_A, pid_B) as discussed above and a party name pid (see also Section 10.1). We assume that parties can ask $\mathcal{F}_{\text{pke}}$ for the public key of the party that owns $\mathcal{F}_{\text{pke}}$. More precisely, given an instance $\underline{\underline{\mathcal{F}_{\text{pke}}}}[sid, pid]$ of $\mathcal{F}_{\text{pke}}$, which is the "encryption/decryption-box" of party pid, we assume $\mathcal{F}_{\text{pke}}$ to be defined in such a way that parties can ask this instance to receive the public key of pid (in session sid). Clearly, the realization of $\mathcal{F}_{\text{pke}}$ would require a public-key infrastructure.

Both machines M_A and M_B have I/O tapes to connect to the environment and to connect to $!\,\underline{\underline{\mathcal{F}_{\text{pke}}}}$. Furthermore, they have network tapes to connect to the adversary. The system $\mathcal{P} \,|\, !\,\underline{\underline{\mathcal{F}_{\text{pke}}}}$ is depicted in Figure 16 and a run of this system is depicted in Figure 17 (w.r.t. an environment that created several sessions).

Let $sid = (sid', pid_A, pid_B)$ be an SID. Next, we describe the actions performed by $M_A[sid]$ and $M_B[sid]$, i.e., the instances of M_A and M_B with SID sid, in more detail. Since $\mathcal{P} \,|\, !\,\underline{\underline{\mathcal{F}_{\text{pke}}}}$ has to provide the same I/O interface as $!\,\underline{\mathcal{F}_{\text{ke}}}$, $M_A[sid]$ and $M_B[sid]$ first wait to receive key exchange requests. If, say, $M_A[sid]$ receives such a request, $M_A[sid]$ asks for the public key of the instance $\underline{\underline{\mathcal{F}_{\text{pke}}}}[sid, pid_B]$;[37] analogously for $M_B[sid]$. Both $M_A[sid]$ and $M_B[sid]$ also inform the adversary about the key exchange requests that they got. In the rest of the protocol, for encryption under k_A and decryption with the corresponding private key, the instance $\underline{\underline{\mathcal{F}_{\text{pke}}}}[sid, pid_A]$ is used by $M_A[sid]$ and $M_B[sid]$, where $M_B[sid]$ of course uses $\underline{\underline{\mathcal{F}_{\text{pke}}}}[sid, pid_A]$ only for encryption; analogously for k_B. Since the joint state realization prefixes all messages by the corresponding SID, we define $M_A[sid]$ and $M_B[sid]$ such that sid is not added to the plaintexts. For example, the first message that $M_A[sid]$ encrypts (using $\underline{\underline{\mathcal{F}_{\text{pke}}}}[sid, pid_B]$) is the nonce N_A rather than the concatenation of sid and N_A. The protocol messages (i.e., the ciphertexts) are sent and received via the network tapes (i.e., to/from the adversary who represents the network). If $M_A[sid]$ (and analogously for $M_B[sid]$) successfully finishes its run (according to the protocol specification), it outputs the session key (i.e., N_A) on the I/O tape to the environment. Note that the system $\mathcal{P} \,|\, !\,\underline{\underline{\mathcal{F}_{\text{pke}}}}$ is environmentally bounded as, similar to $\mathcal{F}_{\text{ke}}$, in every activation it performs at most a polynomial number of steps in the current input and security parameter.

We define static corruption of M_A as follows, following the description in Section 10.1; corruption of M_B is defined analogously. The adversary may corrupt $M_A[sid]$ at the beginning (when the adversary is informed about the key exchange request that $M_A[sid]$ got) by sending a corrupt message to this instance. If corrupted, $M_A[sid]$ forwards all input to the adversary and lets the adversary determine its output. Additionally, $M_A[sid]$

[37] If that instance is invoked for the first time, and hence, has not generated a key pair yet, it first generates a key pair.

Tapes: from/to io_A : $(\mathsf{io}_A^{\text{in}}, \mathsf{io}_A^{\text{out}})$; from/to $\mathsf{io}_A^{\text{pke}}$: $(\mathsf{io}_A^{\text{pke-in}}, \mathsf{io}_A^{\text{pke-out}})$; from/to net_A: $(\mathsf{net}_A^{\text{in}}, \mathsf{net}_A^{\text{out}})$ — *One pair of I/O tapes to connect to the environment/ higher-level protocols, one pair to connect to $\mathcal{F}_{\text{pke}}$, and one pair of network tapes to connect to the adversary. Also see Figure 16*

State:

- $\mathsf{sid} \in \{0,1\}^* \cup \{\bot\}$ — *{SID of this instance of M_A; initially $\bot$}*
- $\mathsf{pid}_A, \mathsf{pid}_B \in \{0,1\}^* \cup \{\bot\}$ — *{Party IDs of A and B; initially $\bot$}*
- $\mathsf{pk}_B \in \{0,1\}^\eta \cup \{\bot\}$ — *{Public encryption key of B; initially $\bot$}*
- $\mathsf{N}_A \in \{0,1\}^* \cup \{\bot\}$ — *{Nonce N_A from the protocol; initially $\bot$}*
- $\mathsf{finalMessage} \in \{0,1\}^* \cup \{\bot\}$ — *{Final protocol message; initially $\bot$}*
- $\mathsf{state}_\mathrm{A} \in \{\bot, \texttt{started}, \texttt{finished}, \texttt{corrupt}\}$ — *{Current status of initiator A; initially $\bot$}*

CheckAddress:

Try to parse an incomming message m as $m = ((sid', pid_A, pid_B), m')$ or, if it is received on the $\mathsf{io}_A^{\text{pke}}$ tape pair, as $m = (((sid', pid_A, pid_B), pid'), m')$. If this fails, reject the message.

if $\mathsf{sid} = \bot$**:**
Accept the message.
else:
Accept the message iff $\mathsf{sid} = (sid', pid_A, pid_B)$.

Initialization: Upon receiving the first message in mode Compute do:

Set $\mathsf{sid} := (sid', pid_A, pid_B)$, $\mathsf{pid}_A := pid_A$, $\mathsf{pid}_B := pid_B$ (where sid', pid_A, pid_B are the same as in mode CheckAddress).

Then, continue processing the first request as defined below.

Compute:

Process Messages on I/O tapes:

recv $(\mathsf{sid}, \texttt{StartExchange})$ **from** io_A **s.t.** $\mathsf{state}_\mathrm{A} = \bot$**:** — *{Start the key exchange*
  $\mathsf{state}_\mathrm{A} := \texttt{started}$
  send $((\mathsf{sid}, \mathsf{pid}_B), \texttt{PubKey?})$ **to** $\mathsf{io}_A^{\text{pke}}$ — *{Get public encryption key of B*
  recv $((\mathsf{sid}, \mathsf{pid}_B), \texttt{PubKey}, pk)$ **from** $\mathsf{io}_A^{\text{pke}}$
  $\mathsf{pk}_B := pk$
  send $(\mathsf{sid}, \texttt{StartExchange})$ **to** net_A — *{Notify adversary about the start of the key exchange. Wait for optional corruption by the adversary*
  recv (sid, m') **from** net_A
  if $m' = \texttt{Corrupt}$**:**
    $\mathsf{state}_\mathrm{A} := \texttt{corrupt}$ — *{Adversary corrupts this instance. Instance stops without sending a message*
  else:
    $\mathsf{N}_A \xleftarrow{\$} \{0,1\}^\eta$ — *{Instance is honest. Generate and send first message of the protocol*
    send $((\mathsf{sid}, \mathsf{pid}_B), \texttt{Enc}, \mathsf{pk}_B, \mathsf{N}_A)$ **to** $\mathsf{io}_A^{\text{pke}}$
    recv $((\mathsf{sid}, \mathsf{pid}_B), \texttt{Ciphertext}, y)$ **from** $\mathsf{io}_A^{\text{pke}}$
    send (sid, y) **to** net_A

recv $(\mathsf{sid}, \texttt{CorrStatus?})$ **from** io_A**:** — *{Ask for corruption status*[a]
  send $((\mathsf{sid}, \mathsf{pid}_A), \texttt{CorrStatus?})$ **to** $\mathsf{io}_A^{\text{pke}}$ — *{Check corruption status of subroutines*
  recv $((\mathsf{sid}, \mathsf{pid}_A), \texttt{CorrStatus}, corr_A)$ **from** $\mathsf{io}_A^{\text{pke}}$
  send $((\mathsf{sid}, \mathsf{pid}_B), \texttt{CorrStatus?})$ **to** $\mathsf{io}_A^{\text{pke}}$
  recv $((\mathsf{sid}, \mathsf{pid}_B), \texttt{CorrStatus}, corr_B)$ **from** $\mathsf{io}_A^{\text{pke}}$
  $corr = corr_A \vee corr_B \vee (\mathsf{state}_\mathrm{A} = \texttt{corrupt})$ — *{Consider instance corrupted if it was corrupted directly or any of the subroutines is corrupted*
  send $(\mathsf{sid}, \texttt{CorrStatus}, corr)$ **to** io_A

recv (sid, m') **from** io_A **s.t.** $\mathsf{state}_\mathrm{A} = \texttt{corrupt}$**:** — *{Forwarding of messages if instance is corrupted*
  send $(\mathsf{sid}, \texttt{Forward}, m')$ **to** net_A

[a] Note that this special request is always processed, even if the protocol instance is currently waiting to receive a response from, e.g., the subroutine $\mathcal{F}_{\text{pke}}$.

Figure 12: Formal definition of the machine M_A in the IITM model, part 1. See Figure 13 for part 2.

Compute (continued):

Process Messages on the network tape:

recv $(\mathsf{sid}, \texttt{ProtocolMessage}, m')$ **from** net_A **s.t.** $\mathsf{state}_\mathrm{A} = \texttt{started}$: {*Receive the second protocol message from B*

  send $((\mathsf{sid}, \mathsf{pid}_A), \texttt{Dec}, m')$ **to** $\mathsf{io}_A^{\mathrm{pke}}$ {*Decrypt and check whether second message fits expected format*

  recv $((\mathsf{sid}, \mathsf{pid}_A), \texttt{Plaintext}, x)$ **from** $\mathsf{io}_A^{\mathrm{pke}}$

  Try to parse x as $x = (\mathsf{N}_A, N_B)$. If not possible, stop this computation.

  $\mathsf{state}_\mathrm{A} := \texttt{finished}$ {*Second protocol message accepted. Generate third message and output key*

  send $((\mathsf{sid}, \mathsf{pid}_B), \texttt{Enc}, \mathsf{pk}_B, N_B)$ **to** $\mathsf{io}_A^{\mathrm{pke}}$

  recv $((\mathsf{sid}, \mathsf{pid}_B), \texttt{Ciphertext}, y)$ **from** $\mathsf{io}_A^{\mathrm{pke}}$

  $\mathsf{finalMessage} := y$

  send $(\mathsf{sid}, \texttt{SessionKey}, \mathsf{N}_A)$ **to** io_A

recv $(\mathsf{sid}, \texttt{GetFinalMessage})$ **from** net_A **s.t.** $\mathsf{state}_\mathrm{A} = \texttt{finished}$: {*Allow the adversary to retrieve the final*[a] *protocol message*

  send $(\mathsf{sid}, \mathsf{finalMessage})$ **to** net_A

recv $(\mathsf{sid}, \texttt{Forward}, m)$ **from** net_A **s.t.** $\mathsf{state}_\mathrm{A} = \texttt{corrupt}$: {*Forward messages to the environment if corrupted*

  send (sid, m) **to** io_A

recv $(\mathsf{sid}, \texttt{Dec}, y)$ **from** net_A **s.t.** $\mathsf{state}_\mathrm{A} = \texttt{corrupt}$: {*Allow adversary to decrypt messages for A*

  send $((\mathsf{sid}, \mathsf{pid}_A), \texttt{Dec}, y)$ **to** $\mathsf{io}_A^{\mathrm{pke}}$

  recv $((\mathsf{sid}, \mathsf{pid}_A), \texttt{Plaintext}, x)$ **from** $\mathsf{io}_A^{\mathrm{pke}}$

  send $(\mathsf{sid}, \texttt{Plaintext}, x)$ **to** net_A

[a] Recall that machines can output at most one message in each activation. However, M_A actually has to send two messages after accepting the second protocol message, namely, it has to output the session key to the environment and the third protocol message to the adversary on the network. We deal with this modeling issue by outputting only the key and allowing the adversary on the network to manually request the last protocol message, which is still a reasonable (albeit slightly artificial) modeling. We emphasize that this limitation of machines occurs in *all* universal composability models and is not specific to the IITM model.

Figure 13: Formal definition of the machine M_A in the IITM model, part 2. See Figure 12 for part 1.

allows the adversary to decrypt messages using $\underline{\underline{\mathcal{F}_{\mathrm{pke}}}}[sid, pid_A]$. (Encryption requests could also be allowed, but they do not make the adversary stronger, because he knows the public keys of all parties and can encrypt messages by himself.)

Just as in the case of $\mathcal{F}_{\mathrm{pke}}$, the environment may ask the instance $M_A[sid]$ (analogously for $M_B[sid]$) about its corruption status. It returns *yes* if and only if it was directly corrupted or $\mathcal{F}_{\mathrm{pke}}[sid, pid_A]$ is corrupted. (We note that the adversary has the ability to corrupt an instance of $\mathcal{F}_{\mathrm{pke}}$ without explicitly corrupting the respective instance of M_A or M_B, and hence, these instances will still follow the prescribed protocol. However, M_A and M_B report to the environment that they are corrupted if their instance of $\mathcal{F}_{\mathrm{pke}}$ that they use is corrupted, because the key exchange protocol cannot guarantee a secure key exchange in this situation.)

It is easy to see that the protocol system $\mathcal{P} \,|\, !\,\underline{\underline{\mathcal{F}_{\mathrm{pke}}}}$ is environmentally strictly bounded. Hence, all preconditions of the composition theorems about runtime properties are trivially satisfied. Furthermore, it is an σ_{prefix}-session version (because all messages are prefixed by an SID) and we can apply the reasoning as described in Section 10.1.

First, we state that the protocol as modeled above is secure in a single-session setting (see also Figure 16).

Claim 1. $\mathcal{P} \,|\, !\,\underline{\underline{\mathcal{F}_{\mathrm{pke}}}} \leq^{SS}_{\sigma_{\mathrm{prefix}}\text{-single}} !\,\underline{\mathcal{F}_{\mathrm{ke}}}$.

The proof of this claim, which we omit for the sake of brevity, is relatively simple because one needs to consider only a single session of the protocol and encryption is idealized by $\mathcal{F}_{\mathrm{pke}}$. In particular, the proof can be done by information-theoretic arguments alone, without reductions to the security of the encryption scheme. We note that the above claim even holds true in an information-theoretic setting with unbounded environments.

We further note that Claim 1 does not hold true if the nonce N_B (instead of N_A) is used as the session key. Canetti and Herzog [11] showed that the Needham-Schroeder-Lowe (NSL) protocol is insecure when N_B is used as the session key. The reason here is the same: When A has output the session key, N_B in this case,

Tapes: from/to io_B : $(\mathsf{io}_B^{\text{in}}, \mathsf{io}_B^{\text{out}})$; from/to $\mathsf{io}_B^{\text{pke}}$: $(\mathsf{io}_B^{\text{pke-in}}, \mathsf{io}_B^{\text{pke-out}})$; from/to net_B: $(\mathsf{net}_B^{\text{in}}, \mathsf{net}_B^{\text{out}})$

{*One pair of I/O tapes to connect to the environment/ higher-level protocols, one pair to connect to $\mathcal{F}_{\text{pke}}$, and one pair of network tapes to connect to the adversary. Also see Figure 16*

State:

- $\mathsf{sid} \in \{0,1\}^* \cup \{\bot\}$ {*SID of this instance of M_B; initially $\bot$*
- $\mathsf{pid}_A, \mathsf{pid}_B \in \{0,1\}^* \cup \{\bot\}$ {*Party IDs of A and B; initially $\bot$*
- $\mathsf{pk}_A \in \{0,1\}^\eta \cup \{\bot\}$ {*Public encryption key of A; initially $\bot$*
- $\mathsf{N}_A, \mathsf{N}_B \in \{0,1\}^* \cup \{\bot\}$ {*Nonces N_A and N_B from the protocol; initially $\bot$*
- $\mathsf{state}_\mathrm{B} \in \{\bot, \texttt{started}, \texttt{sentSecondMessage}, \texttt{finished}, \texttt{corrupt}\}$ {*Current status of responder B; initially $\bot$*

CheckAddress:

Try to parse an incomming message m as $m = ((sid', pid_A, pid_B), m')$ or, if it is received on the $\mathsf{io}_B^{\text{pke}}$ tape pair, as $m = (((sid', pid_A, pid_B), pid'), m')$. If this fails, reject the message.

if $\mathsf{sid} = \bot$**:**
Accept the message.

else:
Accept the message iff $\mathsf{sid} = (sid', pid_A, pid_B)$.

Initialization: Upon receiving the first message in mode Compute do:

Set $\mathsf{sid} := (sid', pid_A, pid_B)$, $\mathsf{pid}_A := pid_A$, $\mathsf{pid}_B := pid_B$ (where sid', pid_A, pid_B are the same as in mode CheckAddress).

Then, continue processing the first request as defined below.

Compute:

Process Messages on I/O tapes:

recv $(\mathsf{sid}, \texttt{StartExchange})$ **from** io_B **s.t.** $\mathsf{state}_\mathrm{B} = \bot$**:** {*Start the key exchange*
  $\mathsf{state}_\mathrm{B} := \texttt{started}$
  send $((\mathsf{sid}, \mathsf{pid}_A), \texttt{PubKey}?)$ **to** $\mathsf{io}_B^{\text{pke}}$ {*Get public encryption key of A*
  recv $((\mathsf{sid}, \mathsf{pid}_A), \texttt{PubKey}, pk)$ **from** $\mathsf{io}_B^{\text{pke}}$
  $\mathsf{pk}_A := pk$
  send $(\mathsf{sid}, \texttt{StartExchange})$ **to** net_B {*Notify adversary about the start of the key exchange. Wait for optional corruption by the adversary*
  recv (sid, m') **from** net_B
  if $m' = \texttt{Corrupt}$**:**
    $\mathsf{state}_\mathrm{B} := \texttt{corrupt}$ {*Adversary corrupts this instance. Instance stops without sending a message.*

recv $(\mathsf{sid}, \texttt{CorrStatus}?)$ **from** io_B**:** {*Ask for corruption status*[a]
  send $((\mathsf{sid}, \mathsf{pid}_A), \texttt{CorrStatus}?)$ **to** $\mathsf{io}_B^{\text{pke}}$ {*Check corruption status of subroutines*
  recv $((\mathsf{sid}, \mathsf{pid}_A), \texttt{CorrStatus}, corr_A)$ **from** $\mathsf{io}_B^{\text{pke}}$
  send $((\mathsf{sid}, \mathsf{pid}_B), \texttt{CorrStatus}?)$ **to** $\mathsf{io}_B^{\text{pke}}$
  recv $((\mathsf{sid}, \mathsf{pid}_B), \texttt{CorrStatus}, corr_B)$ **from** $\mathsf{io}_B^{\text{pke}}$
  $corr = corr_A \vee corr_B \vee (\mathsf{state}_\mathrm{B} = \texttt{corrupt})$ {*Consider instance corrupted if it was corrupted directly or any of the subroutines is corrupted*
  send $(\mathsf{sid}, \texttt{CorrStatus}, corr)$ **to** io_B

recv (sid, m') **from** io_B **s.t.** $\mathsf{state}_\mathrm{B} = \texttt{corrupt}$**:** {*Forwarding of messages if instance is corrupted*
  send $(\mathsf{sid}, \texttt{Forward}, m')$ **to** net_B

[a] Note that this special request is always processed, even if the protocol instance is currently waiting to receive a response from, e.g., the subroutine $\mathcal{F}_{\text{pke}}$.

Figure 14: Formal definition of the machine M_B in the IITM model, part 1. See Figure 15 for part 2.

Compute (continued):

Process Messages on the network tape:

recv $(\mathsf{sid}, \texttt{ProtocolMessage}, m')$ **from** net_B **s.t.** $\mathsf{state}_\mathrm{B} = \texttt{started}$: {*Receive the second protocol message from A*

  send $((\mathsf{sid}, \mathsf{pid}_B), \texttt{Dec}, m')$ **to** $\mathsf{io}_B^{\mathrm{pke}}$ {*Decrypt and check format of first message*

  recv $((\mathsf{sid}, \mathsf{pid}_B), \texttt{Plaintext}, x)$ **from** $\mathsf{io}_B^{\mathrm{pke}}$

  Try to parse x as $x = N_A$. If not possible, stop this computation.

  $\mathsf{state}_\mathrm{B} := \texttt{sentSecondMessage}$ {*First message was accepted, so generate and send second message*

  $\mathsf{N}_A := N_A$

  $\mathsf{N}_B \xleftarrow{\$} \{0,1\}^\eta$

  send $((\mathsf{sid}, \mathsf{pid}_A), \texttt{Enc}, \mathsf{pk}_B, (\mathsf{N}_A, \mathsf{N}_B))$ **to** $\mathsf{io}_B^{\mathrm{pke}}$

  recv $((\mathsf{sid}, \mathsf{pid}_A), \texttt{Ciphertext}, y)$ **from** $\mathsf{io}_B^{\mathrm{pke}}$

  send (sid, y) **to** net_B

recv $(\mathsf{sid}, \texttt{ProtocolMessage}, m')$ **from** net_B **s.t.** $\mathsf{state}_\mathrm{B} = \texttt{sentSecondMessage}$: {*Receive the third protocol message from A*

  send $((\mathsf{sid}, \mathsf{pid}_B), \texttt{Dec}, m')$ **to** $\mathsf{io}_B^{\mathrm{pke}}$ {*Decrypt and check whether third message fits expected format*

  recv $((\mathsf{sid}, \mathsf{pid}_B), \texttt{Plaintext}, x)$ **from** $\mathsf{io}_B^{\mathrm{pke}}$

  Try to parse x as $x = \mathsf{N}_B$. If not possible, stop this computation.

  $\mathsf{state}_\mathrm{B} := \texttt{finished}$ {*Third protocol message accepted. Output key*

  send $(\mathsf{sid}, \texttt{SessionKey}, \mathsf{N}_A)$ **to** io_B

recv $(\mathsf{sid}, \texttt{Forward}, m)$ **from** net_B **s.t.** $\mathsf{state}_\mathrm{B} = \texttt{corrupt}$: {*Forward messages to the environment if corrupted*

  send (sid, m) **to** io_B

recv $(\mathsf{sid}, \texttt{Dec}, y)$ **from** net_B **s.t.** $\mathsf{state}_\mathrm{B} = \texttt{corrupt}$: {*Allow adversary to decrypt messages for A*

  send $((\mathsf{sid}, \mathsf{pid}_B), \texttt{Dec}, y)$ **to** $\mathsf{io}_B^{\mathrm{pke}}$

  recv $((\mathsf{sid}, \mathsf{pid}_B), \texttt{Plaintext}, x)$ **from** $\mathsf{io}_B^{\mathrm{pke}}$

  send $(\mathsf{sid}, \texttt{Plaintext}, x)$ **to** net_B

Figure 15: Formal definition of the machine M_B in the IITM model, part 2. See Figure 14 for part 1.

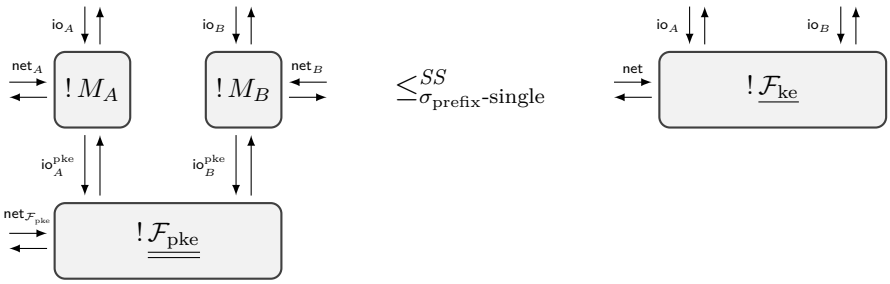

Figure 16: $\mathcal{P} \mid !\,\underline{\underline{\mathcal{F}_{\mathrm{pke}}}}$ single-session realizes $!\,\underline{\mathcal{F}_{\mathrm{ke}}}$ (see Claim 1). In this figure, horizontal arrows denote network tapes and vertical arrows denote I/O tapes. Every pair of input/output tapes is labeled by a name. The actual tape names are decorations of this name, e.g., $\mathcal{F}_{\mathrm{ke}}$ has the input tapes $\mathsf{io}_A^{\mathrm{in}}$, $\mathsf{io}_B^{\mathrm{in}}$, and $\mathsf{net}^{\mathrm{in}}$ and the output tapes $\mathsf{io}_A^{\mathrm{out}}$, $\mathsf{io}_B^{\mathrm{out}}$, and $\mathsf{net}^{\mathrm{out}}$.

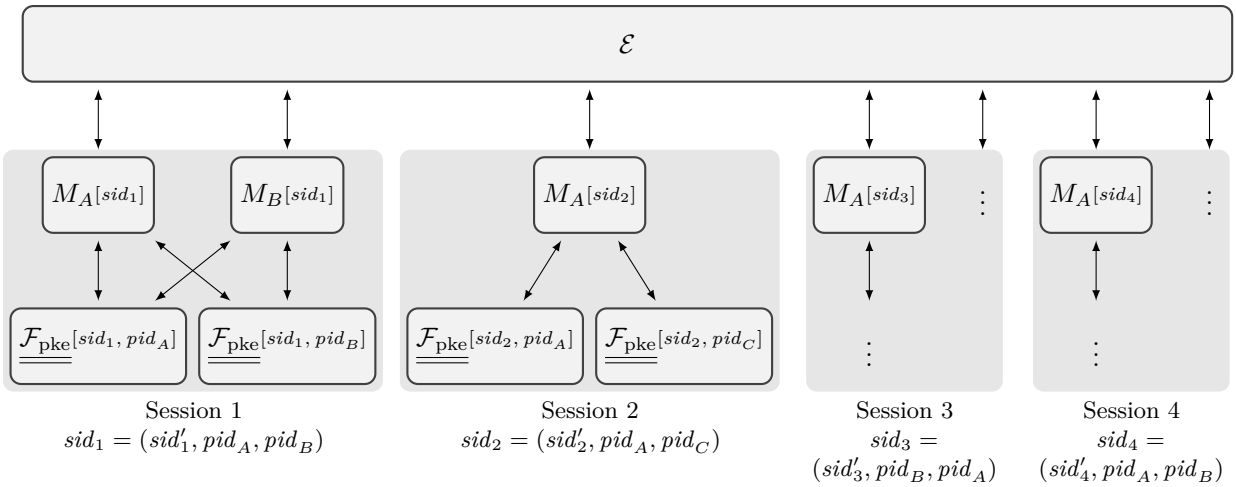

Figure 17: A run of $\mathcal{E} \,|\, \mathcal{P} \,|\, !\underline{\underline{\mathcal{F}_{\text{pke}}}}$ for some environment $\mathcal{E}$ that created four sessions: two sessions with party pid_A playing role A and pid_B playing role B (Sessions 1 and 4), one session with pid_A playing role A and pid_C playing role B (Session 2), and one session with pid_B playing role A and pid_A playing role B (Session 3). Every box denotes an instance of a machine that has been created in this run. For example, $M_A[sid_1]$ is the instance of M_A that is addressed by sid_1 (i.e., party pid_A in role A talking to pid_B in session sid'_1) and $!\underline{\underline{\mathcal{F}_{\text{pke}}}}[sid_1, pid_A]$ is the instance of $\mathcal{F}_{\text{pke}}$ that is addressed by sid_1 and pid_A (modeling the encryption/decryption-box of party pid_A in session sid_1). The arrows denote the connections between the instances via I/O tapes and addressing with SIDs. In addition, all instances connect to $\mathcal{E}$ via network tapes. These connections are not displayed.

but B has not yet received the last protocol message (a situation that an environment can easily create), then the environment can use B to test whether it is in the real world (i.e., interacting with $\mathcal{P} \,|\, !\underline{\underline{\mathcal{F}_{\text{pke}}}}$) or in the ideal world (i.e., interacting with $!\underline{\mathcal{F}_{\text{ke}}}$); see [11] for details. Interestingly, even if N_B is used as a session key, the protocol would still satisfy the key usability property, i.e., it would realize $!\underline{\mathcal{F}_{\text{keyuse}}}$, where $\mathcal{F}_{\text{keyuse}}$ is an ideal functionality for key usability introduced in [28] and extended in [25]. This functionality works just like $\mathcal{F}_{\text{ke}}$ but instead of outputting the session key it allows the party to use the session key in cryptographic operations, e.g., for (ideal) symmetric encryption. This functionality is often more useful than $\mathcal{F}_{\text{ke}}$ in applications (see below for an example), and being weaker than key indistinguishability, it can be realized by more protocols.

As described in Section 10.2, from Claim 1 we can directly deduce, by applying the composition theorem and the joint state theorem for $\mathcal{F}_{\text{pke}}$, that $\mathcal{P}$ is secure when multiple sessions run concurrently and a party uses the same public/private key pair across all sessions. Moreover, by the composition theorem, we can replace $\mathcal{F}_{\text{pke}}$ by its realization $\mathcal{P}_{\text{pke}}$ using an IND-CCA2 secure public-key encryption scheme (in conjunction with a public-key infrastructure), as illustrated in Figure 18.

Claim 2. $\mathcal{P} \,|\, \mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\underline{\mathcal{P}_{\text{pke}}} \leq^{SS} !\underline{\mathcal{F}_{\text{ke}}}$.

We note that the protocol system $\mathcal{P} \,|\, \mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\underline{\mathcal{P}_{\text{pke}}}$ precisely models our example protocol. A run of this system is depicted in Figure 19. As already mentioned, every party uses its public/private key pair across all protocol sessions. This is due to the use of the joint state realization. Furthermore, the joint state realization yields that the SID is added to every plaintext before encryption just as in the example protocol. Finally, the realization $\mathcal{P}_{\text{pke}}$ uses a standard IND-CCA2 secure public-key encryption scheme, no modifications are required, again just like in our example protocol.

Of course, $\mathcal{P} \,|\, \mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\underline{\mathcal{P}_{\text{pke}}}$ precisely models the example protocol only because the protocol uses pre-established SIDs in the way they are used by the joint state realization. As already mentioned in Section 10.1,

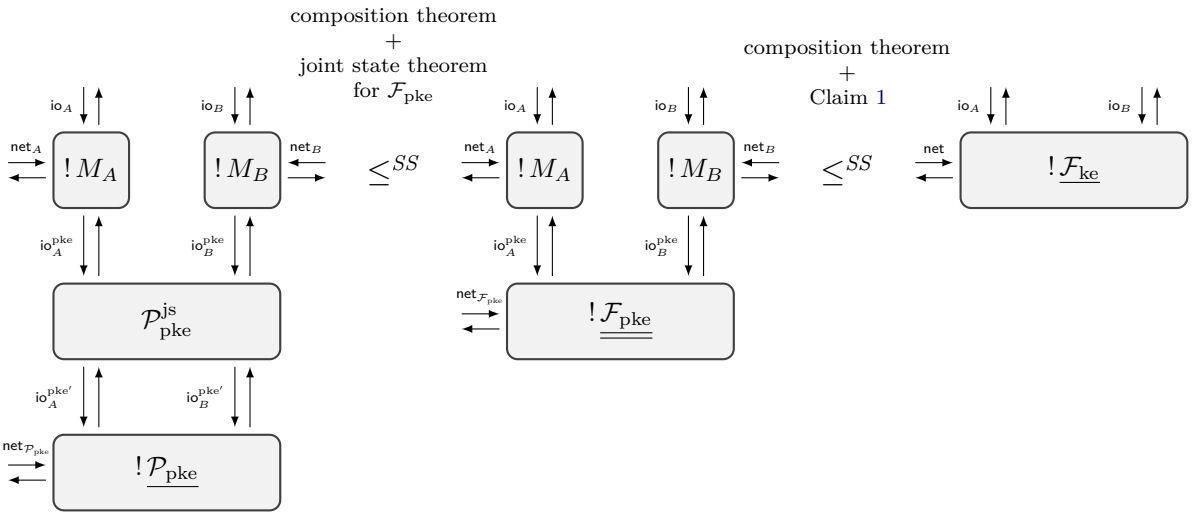

Figure 18: $\mathcal{P} \,|\, \mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\,\underline{\mathcal{P}_{\text{pke}}}$ realizes $!\,\underline{\mathcal{F}_{\text{ke}}}$ (see Claim 2). Tapes are labeled as in Figure 16.

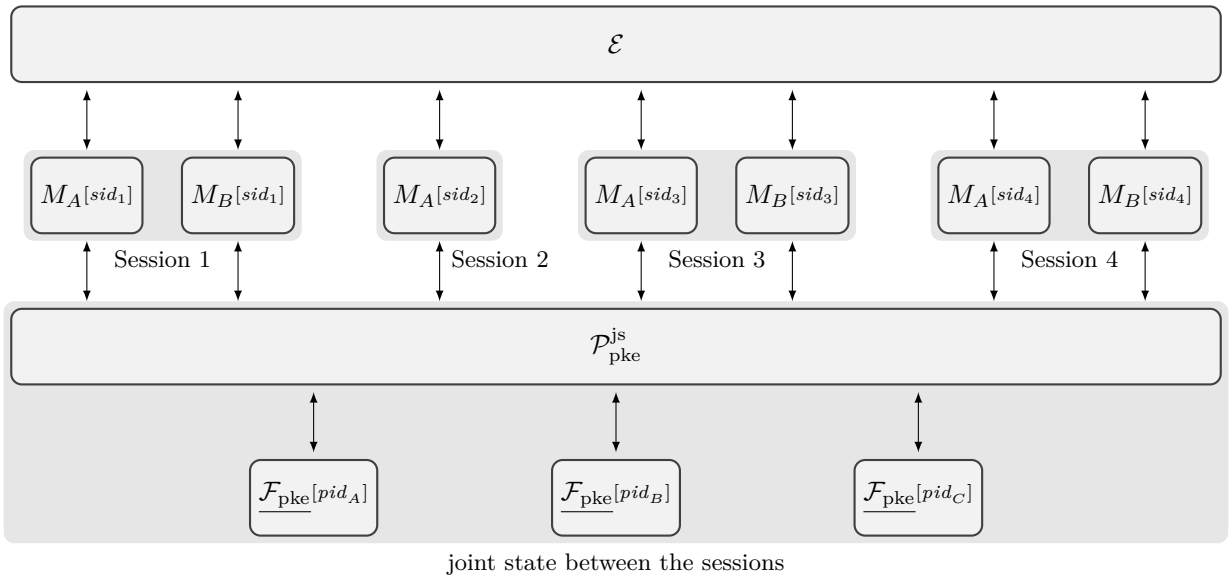

Figure 19: The run of $\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\,\underline{\mathcal{P}_{\text{pke}}}$ that corresponds to the run depicted in Figure 17. In this run, the joint state between the sessions consists of the instances of $\mathcal{F}_{\text{pke}}$. For example, the instance $\underline{\mathcal{F}_{\text{pke}}}[pid_A]$ (which models the encryption/decryption-box of party pid_A) is used by all four sessions.

one can also precisely model and analyze protocols in the IITM model even if they do not use pre-established SIDs (or not in the way stipulated by the joint state realization) [25, 28].

We further emphasize that static corruption is modeled in a strong sense: The adversary can corrupt public/private keys for every party individually (upon first use) and he can corrupt individual roles of sessions (upon session start). Motivation for this kind of fine-grained corruption was already given in Section 10.1. Clearly, this encompasses that the adversary can take complete control of a party by corrupting its public/private key and its roles in all sessions.

Building secure channels from key exchange protocols. To illustrate how more complex protocols can be built from the ideal key exchange functionality (or a realization thereof) using the composition theorems, we consider a secure channel protocol.

An ideal secure channel protocol $\mathcal{F}_{\text{sc}}$ (for two-parties/two-roles) works similarly to $\mathcal{F}_{\text{ke}}$ described above: It expects to receive a request to establish a secure channel from both roles and upon every such request informs the simulator about it, who can then decide to corrupt a role of $\mathcal{F}_{\text{sc}}$, at which point the simulator obtains full control of $\mathcal{F}_{\text{sc}}$. At some point, the simulator can send a "session established" message to $\mathcal{F}_{\text{sc}}$ for one of the two roles. If both roles have sent their requests to establish a secure channel to $\mathcal{F}_{\text{sc}}$ before (this models authentication, similarly to $\mathcal{F}_{\text{ke}}$), $\mathcal{F}_{\text{sc}}$ will inform the role that the session is established. From this point on, this role can send and receive messages to/from $\mathcal{F}_{\text{sc}}$. If a role sends a message via $\mathcal{F}_{\text{sc}}$, this message is delivered to the other role if the adversary instructs $\mathcal{F}_{\text{sc}}$ to deliver the message; the adversary is informed about every "message send" request and learns the length of the message that is to be delivered. The functionality $\mathcal{F}_{\text{sc}}$ guarantees confidentiality and integrity of the messages, that messages are not dropped, and delivered in order. As usual, the environment can ask about the corruption status of the roles in $\mathcal{F}_{\text{sc}}$ (on the respective tapes). For more details, see, e.g., [4, 28]. Just like $\mathcal{F}_{\text{ke}}$, $\mathcal{F}_{\text{sc}}$ captures only a single secure channel between two roles. Secure channels for an unbounded number of sessions and between an unbounded number of pairs of parties can be specified by the multi-session version $!\underline{\mathcal{F}_{\text{sc}}}$ of $\mathcal{F}_{\text{sc}}$ where we define the domain of SIDs that $\underline{\mathcal{F}_{\text{sc}}}$ accepts to be SIDs of the form $sid = (sid', pid_A, pid_B)$.

An example of a simple real secure channel protocol is the following: The roles first establish a session key (using a key exchange protocol). Now, messages are encrypted under the session key using an authenticated symmetric encryption scheme, i.e., an IND-CPA and INT-CTXT secure scheme, along with a counter that is added to every message (in order to prevent messages to be dropped or reordered).

Formally, let $\mathcal{P}_{\text{ke}}$ be a secure (multi-session) key exchange protocol, i.e., $\mathcal{P}_{\text{ke}} \leq^{SS} !\underline{\mathcal{F}_{\text{ke}}}$. For example, $\mathcal{P}_{\text{ke}}$ could be the example protocol $\mathcal{P} \,|\, \mathcal{P}^{\text{js}}_{\text{pke}} \,|\, !\underline{\mathcal{P}_{\text{pke}}}$ discussed above. Furthermore, let $\mathcal{P}_{\text{sc}} = !\, M^{\text{sc}}_A \,|\, !\, M^{\text{sc}}_B$ be a secure channel protocol that uses the key exchange protocol $\mathcal{P}_{\text{ke}}$. The protocol system $\mathcal{P}_{\text{sc}}$ is defined similarly to the key exchange protocol $\mathcal{P}$ above except that it does not use $\mathcal{F}_{\text{pke}}$ but $\mathcal{P}_{\text{ke}}$: Instances of M^{sc}_A and M^{sc}_B are again addressed by SIDs of the form (sid', pid_A, pid_B). They first, at the I/O interface, wait to be instructed to establish a secure channel; the adversary is informed about such a request and can decide to corrupt that instance (see below). The instance then, if not corrupted, sends a key exchange request to $\mathcal{P}_{\text{ke}}$ using its SID (sid', pid_A, pid_B). When it receives a session key, it outputs to the party that the secure channel is established. The instance is now ready to receive and send messages. To send messages it uses a counter and an authenticated symmetric encryption scheme as sketched above.

As mentioned, an instance M^{sc}_A (or M^{sc}_B) can be corrupted by the adversary when the adversary is informed about the request to establish a secure channel by M^{sc}_A. When corrupted, M^{sc}_A provides full control to the adversary, including the interface of M^{sc}_A to $\mathcal{P}_{\text{ke}}$. (We do not require $\mathcal{P}_{\text{ke}}$ to be corrupted as well.) If an instance is asked whether it is corrupted by the environment, it returns *yes* iff it was explicitly corrupted by the adversary or the corresponding role in the instance of $\mathcal{P}_{\text{ke}}$ it interacts with was corrupted. We note that, by this, (static) corruption is again modeled in a quite general and fine-grained way.

Now, we want to show security of the protocol system $\mathcal{P}_{\text{sc}} \,|\, \mathcal{P}_{\text{ke}}$, i.e., the composition of the secure channel protocol $\mathcal{P}_{\text{sc}}$ and the key exchange protocol $\mathcal{P}_{\text{ke}}$. More precisely, we want to show that $\mathcal{P}_{\text{sc}} \,|\, \mathcal{P}_{\text{ke}}$ realizes $!\underline{\mathcal{F}_{\text{sc}}}$, i.e., $\mathcal{P}_{\text{sc}} \,|\, \mathcal{P}_{\text{ke}} \leq^{SS} !\underline{\mathcal{F}_{\text{sc}}}$. For this purpose, it suffices to show that:

$$\mathcal{P}_{\text{sc}} \,|\, !\underline{\mathcal{F}_{\text{ke}}} \leq^{SS}_{\sigma_{\text{prefix-single}}} !\underline{\mathcal{F}_{\text{sc}}} \ . \tag{27}$$

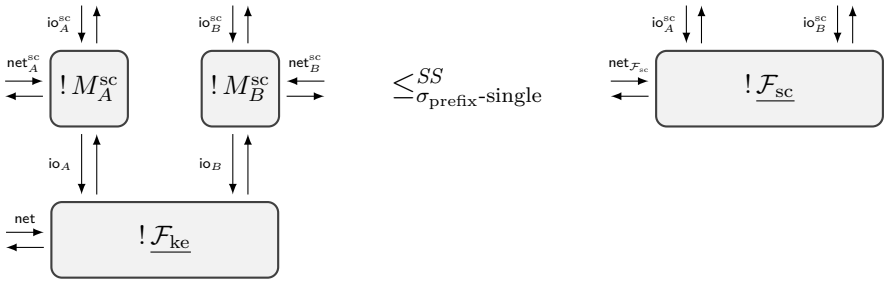

Figure 20: $\mathcal{P}_{\mathrm{sc}} \,|\, !\,\underline{\mathcal{F}_{\mathrm{ke}}}$ single-session realizes $!\,\underline{\mathcal{F}_{\mathrm{sc}}}$ (see Equation (27)). Tapes are labeled as in Figure 16.

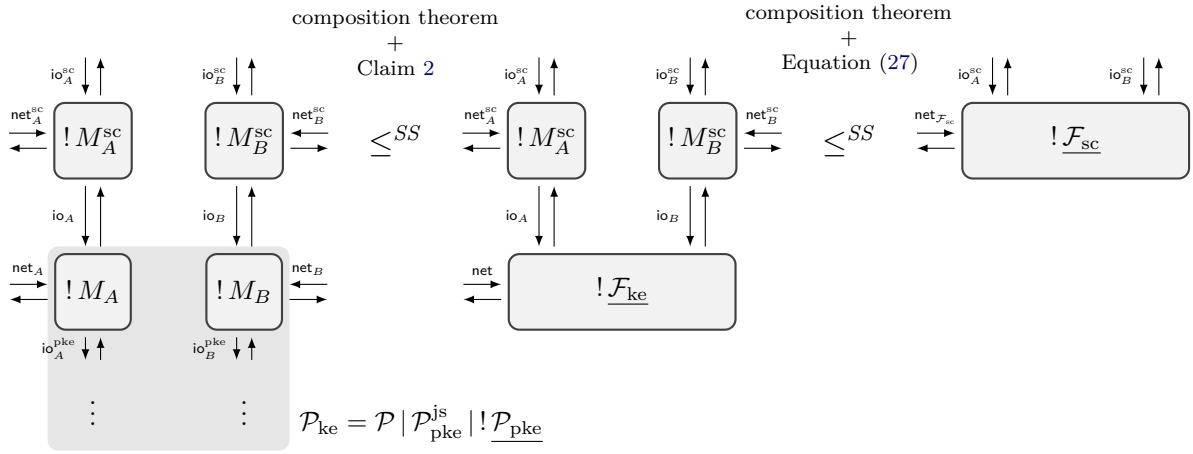

Figure 21: $\mathcal{P}_{\mathrm{sc}} \,|\, \mathcal{P}_{\mathrm{ke}}$ realizes $!\,\underline{\mathcal{F}_{\mathrm{sc}}}$ (see Equation (28)). Tapes are labeled as in Figure 16. There can be multiple instances of the machines M_A^{sc} and M_B^{sc} as well as of the machines M_A and M_B.

This statement is depicted in Figure 20. From (27), $\mathcal{P}_{\mathrm{ke}} \leq^{SS} !\,\underline{\mathcal{F}_{\mathrm{ke}}}$, and by Theorem 8 and 10, we immediately obtain the following (see also Figure 21):

$$\mathcal{P}_{\mathrm{sc}} \,|\, \mathcal{P}_{\mathrm{ke}} \leq^{SS} !\,\underline{\mathcal{F}_{\mathrm{sc}}} \ . \tag{28}$$

This statement says that multiple session of the secure channel protocol realize multiple sessions of the ideal secure channel functionality.

Note that the proof of (27) is relatively simple because it involves reasoning about only a single session of the protocol and because the key exchange is idealized. But it still requires a reduction proof to the security of the authenticated encryption scheme. This can be avoided if we do not use $\mathcal{F}_{\mathrm{ke}}$, which outputs the session key, but instead an ideal key usability functionality $\mathcal{F}_{\mathrm{keyuse}}$ (as already mentioned above), which does not output the session key but instead allows $\mathcal{P}_{\mathrm{sc}}$ to use the session key in an ideal way (similar to encryption and decryption with $\mathcal{F}_{\mathrm{pke}}$). Such a key usability functionality has been proposed in [28] and extended in [25].

10.5 Another Instantiation: the SUC Model

Above we presented one possible set of conventions for modeling real and ideal protocols. Of course other instantiations are conceivable as well. In this subsection, we briefly sketch an alternative instantiation motivated by the SUC model [8].

The SUC model was proposed by Canetti et al. with the goal to adjust and simplify the UC model for secure multi-party computation (MPC) by removing features that are not needed (in this specific case) and

fixing several conventions for the protocol structure that are closer to what protocol designers for MPC are used to.

More precisely, some of the changes compared to the UC model are the following: Most notably, SUC runs with a fixed set of machines, representing one party each; new (instances of) machines may not be created during a run. This change also allows for simplifying the original runtime notion of the UC model, namely, machines in SUC are required to run in polynomial time in the length of their input from the environment plus security parameter. Furthermore, SUC defines a protocol structure that is very close to what protocol designers are used to from classic game based security definitions for MPC. In particular, machines in SUC communicate with each other only via authenticated channels that are controlled by the adversary; no direct connections between machines are allowed.

Because SUC changes some fundamental aspects of the UC model, such as the ability to invoke an arbitrary number of machines and the runtime definition, it is not a mere instantiation of the UC model but rather a different model; thus, the composition theorems have to be re-stated and re-proven for SUC. The work of [8] does so by providing a transformation of a SUC protocol to a UC protocol such that SUC security implies UC security of the transformed protocol and vice versa. However, this transformation is quite cumbersome and yields unnatural protocol specifications that rely on artificial padding of inputs to work. The proof of this transformation is quite involved, taking up several pages just for relating the runtime notions of SUC and UC.

In contrast, we can easily provide a set of conventions to express the SUC model in the IITM model without having to change the model itself: To fix the number of protocol participants, we can use a fixed number of machines that are not in the scope of a bang (alternatively, we can also bound the number of instances of a single machine via the CheckAddress mode). The runtime requirements in the IITM model are even simpler and more general than in SUC, as we do not require every single machine to be polynomial, but only require the protocol as a whole to be polynomial. To model authenticated channels, one can use an appropriate subroutine that all protocol participants connect to and which is controlled by the adversary on the network. Using this set of conventions, we directly obtain the following corollary of Theorem 8 (composition theorem).

Corollary 7. *Let $\mathcal{P}$ be a protocol system modeled via the above SUC conventions for the IITM model, and let $\mathcal{Q}$ be any other protocol system. If $\mathcal{P} \leq^{SS} \mathcal{F}$ for some ideal functionality $\mathcal{F}$, then $\mathcal{Q} \mid \mathcal{P} \leq^{SS} \mathcal{Q} \mid \mathcal{F}$.*

Unlike [8], we do not have to re-prove the composition theorem but can use the SUC conventions for the IITM model straightaway. We also note that, as we do not have to change the underlying model, all security results obtained by using SUC conventions can seamlessly be combined with other results via the same general composition theorems of the IITM model. For example, SUC protocols could be used as subprotocols of more general protocols.

11 Related Work

As already mentioned in the introduction, the IITM model proposed here coincides with the (old) IITM model proposed in [23], except that we now use more general runtime notions. In particular, we introduce the notion of environmentally bounded systems, which conceptually builds on the notion of reactive polynomial time proposed by Hofheinz et al. in [21], but which we adjust to IITMs.

We concentrate our comparison of the IITM model with other models on Canetti's UC model [4] and the GNUC model [19] by Hofheinz and Shoup. Further models, including [7, 24, 36], are not considered as they either fix the runtime of a machine by a fixed polynomial in the security parameter, which has many problems as discussed in [18, 23], or do not consider systems with a polynomial number of instances of machines, which, however, is needed to even state general composition theorems. A recent cryptographic model by Maurer and Renner [33] (see also [32]) does not define a full model on the machine level, and hence, cannot directly be compared with the IITM model.

Before we discuss each model in detail, we provide a brief overview of the history of these models. The original UC model was proposed by Canetti in his seminal paper in 2001 [5]. In this model, the

runtime of machines was bounded by a constant polynomial in the security parameter, which caused several problems [18, 23]. Motivated by this, Canetti proposed a major revision of the UC model in 2005 [4] which includes a new runtime definition. However, even the 2005 version of the UC model has several issues; to fix these issues and create a formally sound universal composability framework, both the original IITM model in 2006 [23] and the GNUC model in 2011 [19] were proposed. This paper, which dates back to January 2013 [31], extends, as mentioned before, the original IITM model with a more general runtime definition. Later in July 2013, Canetti proposed the second major revision of the UC model [4] to fix issues of the 2005 version.

In the following, we will first discuss the 2005 version of the UC model, as that was the most recent one when this paper was written. We then discuss the 2013 version and point at some severe problems present in that version.[38] We then discuss the GNUC model.

11.1 UC Model (2005)

In this section, we discuss the 2005 version of the UC model [4] which was the most up to date one when this paper was written. In-depth discussions of this model can also be found in [19, 21, 23, 26]. We refer to Section 11.2 for the 2013 version of UC.

The UC model has several severe technical flaws, for example, concerning the validity of the composition and joint state theorems. While, formally speaking, the gaps in the model often invalidate cryptographic proofs carried out in the UC model in the literature, conceptually on a higher level of abstraction the cryptographic results and proofs might still be valid. However, it is clearly highly unsatisfying if statements and proofs do not rest on solid ground.

In [19], a concrete example is given showing that the composition theorem does not hold true in the UC model. The main reason is that interactive machines do not necessarily know who they are interacting with. In the composition theorem, this leads to the problem that a simulator might not know what kind of simulation needs to be performed as he is unaware of the protocol structure and the code of the instance he is interacting with. As a result, one can construct, (natural) protocols $\mathcal{Q}$, $\mathcal{F}$, and $\mathcal{P}$ such that $\mathcal{P}$ realizes $\mathcal{F}$ but the composition of $\mathcal{Q}$ and $\mathcal{P}$ does not realize the composition of $\mathcal{Q}$ and $\mathcal{F}$, in contradiction to what the composition theorem says. This problem is not present in the IITM model since the protocol structure is clearly defined by the connections via tapes, and every protocol machine has unique network tapes. Thus, the simulator is aware of the structure and code of the protocol and is able to uniquely identify machines (including their code) via the network tape that messages are sent/received on.

Another reason why the composition theorem is problematic is the fact that there does not exist a so-called dummy adversary in the UC model, i.e., an adversary which forwards all messages back and forth between the environment and the protocol. This in turn is due to the way the runtime of machines is defined in the UC model. In the UC model the *total* runtime of the ITMs is bounded by a polynomial in the security parameter and the length of the input received on I/O tapes (minus the runtime provided to "subroutine ITMs"); input received on communication tapes (the network interface) does not increase the runtime resources available to a machine. Now, in the UC model the adversary has I/O tapes to the environment but only network tapes to the protocol. Consequently, such an adversary cannot forward arbitrarily many and arbitrarily long messages from the protocol to the environment since it might not have the runtime resources to do that. In the UC model, the dummy adversary has to ask for resources from the environment in order to be able to forward arbitrary messages from the protocol to the environment. With a full-fledged dummy adversary, i.e., one that can forward all messages without asking for resources, one can prove that a real protocol realizes an ideal one by simply constructing a simulator for the dummy adversary (instead of constructing simulators for all real adversaries). However, as shown in [21], the restricted dummy adversary available in the UC model—the one that has to ask the environment for resources—is incomplete in this respect: considering only this adversary does not guarantee security for all adversaries. Now, the composition theorem in the UC model has only been proven with respect to the (restricted) dummy adversary. But since in the UC

[38] We note that at the end of 2018, while this paper was still under review, Canetti published a new eprint version of his UC model, which, however, is out of the scope of the discussion in this work.

model dummy adversaries are incomplete in the sense explained above the proof of the composition theorem is incomplete—in addition to the fact that the composition theorem does not hold true due to the above mentioned orthogonal problem. In the IITM model, an unrestricted dummy adversary is obviously available as shown in Lemma 5. Completeness of the dummy adversary follows from Theorem 7 (see also Remark 12).

It was shown in [26] that the general joint state theorem (such theorems have been discussed in Section 10.2) does not hold true in the UC model either; Canetti and Rabin proposed this theorem in [14]. This is mainly due to the way the runtime of ITMs is defined in the UC model (see above). As a consequence of this definition, by sending many messages to an ITM (on the network interface), the ITM can be forced to stop. Also, in general, a single ITM M cannot simulate a concurrent composition of a fixed finite number of ITMs or an unbounded number of (copies of) ITMs: Assume that the ITM M internally simulates (a fixed number of) other machines. Now, by sending many messages to M intended for some internally simulated machine, M will eventually stop, and hence, cannot simulate the other machines internally anymore, even though, in the actual composition these machines could still perform actions. Due to the way the runtime of machines and systems of machines is defined in the IITM model these problems do not occur. In particular, in the IITM model one machine can simulate any system of machines in any polynomial context (see Lemmas 6 and 7) and an IITM cannot be exhausted, it can perform computations in every activation. In particular, it can always read its input in both modes, CheckAddress and Compute.

Besides the technical flaws of the UC model sketched above, the UC model also hard-wires many details into the basic model, including:

- a specific runtime notion that depends on how much data a machine has sent to its subroutines
- an addressing mechanism based on PIDs and SIDs
- a protocol structure with disjoint session state
- a fixed corruption mechanism.

This is in contrast to the IITM model which does not fix these aspects on the level of the model, thus providing greater flexibility: The runtime notions employed in the IITM model are quite intuitive and straightforward, capturing the UC runtime notion for protocols as a special case. The CheckAddress mode allows for a generic and very flexible addressing mechanism where we do not have to talk about SIDs and PIDs. This is important in order to model protocols in a faithful way [28], and it allows for covering a large class of protocols via a single composition theorem (as illustrated in Section 10). The way corruption is handled is entirely left to the specification of the protocols and functionalities, and hence, is also very flexible (as illustrated in Section 10).

As pointed out in [19], there are in fact two problems with the way corruption is handled in the UC model. First, in order for the adversary to be able to corrupt a party, the adversary needs to receive an authorization message from the environment. Such a message contains a machine name which the adversary is allowed to corrupt. However, there are different sets of machines in the real and the ideal world, and hence, it is unclear what such a message means in the different worlds. Second, in the UC model an adversary can create machines and determine their program and IDs. By this he can "impersonate" or "hijack" honest machines without actually corrupting them. But then it is, in most cases, impossible for the simulator to achieve its goals because it is not allowed to corrupt the ideal functionality. So, as already explained in [19], formally most security claims in the literature are simply false.

Compared to the UC model, the flexibility of the IITM model allows for supporting the seamless modeling of a wide range of protocols and their composition:

- In order to deal with joint state or global setup the UC model needed to be extended [9,14]. This is not necessary in the IITM model as discussed in Section 10.2 and 10.3, respectively. The joint state theorem and the main global setup theorems are even direct consequences of the composition theorem in the IITM model. Since the UC model fixes things like the addressing of machines by SIDs and PIDs as well as the way machines can be corrupted, the composition and joint state theorems are less general. For example, if the way machines can be corrupted is changed, these theorems would have to be reproven.

- The composition theorem of the UC model requires protocols to have disjoint sessions (or, in the case of the joint state extension, protocols are required to behave as if they have disjoint sessions). This is because the composition theorem of the UC model combines unbounded self-composition of the same protocol with parallel composition of a fixed number of (different) protocols. The IITM model splits those composition types into two theorems that can be applied separately. As a consequence, the IITM model also directly supports protocols that share state between sessions in arbitrary ways. This includes in particular protocols that are not joint state realizations. Such protocols can then be composed with arbitrary other protocols (with or without disjoint sessions) via our theorem for parallel composition of (different) protocols.

- The way the runtime is defined in the UC model makes it hard to formulate protocols and functionalities. In fact, many, maybe most protocols/functionalities found in the literature cannot be expressed in the UC model. For example, it is very common that a protocol machine/functionality drops messages received from a communication tape (the network interface) if they do not have the correct format. However, as explained before, in the UC model machines can easily be "exhausted" by sending them many useless messages on the communication tapes and there is nothing a machine can do about it. Thus, a machine will not have sufficient resources to even read incoming messages at some point and it will be forced to stop, which typically is not the intended behavior of functionalities/protocols. Dealing with this issue in a formally correct way in the UC model is, if at all possible, very tricky and has, as far as we know, never been done in the literature. Note that in the IITM model one can easily formulate protocols and functionalities within the class of environmentally bounded protocol systems that simply reject messages that have the wrong format.

The UC and IITM models differ also in how they model connections between (instances of) protocol machines. The IITM model introduces input and output tapes that are used to connect two different machines. An instance of one machine can then write a message on an output tape, which will be delivered to an instance of the machine with the corresponding input tape (as explained, which exact instance of the latter machine receives the message is determined by running the instances of that machine in CheckAddress mode), allowing an instance of one machine to send messages to an instance of the other machine. In contrast, the UC model does not have such "dedicated connections" between machines but instead allows protocol machines to specify the machine code of the receiver when sending a message. The message will then be delivered to an instance running that machine code. Both mechanisms of the IITM and UC models serve the same purpose, namely, allowing protocols to send messages to (instances of) a specific machine. However, the two approaches result in technical differences. For example, in the UC model the proof of the composition theorem involves rewriting and sandboxing the code of higher-level protocols to replace their subroutines. This is not necessary in the IITM model since tapes can be connected to different subroutines without modifying higher-level protocols.

11.2 UC Model (2013)

The 2013 version of the UC model [4] adds several modifications to address some of the problems mentioned in Section 11.1. These modifications include:

- Instances of protocol machines are now required to send a special setup notification to the adversary upon first activation. This notification contains the code of the machine and some information about the protocol structure, i.e., the parent of the instance. This solves the problem that the adversary does not know who he is talking to. As mentioned in Section 11.1, this invalidated the composition theorem in the 2005 version; as mentioned above, the IITM model does not need such a requirement as the adversary can identify the machine (code) of the sender of a message via the tape the message was received on.

- The class of environments is restricted to *balanced environments* that provide at least a minimal amount of runtime to the adversary, which addresses the incompleteness of the dummy adversary. This restriction is not necessary in the IITM model due to its general runtime notion.

- The corruption mechanism has been redefined, which addresses the ambiguities between the real and ideal world, and is no longer hard-wired into the underlying model. In this aspect the current UC model follows the spirit of the IITM model.

However, not all issues of the 2005 version have been addressed: the runtime definition is still the same, leading to all of the aforementioned problems such as exhaustible machines. Some parts remain hard-wired into the model, such as addressing mechanisms and protocol structure (including the assumption of disjoint protocol sessions); thus the composition theorem of the basic UC model is still not directly applicable to protocols with joint state, global state, or other kinds of shared state. Besides these issues remaining from the 2005 version, the 2013 also has two additional major issues: Firstly, the composition theorem is still formally invalid. Secondly, it is very hard to actually prove the realization relation for any two protocols due to model related artificial attacks. We explain both issues in the following.

The composition theorem of the 2013 version of the UC model does not hold true, even for very simple protocols. To see this, first recall that an environment may send messages to the challenge protocol[39] in the name of other instances, where both the machine code and the ID (consisting of PID and SID) of the sender are chosen by the environment. This allows the environment to simulate higher level protocols that use the challenge protocol, which is the underlying argument used in the proof of the composition theorem. However, in the 2013 version of UC, the environment may not send any messages in the name of an instance that has the same SID as the challenge session;[40] this is probably meant to prevent the environment from taking part in the challenge session in the name of one of the (uncorrupted) parties (cf. [4, p. 35]). This restriction causes the composition theorem to fail as the environment cannot simulate a higher level protocol in the same session. Let us illustrate this issue by giving a concrete example where composition fails.

Recall that the composition theorem (cf. [4, p. 37 and p. 48]) essentially states the following: Given protocol π that realizes protocol ϕ, we have that protocol ρ using (possibly several instances of a) subroutine π (written ρ^{π}) realizes ρ using (possibly several instances of a) subroutine ϕ (written ρ^{ϕ}). In other words, we can replace the subroutine such that no environment notices the difference. We now construct three protocols π, ϕ, and ρ such that π realizes ϕ but ρ^{π} does not realize ρ^{ϕ}. The protocol π accepts only inputs from higher level protocols; all other messages are ignored. Each time some input is received from a higher level protocol, π returns 0 to the sender. Protocol ϕ behaves just like π, except for cases where the SID of the sender of an input is the same as the SID of this instance of ϕ. In such a case, ϕ returns 1 instead of 0. Clearly, we have that π realizes ϕ, as the only way to distinguish both protocols is to send an input in the name of the challenge session, which the environment is not allowed to do. Now, let ρ be a "dummy protocol" that forwards inputs from higher level protocols to a subroutine of π/ϕ and returns the responses. Importantly, we require ρ to use its own SID (and a different PID) to create an instance of its subroutine π/ϕ. Note that ρ is allowed to do this in the UC model as the model does not impose any restrictions on the way SIDs are chosen. Hence, an instance of π/ϕ will be invoked via ρ with an SID that coincides with its own one. As a result, ρ^{π} always outputs 0, whereas ρ^{ϕ} always outputs 1; thus both protocols can easily be distinguished.

In Appendix C, we show that the composition theorem fails even if the requirement were added that on every protocol layer different SIDs are used. Also, the restriction placed on environments is not just a minor technical detail but rather seems to be an important feature of the UC model that is actively used in the literature. For example, Canetti uses this property in [12] to prevent the environment from accessing a global random oracle in the same session as the challenge session. Furthermore, this restriction forbids the environment from mounting some severe and artificial distinguishing attacks that can potentially prevent natural and reasonable realizations (see the following paragraph and Appendix D for details). So if one were to drop this restriction to fix the composition theorem, much of the current UC literature would be invalidated as security proofs might no longer hold true in the presence of a more powerful environment.

Another major issue of the 2013 version of the UC model (and partly also older versions) is that it allows

[39] The term "challenge protocol" denotes the protocol, including its subroutines, that the environment and adversary/simulator is interacting with (in the real or ideal world, depending on the context).

[40] We use the term "challenge session" to denote the session/SID of the top layer of the challenge protocol, i.e., the layer that can be accessed by the environment. Note that this session is uniquely defined in every run as the environment may invoke at most one session.

the environment to perform several artificial distinguishing attacks. That is, it enables the environment to artificially distinguish the real world from the ideal world even in cases where, intuitively, this should not be the case. In particular, due to some of these attacks, in common settings it becomes impossible to formally prove the realization relation for any pair of real and ideal protocols altogether.

More concretely, the artificial distinguishing attacks make use of the fact that environments can do the following (and more), with details provided in Appendix D:

- Exhaust machines in the real protocol,
- Prevent a real protocol from using (some of) its subroutines,
- Intercept and read outputs sent from subroutines (such as ideal secure channel functionalities) to higher level protocols,
- Prevent the simulator from interacting with the ideal functionality.

While dealing with distinguishing attacks is an important part of both protocol design and security analysis, the above capabilities of the environment do not relate to any meaningful attack scenarios in reality and thus only hinder a protocol designer. Indeed, because these capabilities are so unnatural and very hard or even impossible to deal with, they are ignored in the literature. In Appendix D, the capabilities are described in more detail and it is shown that due to some of these capabilities it in fact is impossible to prove any realization relations in common settings.

Clearly, this is an undesirable situation that formally invalidates most, if not all, of the current UC literature. In contrast, none of these artificial attacks exist in the IITM model. We note that, on a conceptual level, it should be possible to transfer most of the UC literature to the IITM model to obtain sound results.

11.3 GNUC Model

The GNUC model was proposed by Hofheinz and Shoup in [19]. Compared to the IITM model, many things are hard-wired into the model and fixed in a specific way, as explained next.

Features of the GNUC model compared to the IITM model. In the GNUC model, a strict hierarchical structure is assumed for protocols: In a run of a system, every machine has a unique caller, hence machines form a call tree. Also, an acyclic (static) subroutine graph is assumed. In the IITM model, we do not put such restrictions on systems. A (protocol) system is simply a system of IITMs which are connected via (named) I/O tapes, with network tapes to the adversary; the notion of a subroutine is not even explicitly defined in the IITM model. For instance, it is possible to specify a system of the form $\mathcal{P}_1 \,|\, \mathcal{P}_2 \,|\, \mathcal{P}_3$ where all three protocols are mutually connected via I/O tapes or where $\mathcal{P}_1$ is connect to $\mathcal{P}_2$, $\mathcal{P}_2$ to $\mathcal{P}_3$, and $\mathcal{P}_3$ to $\mathcal{P}_1$. In particular, no acyclic subroutine graph is imposed. Of course, if desired, acyclic subroutine graphs and call trees can also be modeled in the IITM model. For instance, if $\mathcal{P}_2$ is the only subroutine that the protocols $\mathcal{P}_1$ and $\mathcal{P}_3$ should be able to use, one would have I/O tapes only between $\mathcal{P}_1$ and $\mathcal{P}_2$ as well as between $\mathcal{P}_3$ and $\mathcal{P}_2$, but not between $\mathcal{P}_1$ and $\mathcal{P}_3$.[41] If one requires call trees, one can easily define the machines in $\mathcal{P}_1$ and $\mathcal{P}_2$ in such a way that they generate different IDs to address (different) copies of $\mathcal{P}_2$. (As usual, the CheckAddress mode of machines in $\mathcal{P}_2$ could be defined as (σ-)session versions.) However, it is also possible to specify the protocols in such a way that, say, both an instance of $\mathcal{P}_1$ and an instance of $\mathcal{P}_3$ access the same instance of $\mathcal{P}_2$, as required for example in joint state realizations. It is also not a problem to specify that all (instances of) machines in a system, including (the machines of) the protocol, the environment, and the adversary access the same instance of another machine, as required for example for settings with global state.

The GNUC model also fixes the way instances of machines are addressed by SIDs and PIDs in a very specific way. PIDs have to contain, besides the actual PID, a label whether a machine is a so-called real or ideal machine. Also, SIDs need to contain, besides the actual SID, the protocol name. It is required that if a

[41] We note that even if there are I/O tapes between say $\mathcal{P}_1$ and $\mathcal{P}_3$, this does not mean that the protocols have to use them. The protocols could be specified in such a way that they ignore such tapes and don't communicate over such tapes.

machine calls a subroutine, then it extends its SID in a unique way. In this way, a call tree is enforced where every machine has a unique caller. This, however, is problematic in settings with joint and/or global state (see below). In contrast, the IITM model does not fix a specific addressing mechanism, but allows for a very flexible way of addressing machines by a general addressing mechanism, namely using the CheckAddress mode of IITMs. In particular, it would be easy to model exactly the kind of addressing enforced in the GNUC model. However, other ways of addressing machines in the IITM are possible and often desirable or necessary (e.g., for joint/global state), as illustrated in Section 10.

Corruption is also fixed in a specific way in the GNUC model. To corrupt a machine, a machine expects to receive a corrupt message at its I/O interface. This implies that in a call tree first the top-level machines have to be corrupted by the environment (the environment in the GNUC model can only access machines on the top-level). If a machine is corrupted, the adversary can corrupt subroutines of that machine so that altogether corruption spreads from the top of a call tree to the bottom. If a machine is corrupted, it forwards all messages to the adversary. However, the adversary is not allowed to send messages to the I/O interface of subroutines of a corrupted machine, except for special corrupt messages. This form of corruption is quite specific and restricted, as discussed below. In contrast, in the IITM model corruption is not hard-wired into the model. We do not fix any form of corruption at all. Corruption can be modeled in a very flexible way as part of the specification of protocols, as illustrated in Section 10. In particular, it would be easy to model the kind of corruption fixed in the GNUC model in the IITM model, i.e., to specify protocol systems in such a way that corruption is modeled as in the GNUC model. However, this is just one possible way of modeling corruption in the IITM model, other sometimes more desirable formulations, see the explanation below, are illustrated in Section 10.

As mentioned before, in the GNUC model, the environment can only access top-level machines. It cannot directly access lower-level machines, which, for example, in settings with global state would be necessary. Also, unlike the IITM model, the adversary cannot create machines. This can be useful, for instance, in a secure channel protocol where a machine should be created on the receiver side upon receipt of the first message.

In order to guarantee that a system consisting of an environment, an adversary, and a protocol runs in polynomial time in the security parameter, in the GNUC model the runtime of the adversary is restricted in that the number of bits he is allowed to send to the protocol is bounded by a polynomial in the number of bits the adversary received from the environment. This is a kind of acyclicity condition in terms of the length of the messages that may flow between the different system components (environment, adversary, protocol). However, by this the adversary/simulator is quite restricted. For example, consider the situation where an environment invokes many instances of a protocol/functionality. Now, even if every such instance contacts the adversary, the adversary could not respond to all instances because he is restricted in the number of bits he may send to the protocol. To slightly mitigate this problem, the concept of so-called "invited messages" is introduced in the GNUC model. These messages can be sent by the protocol to invite the adversary to send a message, even though he otherwise would not be allowed to send a message. However, the adversary is only allowed to send exactly the invited messages received from the protocol before. So if the protocol cannot know in advance which message the adversary will send, this mechanism is still insufficient. For example, due to the restriction put on the adversary/simulator, it is impossible to realize certain common functionalities in the GNUC model, e.g., for public-key and symmetric encryption, digital signatures, and MACs (see [4, 26, 27, 29]). In the IITM model, the adversary/simulator is not restricted in this way. In particular, the mentioned functionalities can easily be expressed and realized in the IITM model.

Due to the design choices and restrictions of the GNUC model sketched above, the GNUC model has several disadvantages in terms of simplicity, expressivity, and generality as discussed next.

Simplicity. The GNUC model introduces many concepts and fixes many details, as sketched above: distinction between regular and ideal machine, dummy parties, hierarchical protocol structure, very specific addressing mechanism with SID and PID having specific forms, specific form of corruption, invited and uninvited messages. This makes the model much more complex than the IITM model, where these concepts are not required and these things are not fixed. Also, when formulating protocols and functionalities in the GNUC

model one has to think more about runtime issues. In particular, due to the need for invited messages the specifications tend to be artificial. In the IITM one does not have to think about runtime issues too much when specifying protocols and functionalities because typical formulations will simply be environmentally (strictly) bounded.

A slight advantage of the runtime notion of the GNUC model is that it guarantees that if a protocol running with an ideal functionality is polynomially bounded in the GNUC sense, then this is also the case when the ideal functionality is replaced by its realization. However, since it is typically easy to see whether a protocol is polynomially bounded in the IITM sense, i.e., whether it is environmentally (strictly) bounded, and since in most applications protocol systems are environmentally (strictly) bounded anyway, we do not consider this to be a big advantage; even more so considering the disadvantages of the runtime notion used in the GNUC model, which makes the model harder to use and less expressive (additional flow restrictions, necessity of invited messages, certain natural functionality cannot be realized at all due to the restrictions put on simulators).

Expressivity and generality. The IITM model is much more expressive than the GNUC model in many respects. As already mentioned, the IITM model does not impose a hierarchical structure on protocol systems. This, in combination with the fact that in the IITM model the environment is not restricted to only access top-level protocols, not only allows for modeling a bigger class of protocols (see above), but also allows us to handle joint and global state without changing the model. The joint state theorem and the main global setup theorems are even direct consequences of the composition theorem. In the GNUC model, dealing with joint state and global state required non-trivial extensions of the model and to reprove theorems, such as the composition theorem, where global state is not even dealt with in its full generality. The IITM model is also more expressive and flexible in terms of the kind of corruption it can handle, as already explained above and illustrated in Section 10. In fact, the way corruption is handled (and fixed) in the GNUC model is quite restricted and sometimes insufficient. As explained, machines can only be corrupted top-down. However, it might make sense to corrupt a subroutine without corrupting the top-level protocol. For example, a protocol might use some kind of double encryption to guarantee that if one encryption fails, confidentiality is still guaranteed. So the higher level protocol can potentially still achieve its task even though parts of the subroutines are corrupted. Another example is that in e-voting a fraction of mixnet servers can be corrupted while the overall e-voting system is still secure. Such settings cannot be handled in the GNUC model. Moreover, the fact that in the GNUC model an adversary cannot send messages via a corrupted machine to uncorrupted subroutines is also a restriction. Consider for example a system were several processes use the same secure channel (which might be established using some hardware tokens). If one of the processes is corrupted, the adversary should be allowed to send and receive messages over the secure channel via the corrupted process. In the current formulation of the GNUC model this cannot be modeled. Another aspect where the IITM model is much more expressive and flexible is the addressing of machines. This, as explained above, is fixed in the GNUC model by a specific use and form of SIDs and PIDs. Since, as explained in [28], such IDs are used in an essential way in protocols (not just for the purpose of addressing instances), the kind of protocols that can be designed in the GNUC model is restricted and the faithful analysis of existing protocols is often impossible. The notion of runtime used in the GNUC model is more restricted than the one used in the IITM model as well. In particular, certain (natural) functionalities cannot be realized due to the notion of runtime that is used, as already mentioned above. Finally, we remark that in the GNUC model, unlike the IITM model, the adversary may not create machines. However, this is inconvenient. It, for example, does not allow one to model a secure channel in such a way that only one party indicates that it wants to establish a secure channel and an instance of the other party is created when that instance receives the first message from the network.

The fact that the IITM model does not a priori fix certain details (addressing of machines, corruption, protocol structure, etc.) has also the big advantage that the theorems proved, such as the composition, joint state, and global setup theorems, are much more general than in the GNUC model, as they hold true independently of specific choices; put otherwise, they hold true for all specific choices. The generality also often makes the proofs simpler and more elegant in that they are not cluttered with unessential details. In

the GNUC model, the proof of the joint state theorem, for example, used in an essential way the concept of invited messages and the way corruption is defined.

Compared to the GNUC model, one could take the simplicity, expressivity, and generality of the IITM model against the model: For the design and analysis of concrete protocols several things have to be fixed, as part of the protocol specification, because they are not a priori fixed in the IITM model. However, we consider the high flexibility a big advantage because it makes the model much simpler as well as more expressive and general compared to the GNUC model and other models, as explained above. Also, one can easily define conventions (as illustrated in Section 10.1), e.g., for corruption, and later refer to them for concrete design and analysis tasks.

References

[1] B. Barak, Y. Lindell, and T. Rabin. Protocol Initialization for the Framework of Universal Composability. Technical Report 2004/006, Cryptology ePrint Archive, 2004. Available at http://eprint.iacr.org/2004/006.

[2] J. Camenisch, R. R. Enderlein, S. Krenn, R. Küsters, and D. Rausch. Universal Composition with Responsive Environments. In J. H. Cheon and T. Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security*, volume 10032 of *Lecture Notes in Computer Science*, pages 807–840. Springer, 2016. A full version is available at https://eprint.iacr.org/2016/034.

[3] J. Camenisch, S. Krenn, R. Küsters, D. Rausch. iUC: Flexible Universal Composability Made Simple. In *Advances in Cryptology - ASIACRYPT 2019, 25th International Conference on the Theory and Applications of Cryptology and Information Security*. 2019. A full version is available at http://eprint.iacr.org/2019/1073.

[4] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. Technical Report 2000/067, Cryptology ePrint Archive, 2000. Available at http://eprint.iacr.org/2000/067 with new versions from December 2005, July 2013, and December 2018.

[5] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS 2001)*, pages 136–145. IEEE Computer Society, 2001.

[6] R. Canetti. Security and Composition of Cryptographic Protocols: A Tutorial. Technical Report 2006/465, Cryptology ePrint Archive, 2006. Available at http://eprint.iacr.org/2006/465.

[7] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Time-Bounded Task-PIOAs: A Framework for Analyzing Security Protocols. In S. Dolev, editor, *20th International Symposium on Distributed Computing (DISC 2006)*, pages 238–253. Springer, 2006.

[8] R. Canetti, A. Cohen, and Y. Lindell. A Simpler Variant of Universally Composable Security for Standard Multiparty Computation. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2015.

[9] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally Composable Security with Global Setup. In S. P. Vadhan, editor, *Theory of Cryptography, Proceedings of TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85. Springer, 2007.

[10] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie. Universally Composable Password-Based Key Exchange. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421. Springer, 2005.

[11] R. Canetti and J. Herzog. Universally Composable Symbolic Analysis of Mutual Authentication and Key-Exchange Protocols. In S. Halevi and T. Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 380–403. Springer, 2006.

[12] R. Canetti, A. Jain, and A. Scafuro. Practical UC security with a Global Random Oracle. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 597–608. ACM, 2014.

[13] R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 2002.

[14] R. Canetti and T. Rabin. Universal Composition with Joint State. In *Advances in Cryptology, 23rd Annual International Cryptology Conference (CRYPTO 2003), Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 265–281. Springer, 2003.

[15] R. Canetti, D. Shahaf, and M. Vald. Universally Composable Authentication and Key-Exchange with Global PKI. In *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part II*, volume 9615 of *Lecture Notes in Computer Science*, pages 265–296. Springer, 2016.

[16] A. Datta, R. Küsters, J. Mitchell, and A. Ramanathan. On the Relationships Between Notions of Simulation-Based Security. In J. Kilian, editor, *Proceedings of the 2nd Theory of Cryptography Conference (TCC 2005)*, volume 3378 of *Lecture Notes in Computer Science*, pages 476–494. Springer, 2005.

[17] C. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[18] D. Hofheinz, J. Müller-Quade, and D. Unruh. Polynomial Runtime in Simulatability Definitions. In *18th IEEE Computer Security Foundations Workshop (CSFW-18 2005)*, pages 156–169. IEEE Computer Society, 2005.

[19] D. Hofheinz and V. Shoup. GNUC: A New Universal Composability Framework. Technical Report 2011/303, Cryptology ePrint Archive, 2011. Available at http://eprint.iacr.org/2011/303.

[20] D. Hofheinz and D. Unruh. Comparing two Notions of Simulatability. In J. Kilian, editor, *Theory of Cryptography, Proceedings of TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 86–103. Springer, 2005.

[21] D. Hofheinz, D. Unruh, and J. Müller-Quade. Polynomial Runtime and Composability. Technical Report 2009/023, Cryptology ePrint Archive, 2009. Available at http://eprint.iacr.org/2009/023.

[22] D. Hofheinz, D. Unruh, and J. Müller-Quade, 2011. [21, footnote 26 on page 40] and personal communication.

[23] R. Küsters. Simulation-Based Security with Inexhaustible Interactive Turing Machines. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW-19 2006)*, pages 309–320. IEEE Computer Society, 2006. See http://eprint.iacr.org/2013/025/ for a full and revised version.

[24] R. Küsters, A. Datta, J. C. Mitchell, and A. Ramanathan. On the Relationships Between Notions of Simulation-Based Security. *Journal of Cryptology*, 21(4):492–546, 2008.

[25] R. Küsters and D. Rausch. A Framework for Universally Composable Diffie-Hellman Key Exchange. In *IEEE 38th Symposium on Security and Privacy (S&P 2017)*, pages 881–900. IEEE Computer Society, 2017.

[26] R. Küsters and M. Tuengerthal. Joint State Theorems for Public-Key Encryption and Digital Signature Functionalities with Local Computation. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF 2008)*, pages 270–284. IEEE Computer Society, 2008. The full version is available at https://eprint.iacr.org/2008/006, see also [30].

[27] R. Küsters and M. Tuengerthal. Universally Composable Symmetric Encryption. In *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF 2009)*, pages 293–307. IEEE Computer Society, 2009.

[28] R. Küsters and M. Tuengerthal. Composition Theorems Without Pre-Established Session Identifiers. In Y. Chen, G. Danezis, and V. Shmatikov, editors, *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS 2011)*, pages 41–50. ACM, 2011.

[29] R. Küsters and M. Tuengerthal. Ideal Key Derivation and Encryption in Simulation-based Security. In A. Kiayias, editor, *Topics in Cryptology – CT-RSA 2011, The Cryptographers' Track at the RSA Conference 2011, Proceedings*, volume 6558 of *Lecture Notes in Computer Science*, pages 161–179. Springer, 2011.

[30] R. Küsters, M. Tuengerthal, and D. Rausch. Joint State Theorems for Public-Key Encryption and Digital Signature Functionalities with Local Computation. *Journal of Cryptology*. To appear. Available at http://eprint.iacr.org/2008/006. This is the full version of [26].

[31] R. Küsters, M. Tuengerthal, and D. Rausch. The IITM Model: a Simple and Expressive Model for Universal Composability. Technical Report 2013/025, Cryptology ePrint Archive, 2013. Available at http://eprint.iacr.org/2013/025.

[32] U. Maurer. Constructive Cryptography - A New Paradigm for Security Definitions and Proofs. In *Theory of Security and Applications - Joint Workshop, TOSCA 2011, Saarbrücken, Germany, March 31 - April 1, 2011, Revised Selected Papers*, volume 6993 of *Lecture Notes in Computer Science*, pages 33–56. Springer, 2011.

[33] U. Maurer and R. Renner. Abstract Cryptography. In B. Chazelle, editor, *Innovations in Computer Science - ICS 2010. Proceedings*, pages 1–21. Tsinghua University Press, 2011.

[34] R. Milner. *A Calculus of Communicating Systems*. Springer, 1980.

[35] R. Needham and M. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, 1978.

[36] B. Pfitzmann and M. Waidner. A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In *IEEE Symposium on Security and Privacy*, pages 184–201. IEEE Computer Society, 2001.

[37] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-Lock Puzzles and Timed-Release Crypto. Technical Report MIT/LCS/TR-684, MIT, March 1996. Available at http://theory.lcs.mit.edu/~rivest/RivestShamirWagner-timelock.ps.

A Proof of Lemma 9

Let $\mathcal{P}, \mathcal{Q}$ be protocol systems such that $!\underline{\mathcal{P}}$ is environmentally bounded, $\mathcal{Q}$ is environmentally bounded, and $\mathcal{P} \cong \mathcal{Q}$. Let $\mathcal{E} \in \mathrm{Env}(!\underline{\mathcal{P}}) = \mathrm{Env}(!\underline{\mathcal{Q}})$. In the following, by a hybrid argument where we replace the copies of $\mathcal{P}$ by copies of $\mathcal{Q}$, we show that $\mathcal{E} \,|\, !\underline{\mathcal{Q}}$ is almost bounded, and hence, that $!\underline{\mathcal{Q}}$ is environmentally bounded.

By Lemma 7, we may assume that $\mathcal{E}$ is a single IITM which, in mode CheckAddress, accepts all messages. Moreover, we may assume, without loss of generality, that $\mathcal{E}$ is such that every message m that $\mathcal{E}$ outputs (except if m is output on tape decision) is prefixed by some SID, i.e., $m = (s, m')$ for some bit strings s and

m':[42] since $\mathcal{E}$ will only interact with session versions, message not of the form (s, m') would be rejected by these session versions anyway. Since $\mathcal{E}$ is universally bounded, it follows that there exists a polynomial $p_{\mathcal{E}}$ such that the number of different sessions (i.e., messages with distinct SIDs output by $\mathcal{E}$) is bounded from above by $p_{\mathcal{E}}(\eta + |a|)$ (where η is the security parameter and a is the external input).

In what follows, let $\underline{\mathcal{Q}}'$ be the variant of $\underline{\mathcal{Q}}$ obtained from $\underline{\mathcal{Q}}$ by renaming every tape c occurring in $\underline{\mathcal{Q}}$ to c'. Analogously, let $\underline{\mathcal{P}}''$ be obtained from $\underline{\mathcal{P}}$ by renaming every tape c occurring in $\underline{\mathcal{P}}$ to c''. By this, we have that $\mathcal{P}$, $\underline{\mathcal{Q}}'$, and $\underline{\mathcal{P}}''$ have pairwise disjoint sets of external tapes, and hence, these systems are pairwise connectable.

We now define an IITM $\mathcal{E}_r^{perm}$ (for every $r \in \mathbf{N}$) which basically simulates $\mathcal{E}$ and which will run in the system $\mathcal{E}_r^{perm} \,|\, !\,\underline{\mathcal{P}}'' \,|\, !\,\underline{\mathcal{Q}}' \,|\, \mathcal{P}$ or $\mathcal{E}_r^{perm} \,|\, !\,\underline{\mathcal{P}}'' \,|\, !\,\underline{\mathcal{Q}}' \,|\, \mathcal{Q}$, respectively. The IITM $\mathcal{E}_r^{perm}$ randomly shuffles the copies of the protocols invoked by $\mathcal{E}$ by choosing a random permutation on $\{1, \ldots, p_{\mathcal{E}}(\eta + |a|)\}$. The first $r-1$ copies (after shuffling) of the protocol invoked by $\mathcal{E}$ will be copies of $\underline{\mathcal{Q}}'$, the r-th copy will be the external system $\mathcal{P}$ or $\mathcal{Q}$, respectively, and the remaining copies will be copies of $\underline{\mathcal{P}}''$. Intuitively, the permutation of protocol instances is needed because $\mathcal{E}$ must not "know" whether a copy invoked in the system $!\,\underline{\mathcal{P}}'' \,|\, !\,\underline{\mathcal{Q}}' \,|\, \mathcal{P}$ is a copy of $\mathcal{P}$ or $\underline{\mathcal{P}}''$; analogously for the system $!\,\underline{\mathcal{P}}'' \,|\, !\,\underline{\mathcal{Q}}' \,|\, \mathcal{Q}$. Without the permutation in $\mathcal{E}_r^{perm}$, the internally simulated environment $\mathcal{E}$ might create, for example, protocol instances with increasingly large runtime, i.e., the runtime of the r-th instance would depend on r. However, we need to establish a runtime bound that is independent of r in Lemma 28, which then directly implies this theorem. Due to the permutation, even if the simulated $\mathcal{E}$ tries to use one instance much more than others, the expected runtimes of every instance of $\mathcal{P}$ and $\underline{\mathcal{P}}''$ ($\mathcal{Q}$ and $\underline{\mathcal{Q}}'$) are identical, allowing us to prove Lemma 28.

Formally, $\mathcal{E}_r^{perm}$ is obtained from $\mathcal{E}$ as follows (recall that we assume that $\mathcal{E}$ is a single IITM which accepts every message in mode CheckAddress). The IITM $\mathcal{E}_r^{perm}$ will also always accept every message in mode CheckAddress. The behavior of $\mathcal{E}_r^{perm}$ in mode Compute is specified next.

First, we need to make sure that $\mathcal{E}_r^{perm}$ has the appropriate tapes to connect to the different entities. The IITM $\mathcal{E}$ already has tapes to connect to the external tapes of $\mathcal{P}$ and $\mathcal{Q}$. For each such tape c, we add to $\mathcal{E}_r^{perm}$ a tape c' and c'' to connect to the external tapes of $\underline{\mathcal{Q}}'$ and $\underline{\mathcal{P}}''$, respectively.

Next, we need to specify how $\mathcal{E}_r^{perm}$ redirects protocol invocations of $\mathcal{E}$ in the way sketched above: $\mathcal{E}_r^{perm}$ keeps a list L of SIDs, which initially is empty, and the length l of the list, which initially is 0. By definition of $p_{\mathcal{E}}$, it will always hold that $l \leq p_{\mathcal{E}}(\eta + |a|)$. Furthermore, in the first activation with security parameter $\eta \in \mathbf{N}$ and external input $a \in \{0,1\}^*$, $\mathcal{E}_r^{perm}$ chooses a permutation π of $\{1, \ldots, p_{\mathcal{E}}(\eta + |a|)\}$ uniformly at random. From now on, $\mathcal{E}_r^{perm}$ simulates $\mathcal{E}$ with security parameter η and external input a. In particular, if $\mathcal{E}$ produces output, then so does $\mathcal{E}_r^{perm}$, and if $\mathcal{E}_r^{perm}$ receives input, then $\mathcal{E}$ is simulated with this input. However, as explained next, the behavior of $\mathcal{E}_r^{perm}$ deviates from that of $\mathcal{E}$ when it comes to sending and receiving messages to the different copies of protocols.

1. If $\mathcal{E}$ produces output $m = (s, m')$, for some SID s and some message m', on some external tape c of $\underline{\mathcal{P}}$ (and hence, $\underline{\mathcal{Q}}$), then $\mathcal{E}_r^{perm}$ checks whether s occurs in L. If s does not occur in L, s is first appended at the end of L and l is increased by 1. Let $j \in \{1, \ldots, l\}$ be the position where s occurs in L.
   (a) If $\pi(j) < r$, then $\mathcal{E}_r^{perm}$ writes m on tape c'.
   (b) If $\pi(j) = r$, then $\mathcal{E}_r^{perm}$ outputs m' on c.
   (c) If $\pi(j) > r$, then $\mathcal{E}_r^{perm}$ writes m on tape c''.
2. If $\mathcal{E}_r^{perm}$ receives input on tape c'' (where c'' is an external tape of $\underline{\mathcal{P}}''$ corresponding to an external tape c of $\underline{\mathcal{P}}$), then $\mathcal{E}_r^{perm}$ behaves as $\mathcal{E}$ in case input was received on tape c.
3. If $\mathcal{E}_r^{perm}$ receives input on tape c' (where c' is an external tape of $\underline{\mathcal{Q}}'$ corresponding to an external tape c of $\underline{\mathcal{Q}}$), then $\mathcal{E}_r^{perm}$ behaves as $\mathcal{E}$ in case input was received on tape c.
4. If $\mathcal{E}_r^{perm}$ receives input m on tape c where c is an external tape of $\mathcal{P}$ (and hence, $\mathcal{Q}$), then $\mathcal{E}_r^{perm}$ behaves as $\mathcal{E}$ in case input $(L[\pi^{-1}(r)], m)$ was received on tape c where $L[\pi^{-1}(r)]$ denotes the $\pi^{-1}(r)$-st entry

[42] More formally, for every system $\mathcal{S} \in \mathrm{Con}(\mathcal{E})$ and all parameters η, a in every run of $(\mathcal{E} \,|\, \mathcal{S})(1^\eta, a)$ the system $\mathcal{E}$ should only output messages of the described form.

of L. By construction, this entry exists in L (i.e., $\pi^{-1}(r) \leq l$) since $\mathcal{E}$ must have invoked the $\pi^{-1}(r)$-th copy.

It is easy to see that $\mathcal{E}_r^{perm}$ is universally bounded for every $r \in \mathbf{N}$ since $\mathcal{E}$ is universally bounded.

We also consider a variant $\mathcal{E}_\$$ of $\mathcal{E}_r^{perm}$ which in the first activation additionally chooses $r \in \{1, \ldots, p_{\mathcal{E}}(\eta + |a|)\}$ uniformly at random and then behaves exactly like $\mathcal{E}_r^{perm}$.

We define the following hybrid systems, for every $r \in \mathbf{N}$:

$$\mathcal{H}_r := \mathcal{E}_r^{perm} \,|\, !\,\underline{\mathcal{P}''} \,|\, !\,\underline{\mathcal{Q}'} \ ,$$

which can be connected to $\mathcal{P}$ (and hence $\mathcal{Q}$), i.e., $\mathcal{H}_r \in \mathrm{Con}(\mathcal{P}) = \mathrm{Con}(\mathcal{Q})$.

By $\mathcal{E}_{p_\mathcal{E}}^{perm}$ and $\mathcal{H}_{p_\mathcal{E}}$ we denote the systems $\mathcal{E}_r^{perm}$ and $\mathcal{H}_r$, respectively, which use $r = p_{\mathcal{E}}(\eta + |a|)$. By construction, for every $r \in \mathbf{N}$, the systems $\mathcal{E} \,|\, !\,\underline{\mathcal{P}}$ and $\mathcal{H}_1 \,|\, \mathcal{P}$, $\mathcal{H}_r \,|\, \mathcal{Q}$ and $\mathcal{H}_{r+1} \,|\, \mathcal{P}$, and $\mathcal{E} \,|\, !\,\underline{\mathcal{Q}}$ and $\mathcal{H}_{p_\mathcal{E}} \,|\, \mathcal{Q}$, respectively, behave exactly the same (see below). In particular, for all $r \in \mathbf{N}$, we have that:

$$\mathcal{E} \,|\, !\,\underline{\mathcal{P}} \equiv_0 \mathcal{H}_1 \,|\, \mathcal{P} \ , \tag{29}$$

$$\mathcal{H}_r \,|\, \mathcal{Q} \equiv_0 \mathcal{H}_{r+1} \,|\, \mathcal{P} \ , \text{ and} \tag{30}$$

$$\mathcal{E} \,|\, !\,\underline{\mathcal{Q}} \equiv_0 \mathcal{H}_{p_\mathcal{E}} \,|\, \mathcal{Q} \ . \tag{31}$$

For (29) we use that $\mathcal{P}$ is a protocol system. In particular, we use property (ii) of protocol systems (see Definition 11). If this property were not satisfied, i.e., $\mathcal{P}$ contains an IITM M which is not in the scope of a bang but which could reject a message in mode CheckAddress, the following could happen. In a run of $(\mathcal{H}_1 \,|\, \mathcal{P})(1^\eta, a)$ a message is sent to M, but it is rejected by M (in mode CheckAddress). Then, since M is not in the scope of a bang, no new copy of M will be generated. Conversely, if in a run of $\mathcal{E} \,|\, !\,\underline{\mathcal{P}}$ a message is sent to a copy of the session version $\underline{M}$ of M prefixed with the first SID generated by $\mathcal{E}$ and the simulated M in $\underline{M}$ would reject the message, then it could happen that a new copy of $\underline{M}$ is generated (since $\underline{M}$ is in the scope of a bang in $\mathcal{E} \,|\, !\,\underline{\mathcal{P}}$) which then would not have a corresponding entity in a run of the system $(\mathcal{H}_1 \,|\, \mathcal{P})(1^\eta, a)$. In short, by property (ii) of protocol systems it is guaranteed that for IITMs that do not occur in the scope of a bang in $\mathcal{P}$ only at most one copy is generated per SID in the run of $\mathcal{E} \,|\, !\,\underline{\mathcal{P}}$. A similar argument is used to prove (30) and (31).

Since $\mathcal{E} \,|\, !\,\underline{\mathcal{P}}$ is almost bounded, it is easy to see that $\mathcal{H}_1 \,|\, \mathcal{P}$ is almost bounded too. Moreover, it is easy to see that $\mathcal{E} \,|\, !\,\underline{\mathcal{Q}}$ is almost bounded if and only if $\mathcal{H}_{p_\mathcal{E}} \,|\, \mathcal{Q}$ is almost bounded.

Next, we define the event (i.e., a set of runs or equivalently a set of random coins) that the j-th copy of the protocol takes more than $q(\eta + |a|)$ steps. More specifically, for every polynomial q, natural numbers $r, i, j, \eta \in \mathbf{N}$, and $a \in \{0,1\}^*$ we define:

1. $B^{q,j}_{\mathcal{H}_r \,|\, \mathcal{P}} = B^{q,j}_{\mathcal{H}_r \,|\, \mathcal{P}}(1^\eta, a)$ is the following set of runs of $(\mathcal{H}_r \,|\, \mathcal{P})(1^\eta, a)$. A run ρ belongs to $B^{q,j}_{\mathcal{H}_r \,|\, \mathcal{P}}$ iff one of the following conditions is satisfied:
   (a) $\pi(j) < r$ and the copies of machines in $\underline{\mathcal{Q}'}$ with SID $L[j]$ took more than $q(\eta + |a|)$ steps in ρ, where according to Section 3.4 only the steps in mode Compute are counted and where in addition the overhead created by the session version is not counted either; in other words, only the computation steps carried out by $\mathcal{Q}'$ (and hence, $\mathcal{Q}$) are counted.
   (b) $\pi(j) = r$ and the machines in $\mathcal{P}$ took more than $q(\eta + |a|)$ steps in ρ, with the steps counted as above.
   (c) $\pi(j) > r$ and the copies of machines in $\underline{\mathcal{P}''}$ with SID $L[j]$ took more than $q(\eta + |a|)$ steps in ρ, with the steps counted as above.
2. $B^{q}_{\mathcal{H}_r \,|\, \mathcal{P}} := \bigcup_{i \in \mathbf{N}} B^{q,i}_{\mathcal{H}_r \,|\, \mathcal{P}}$.
3. $B^{q,\neq j}_{\mathcal{H}_r \,|\, \mathcal{P}} := \bigcup_{i \in \mathbf{N} \setminus \{j\}} B^{q,i}_{\mathcal{H}_r \,|\, \mathcal{P}}$.

4. A run ρ belongs to $B^{q,\pi^{-1}(i)}_{\mathcal{H}_r \mid \mathcal{P}}$ iff for the permutation, say π', chosen in ρ, it holds that $\rho \in B^{q,j}_{\mathcal{H}_r \mid \mathcal{P}}$ with $j = \pi'^{-1}(i)$. The event $B^{q,\neq\pi^{-1}(i)}_{\mathcal{H}_r \mid \mathcal{P}}$ is defined analogously.

5. Analogously to the above events, we define the following events where the external system $\mathcal{P}$ is replaced by $\mathcal{Q}$: $B^{q,j}_{\mathcal{H}_r \mid \mathcal{Q}}$, $B^{q}_{\mathcal{H}_r \mid \mathcal{Q}}$, $B^{q,\neq j}_{\mathcal{H}_r \mid \mathcal{Q}}$, $B^{q,\pi^{-1}(i)}_{\mathcal{H}_r \mid \mathcal{Q}}$, $B^{q,\neq\pi^{-1}(i)}_{\mathcal{H}_r \mid \mathcal{Q}}$.

We note that for all r, the systems $\mathcal{H}_r \mid \mathcal{P}$ (resp., $\mathcal{H}_r \mid \mathcal{Q}$) is almost bounded if and only if there exists a negligible function f and a polynomial q such that $\text{Prob}[B^{q}_{\mathcal{H}_r \mid \mathcal{P}}(1^\eta, a)] \leq f(\eta, a)$ (resp., $\text{Prob}[B^{q}_{\mathcal{H}_r \mid \mathcal{Q}}(1^\eta, a)] \leq f(\eta, a)$) for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$. The direction from left to right is trivial. For the other direction first recall that (the simulated) $\mathcal{E}$ is universally bounded, and hence, only a polynomial number of copies of the protocol is created (the length l of the list L in $\mathcal{H}_r$ is bounded by $p_{\mathcal{E}}(\eta + |a|)$). Now since, by assumption, the number of steps taken by every copy of the protocol is polynomially bounded (except with negligible probability) and the number of steps taken by (the simulated) $\mathcal{E}$ is polynomially bounded, the number of steps taken by $\mathcal{H}_r \mid \mathcal{P}$ (resp., $\mathcal{H}_r \mid \mathcal{Q}$) is polynomially bounded. (The overhead for the session versions is only polynomial.)

The next simple lemma shows that there exists a polynomial q_1 such that the event $B^{q_1}_{\mathcal{H}_1 \mid \mathcal{Q}}$ occurs with negligible probability. As argued above, this is equivalent to $\mathcal{H}_1 \mid \mathcal{Q}$ being almost bounded. By induction on r, using Theorem 4 and $\mathcal{P} \cong \mathcal{Q}$, one can even easily show that $\mathcal{H}_r \mid \mathcal{Q}$ is almost bounded for every $r \in \mathbf{N}$. This however does not suffice to prove Lemma 9 because we need to show that $\mathcal{H}_{p_\mathcal{E}} \mid \mathcal{Q}$ is almost bounded. That is, we need a uniform bound that is independent of r (as will be established in Lemma 28). This is in fact the only real difficulty in proving this lemma and the only reason the permutation π is required.

Lemma 26. *There exists a polynomial q_1 and a negligible function f_1 such that for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$:*

$$\text{Prob}[B^{q_1}_{\mathcal{H}_1 \mid \mathcal{Q}}(1^\eta, a)] \leq f_1(\eta, a) \ .$$

Proof. By construction, $\mathcal{H}_1 \mid \mathcal{P}$ behaves exactly like $\mathcal{E} \mid !\underline{\mathcal{P}}$, which is almost bounded by assumption. By definition of $\mathcal{H}_1$, the subsystem $!\underline{\mathcal{Q}'}$ of $\mathcal{H}_1$ is never invoked, and hence, can be removed from $\mathcal{H}_1$. Since $!\underline{\mathcal{P}}$ is environmentally bounded by assumption, it is easy to see that the system $!\underline{\mathcal{P}''} \mid \mathcal{P}$ is environmentally bounded as well. Now, by Theorem 4 (with $\mathcal{S} = !\underline{\mathcal{P}''}$) and $\mathcal{P} \cong \mathcal{Q}$, we obtain that $\mathcal{E} \mid !\underline{\mathcal{P}''} \mid \mathcal{Q}$ is almost bounded, and hence, $\mathcal{H}_1 \mid \mathcal{Q}$ is almost bounded. As argued above, this implies the existence of a polynomial q_1 and a negligible function f_1 with $\text{Prob}[B^{q_1}_{\mathcal{H}_1 \mid \mathcal{Q}}(1^\eta, a)] \leq f_1(\eta, a)$ for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$. □

Throughout the rest of the proof we fix the polynomial q_1 and the negligible function f_1 from the above lemma.

The next lemma shows that for all $r \leq p_\mathcal{E}(\eta + |a|)$ the difference of the probabilities that the events $B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \mid \mathcal{P}}$ and $B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \mid \mathcal{Q}}$ occur is negligible. Note that this statement does not consider the number of steps taken by the external protocol copy $\mathcal{P}$ and $\mathcal{Q}$, respectively, because this copy, which corresponds to $\pi^{-1}(r)$, is excluded in these events.

Lemma 27. *There exists a negligible function f_2 such that for all $\eta \in \mathbf{N}$, $a \in \{0,1\}^*$, and $r \leq p_\mathcal{E}(\eta + |a|)$:*

$$\left|\text{Prob}\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \mid \mathcal{P}}(1^\eta, a)\right] - \text{Prob}\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \mid \mathcal{Q}}(1^\eta, a)\right]\right| \leq f_2(\eta, a) \ .$$

Proof. We first prove the lemma for non-uniform environments, i.e., environments that use the external input a, because this proof is simpler and better highlights the main idea. In Appendix B we present a proof for this lemma in the uniform setting.

We define an IITM $D \in \text{Env}(\mathcal{P})$ which expects to receive (r, a) as external input and then simulates $\mathcal{H}_r$ with external input a until the runtime bound $q_1(\eta + |a|)$ is hit by any of the internally simulated sessions. If the runtime bound is hit, D outputs 1 on decision. If the run stops and the runtime bound has not been hit, D outputs 0 on decision.

More formally, D is defined as follows. In mode CheckAddress, D accepts every message. In mode Compute, D behaves as follows: First, D parses the external input on start as (r, a) with $r \in \{1, \ldots, p_\mathcal{E}(\eta + |a|)\}$ and

$a \in \{0,1\}^*$. If the external input is not of this form, D outputs 0 on decision. Otherwise, D simulates the system $\mathcal{H}_r = \mathcal{E}_r^{perm} \,|\, !\underline{\mathcal{P}}'' \,|\, !\,\underline{\mathcal{Q}}'$ with external input a, i.e., it first activates $\mathcal{E}_r^{perm}$ with input a on start and then simulates all the machines in $\mathcal{H}_r$. If $\mathcal{H}_r$ produces output, then so does D and if D receives input, then D forwards this input to the simulated $\mathcal{H}_r$. Additionally, D counts the number of transitions taken by all the simulated machines and checks if the conditions of the event $B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{P}}(1^\eta, a)$ and $B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)$, respectively, are satisfied. (Note that D can do this even though it cannot inspect the number of transitions in the external systems, $\mathcal{P}$ and $\mathcal{Q}$, respectively.) If such a condition is satisfied (i.e., some internal session takes more than $q_1(\eta + |a|)$ steps), then D halts with output 1 on decision. If the run terminates (i.e., $\mathcal{E}_r^{perm}$ outputs something on decision or $\mathcal{E}_r^{perm}$ produces empty output, which terminates the run because $\mathcal{E}_r^{perm}$ is a master IITM) but these conditions are not satisfied, then D halts with output 0 on decision.

It is easy to see that D is universally bounded (i.e., $D \in \mathrm{Env}(\mathcal{P})$) and that for all $\eta \in \mathbf{N}$, $a \in \{0,1\}^*$, and $r \leq p_{\mathcal{E}}(\eta + |a|)$:

$$\begin{aligned} \mathrm{Prob}[(D \,|\, \mathcal{P})(1^\eta, (r,a)) = 1] &= \mathrm{Prob}\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{P}}(1^\eta, a)\right] \text{ and} \\ \mathrm{Prob}[(D \,|\, \mathcal{Q})(1^\eta, (r,a)) = 1] &= \mathrm{Prob}\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right] \ . \end{aligned} \tag{32}$$

Since $\mathcal{P} \cong \mathcal{Q}$ and $D \in \mathrm{Env}(\mathcal{P})$, we have that $D \,|\, \mathcal{P} \equiv D \,|\, \mathcal{Q}$. In particularly, there exists a negligible function f_2' such that for all $\eta \in \mathbf{N}$, $a \in \{0,1\}^*$, and $r \leq p_{\mathcal{E}}(\eta + |a|)$:

$$\begin{aligned} f_2'(\eta, (r,a)) \ &\geq \ |\mathrm{Prob}[(D \,|\, \mathcal{P})(1^\eta, (r,a)) = 1] - \mathrm{Prob}[(D \,|\, \mathcal{Q})(1^\eta, (r,a)) = 1]| \\ &\overset{(32)}{=} \left|\mathrm{Prob}\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{P}}(1^\eta, a)\right] - \mathrm{Prob}\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right]\right| \ . \end{aligned}$$

Let $f_2(\eta, a) := \max_{r \leq p_{\mathcal{E}}(\eta+|a|)} f_2'(\eta, (r,a))$. It is easy to see that f_2 is a negligible function. Now, obviously we have that

$$\left|\mathrm{Prob}\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{P}}(1^\eta, a)\right] - \mathrm{Prob}\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right]\right| \leq f_2(\eta, a)$$

for all $\eta \in \mathbf{N}$, $a \in \{0,1\}^*$, and $r \leq p_{\mathcal{E}}(\eta + |a|)$. This concludes the proof. □

Throughout the rest of the proof, we fix the negligible function f_2 from the above lemma. Finally, we show that there exists a negligible function such that for all $r \leq p_{\mathcal{E}}(\eta + |a|)$ the probability that the event $B^{q_1}_{\mathcal{H}_r \,|\, \mathcal{Q}}$ occurs is bounded by this negligible function. This will conclude the proof.

Lemma 28. *There exists a negligible function f_3 such that for all $\eta \in \mathbf{N}$, $a \in \{0,1\}^*$, and $r \leq p_{\mathcal{E}}(\eta + |a|)$:*

$$\mathrm{Prob}\left[B^{q_1}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right] \leq f_3(\eta, a) \ .$$

Proof. Let $\eta \in \mathbf{N}$, and $a \in \{0,1\}^*$. For all $r \leq p_{\mathcal{E}}(\eta + |a|)$ we define

$$t_r := t_r(1^\eta, a) := \mathrm{Prob}\left[B^{q_1}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right] \ .$$

We need to show that there exists a negligible function that bounds t_r from above for every $r \leq p_{\mathcal{E}}(\eta + |a|)$.

Let $r \leq p_{\mathcal{E}}(\eta + |a|)$. If $r = 1$, then $t_r = t_1 \leq f_1(\eta, a)$ by Lemma 26. Next, we consider the case $r > 1$. Since the permutation π is chosen uniformly at random, we obtain for all $j \leq r$ the following equality:

$$\mathrm{Prob}\left[B^{q_1,\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a) \setminus B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right] = \mathrm{Prob}\left[B^{q_1,\pi^{-1}(j)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a) \setminus B^{q_1,\neq\pi^{-1}(j)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right] \ . \tag{33}$$

Intuitively, the event $B_1 := B^{q_1,\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a) \setminus B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)$ says that the number of steps taken in the external protocol system $\mathcal{Q}$ exceeded q_1, but not the number of steps in the internal systems. The event $B_2 := B^{q_1,\pi^{-1}(j)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a) \setminus B^{q_1,\neq\pi^{-1}(j)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)$ says that the number of steps taken in the jth copy of the internally simulated protocol system $\mathcal{Q}$ exceeded q_1, but not the number of steps taken in the other protocol

systems, i.e., in the other internal copies and the external system. Since π is chosen uniformly at random, it does not make a difference whether q_1 is exceeded by the external system or one of the internal copies of $\mathcal{Q}$.

Formally, the equality (33) can easily be established by defining a bijection between the runs in B_1 and those in B_2: a run $\rho \in B_1$ in which the permutation π was chosen is mapped to the corresponding run in B_2 where a permutation π' is chosen which coincides with π except that $\pi^{-1}(r)$ and $\pi^{-1}(j)$ are swapped, i.e., $\pi'^{-1}(r) = \pi^{-1}(j)$ and $\pi'^{-1}(j) = \pi^{-1}(r)$.

Now, with (33) we have that:

$$\begin{aligned}
\mathrm{Prob}[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)] &\geq \mathrm{Prob}\left[\bigcup_{j=1}^{r-1} B^{q_1,\pi^{-1}(j)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right] \\
&\geq \mathrm{Prob}\left[\bigcup_{j=1}^{r-1} B^{q_1,\pi^{-1}(j)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a) \setminus B^{q_1,\neq\pi^{-1}(j)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right] \\
&= \sum_{j=1}^{r-1} \mathrm{Prob}\left[B^{q_1,\pi^{-1}(j)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a) \setminus B^{q_1,\neq\pi^{-1}(j)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right] \\
&\overset{(33)}{=} (r-1) \cdot \mathrm{Prob}\left[B^{q_1,\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a) \setminus B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right] \quad .
\end{aligned} \tag{34}$$

We conclude that:

$$\begin{aligned}
t_r &= \mathrm{Prob}\left[B^{q_1}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right] \\
&= \mathrm{Prob}\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right] + \mathrm{Prob}\left[B^{q_1,\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a) \setminus B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right] \\
&\leq \frac{r}{r-1} \cdot \mathrm{Prob}\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)\right] && \text{due to (34)} \\
&\leq \frac{r}{r-1} \cdot \left(\mathrm{Prob}\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{P}}(1^\eta, a)\right] + f_2(\eta, a)\right) && \text{due to Lemma 27} \\
&\leq \frac{r}{r-1} \cdot \left(\mathrm{Prob}\left[B^{q_1}_{\mathcal{H}_r \,|\, \mathcal{P}}(1^\eta, a)\right] + f_2(\eta, a)\right) \quad .
\end{aligned}$$

Since the systems $\mathcal{H}_r \,|\, \mathcal{P}$ and $\mathcal{H}_{r-1} \,|\, \mathcal{Q}$ behave exactly the same, we have that

$$\mathrm{Prob}[B^{q_1}_{\mathcal{H}_r \,|\, \mathcal{P}}(1^\eta, a)] = \mathrm{Prob}[B^{q_1}_{\mathcal{H}_{r-1} \,|\, \mathcal{Q}}(1^\eta, a)] = t_{r-1}$$

and we obtain the following simple recurrence relation:

$$t_r \leq \frac{r}{r-1} \cdot (t_{r-1} + f_2(\eta, a)) \quad .$$

By induction on r it can be shown that:

$$t_r \leq \left(\prod_{j=1}^{r-1} \frac{j+1}{j}\right) \cdot t_1 + \left(\sum_{j=1}^{r-1} \prod_{i=j}^{r-1} \frac{i+1}{i}\right) \cdot f_2(\eta, a) \quad . \tag{35}$$

From this we obtain:

$$\begin{aligned}
t_r &\leq \left(\prod_{j=1}^{r-1} \frac{j+1}{j}\right) \cdot t_1 + \left(\sum_{j=1}^{r-1} \prod_{i=j}^{r-1} \frac{i+1}{i}\right) \cdot f_2(\eta, a) && \text{due to (35)} \\
&= r \cdot t_1 + \left(\sum_{j=1}^{r-1} \frac{r}{j}\right) \cdot f_2(\eta, a) \\
&\leq r \cdot t_1 + (r-1) \cdot r \cdot f_2(\eta, a) \\
&\leq p_{\mathcal{E}}(\eta + |a|) \cdot f_1(\eta, a) + p^2_{\mathcal{E}}(\eta + |a|) \cdot f_2(\eta, a) && \text{due to Lemma 26.}
\end{aligned}$$

Hence, with

$$f_3(\eta, a) := p_{\mathcal{E}}(\eta + |a|) \cdot f_1(\eta, a) + p_{\mathcal{E}}^2(\eta + |a|) \cdot f_2(\eta, a)$$

it holds that

$$\mathrm{Prob}[B^{q_1}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta, a)] = t_r \leq f_3(\eta, a) \ ,$$

for all $r \leq p_{\mathcal{E}}(\eta + |a|)$. Note that f_3 is negligible and does not depend on r. □

Lemma 28 immediately implies that $\mathrm{Prob}[B^{q_1}_{\mathcal{H}_{p_{\mathcal{E}}(\eta+|a|)} \,|\, \mathcal{Q}}(1^\eta, a)] \leq f_3(\eta, a)$ for all $\eta \in \mathbf{N}$ and $a \in \{0,1\}^*$. As argued above, it follows that $\mathcal{H}_{p_{\mathcal{E}}} \,|\, \mathcal{Q}$, and hence, $\mathcal{E} \,|\, !\,\underline{\mathcal{Q}}$ are almost bounded. Since this is true for any $\mathcal{E} \in \mathrm{Env}(!\,\underline{\mathcal{Q}})$, we obtain that $!\,\underline{\mathcal{Q}}$ is environmentally bounded, which concludes the proof of Lemma 9.

B Proof of Lemma 27 for Uniform Environments

We present the proof of Lemma 27 for uniform environments, i.e., environments which do not get any external input. Let $\eta \in \mathbf{N}$. We define:

$$\delta_r := \delta_r(1^\eta) := \mathrm{Prob}\left[B^{q_1, \neq \pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{P}}(1^\eta)\right] - \mathrm{Prob}\left[B^{q_1, \neq \pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{Q}}(1^\eta)\right]$$

for all $r \leq p_{\mathcal{E}}(\eta)$. (Recall that $p_{\mathcal{E}}$ is the polynomial that bounds the number of different sessions created by $\mathcal{E}$. Since $\mathcal{E}$ does not get any external input this polynomial is now in the security parameter alone.) Let $r^* \leq p_{\mathcal{E}}(\eta)$ such that it maximizes $|\delta_r|$, i.e., $|\delta_{r^*}| = \max\{|\delta_r| \mid r \leq p_{\mathcal{E}}(\eta)\}$. The basic idea of this proof is due to [22] and goes as follows: D, instead of reading r from the external input (recall that in this setting D does not obtain external input) first samples runs of $(\mathcal{H}_r \,|\, \mathcal{P})(1^\eta)$ and $(\mathcal{H}_r \,|\, \mathcal{Q})(1^\eta)$ and tries to compute r^*. Then D simulates $\mathcal{H}_r$ as in the non-uniform case (see Appendix A) with $r = r^*$. Of course, D might only find an r such that $|\delta_r|$ approximates $|\delta_{r^*}|$, but this will be good enough.

Let $r^+, r^- \leq p_{\mathcal{E}}(\eta)$ such that it maximizes/minimizes δ_r, i.e., $\delta_{r^+} = \max\{\delta_r \mid r \leq p_{\mathcal{E}}(\eta)\}$ and $\delta_{r^-} = \min\{\delta_r \mid r \leq p_{\mathcal{E}}(\eta)\}$. Note that $|\delta_{r^*}| = \max\{|\delta_{r^+}|, |\delta_{r^-}|\}$. To simplify the proof, we show that both $|\delta_{r^+}|$ and $|\delta_{r^-}|$ are negligible (as functions in η), which implies that $|\delta_{r^*}|$ is negligible. Since $|\delta_{r*}| \geq |\delta_r|$ for all r, this concludes the proof of the lemma.

First, we show that $|\delta_{r^+}|$ is negligible. Let q be a polynomial. We define D_q^+, which is parameterized by q, as follows:

1. For every $r \leq p_{\mathcal{E}}(\eta)$ and $i \leq q(\eta)$, D_q^+ simulates a run of $(\mathcal{H}_r \,|\, \mathcal{P})(1^\eta)$ and checks if the event $B^{q_1, \neq \pi^{-1}(r)}_{\mathcal{H}_r \,|\, \mathcal{P}}(1^\eta)$ occurs. Note that with negligible probability this run might not be polynomial time, in which case D_q^+ aborts the run. Without loss of generality and for simplicity of presentation, we ignore such runs in the following. Let $P_i^r \in \{0,1\}$ be 1 if the event occurred and 0 otherwise. Similarly, D_q^+ simulates runs of $(\mathcal{H}_r \,|\, \mathcal{Q})(1^\eta)$ and obtains the results $Q_i^r \in \{0,1\}$.

2. For every $r \leq p_{\mathcal{E}}(\eta)$, D_q^+ computes:

$$\tilde{\delta}_r := \frac{1}{q(\eta)} \sum_{i=1}^{q(\eta)} P_i^r - Q_i^r \ .$$

3. D_q^+ computes $\tilde{r}^+$ which maximizes $\tilde{\delta}_r$, i.e., $\tilde{\delta}_{\tilde{r}^+} = \max\{\tilde{\delta}_r \mid r \leq p_{\mathcal{E}}(\eta)\}$. (If there exist multiple such $\tilde{r}^+$, D_q^+ chooses one arbitrarily.)

4. D_q^+ simulates $\mathcal{H}_r$ as in the non-uniform case (see the proof of Lemma 27 in Appendix A) with $r = \tilde{r}^+$.

Note that for all $r \leq p_{\mathcal{E}}(\eta)$ (considering $\tilde{r}^+$ as a random variable):

$$\delta_r = \mathrm{Prob}\left[(D_q^+ \,|\, \mathcal{P})(1^\eta) = 1 \mid \tilde{r}^+ = r\right] - \mathrm{Prob}\left[(D_q^+ \,|\, \mathcal{Q})(1^\eta) = 1 \mid \tilde{r}^+ = r\right] \ .$$

Furthermore, it is easy to see that D_q^+ is universally bounded (i.e., $D_q^+ \in \mathrm{Env}(\mathcal{P})$). Since, $\mathcal{P} \cong \mathcal{Q}$, there exists a negligible function f_q such that:

$$f_q(\eta) \geq \left|\mathrm{Prob}[(D_q^+ \mid \mathcal{P})(1^\eta) = 1] - \mathrm{Prob}[(D_q^+ \mid \mathcal{Q})(1^\eta) = 1]\right| \ . \tag{36}$$

For every r, the random variables $P_1^r, \ldots, P_{q(\eta)}^r$ (resp., $Q_1^r, \ldots, Q_{q(\eta)}^r$) are independent and identically distributed, all Bernoulli distributed with success probability $p_{\mathcal{P}}^r := \mathrm{Prob}[B_{\mathcal{H}_r \mid \mathcal{P}}^{q_1, \neq \pi^{-1}(r)}(1^\eta)]$ (resp., $p_{\mathcal{Q}}^r := \mathrm{Prob}[B_{\mathcal{H}_r \mid \mathcal{Q}}^{q_1, \neq \pi^{-1}(r)}(1^\eta)]$). So, the expected value of $\tilde{\delta}_r$ (considered as a random variable where the probability space is over runs of D_q^+) is:

$$\mathrm{E}[\tilde{\delta}_r] = p_{\mathcal{P}}^r - p_{\mathcal{Q}}^r = \delta_r$$

and its variance is:

$$\mathrm{Var}(\tilde{\delta}_r) = \frac{p_{\mathcal{P}}^r(1 - p_{\mathcal{P}}^r) + p_{\mathcal{Q}}^r(1 - p_{\mathcal{Q}}^r)}{q(\eta)} \leq \frac{1}{2q(\eta)} \ .$$

Using Chebyshev's inequality, we obtain that for every $r \leq p_{\mathcal{E}}(\eta)$ and $\varepsilon > 0$:

$$\mathrm{Prob}\left[\left|\tilde{\delta}_r - \delta_r\right| \geq \varepsilon\right] \leq \frac{\mathrm{Var}(\tilde{\delta}_r)}{\varepsilon^2} \leq \frac{1}{2q(\eta)\varepsilon^2} \ .$$

Let p be a polynomial. Then, choosing $\varepsilon := \frac{1}{p(\eta)}$ and $q(\eta) := \frac{p^3(\eta)}{2}$ yields for all r:

$$\mathrm{Prob}\left[\left|\tilde{\delta}_r - \delta_r\right| \geq \frac{1}{p(\eta)}\right] \leq \frac{1}{p(\eta)} \ . \tag{37}$$

Let B_p be the event that in a run of D_q^+ (with some system) there exists an $r \leq p_{\mathcal{E}}(\eta)$ such that $|\tilde{\delta}_r - \delta_r| \geq \frac{1}{p(\eta)}$ (i.e., the sampling is off by too much). By $\overline{B_p}$ we denote the complement of B_p, i.e., that for all $r \leq p_{\mathcal{E}}(\eta)$: $|\tilde{\delta}_r - \delta_r| < \frac{1}{p(\eta)}$. With this, we obtain that:

$$\mathrm{Prob}\left[B_p\right] \leq \sum_{r=1}^{p_{\mathcal{E}}(\eta)} \mathrm{Prob}\left[\left|\tilde{\delta}_r - \delta_r\right| \geq \frac{1}{p(\eta)}\right] \overset{(37)}{\leq} \frac{p_{\mathcal{E}}(\eta)}{p(\eta)} \ . \tag{38}$$

It is easy to see that if B_p does not hold, then $|\delta_{\tilde{r}^+} - \delta_{r^+}| < \frac{2}{p(\eta)}$ where $\tilde{r}^+$ is the value computed by D_q^+ in step 3. That is, it holds that

$$\mathrm{Prob}\left[|\delta_{\tilde{r}^+} - \delta_{r^+}| \geq \frac{2}{p(\eta)} \wedge \overline{B_p}\right] = 0 \ . \tag{39}$$

where, as usual, $\tilde{r}^+$ is a random variable and the probability space is taken over runs of D_q^+.

By (39), we conclude that $\mathrm{Prob}\left[\tilde{r}^+ = r \mid \overline{B_p}\right] = 0$ for all $r \leq p_{\mathcal{E}}(\eta)$ such that $|\delta_r - \delta_{r^+}| \geq \frac{2}{p(\eta)}$. Hence, for all $r \leq p_{\mathcal{E}}(\eta)$:

$$\mathrm{Prob}\left[\tilde{r}^+ = r \mid \overline{B_p}\right] \cdot |\delta_r - \delta_{r^+}| < \frac{2}{p(\eta)} \ . \tag{40}$$

For all $r \leq p_{\mathcal{E}}(\eta)$, we define:

$$\delta'_r := \mathrm{Prob}\left[(D_q^+ \mid \mathcal{P})(1^\eta) = 1 \mid \overline{B_p} \wedge \tilde{r}^+ = r\right] - \mathrm{Prob}\left[(D_q^+ \mid \mathcal{Q})(1^\eta) = 1 \mid \overline{B_p} \wedge \tilde{r}^+ = r\right] \ . \tag{41}$$

That is, δ'_r is defined like δ_r but under the condition that B_p does not occur. It is easy to show that for all $r \leq p_{\mathcal{E}}(\eta)$:[43]

$$|\delta'_r - \delta_r| \leq 2 \cdot \mathrm{Prob}\left[B_p\right] \ . \tag{42}$$

[43] We use the obvious fact that for $\mathrm{Prob}[C] > 0$ it holds that $|(\mathrm{Prob}[A] - \mathrm{Prob}[B]) - (\mathrm{Prob}[A \mid C] - \mathrm{Prob}[B \mid C])| \leq 2 \cdot \mathrm{Prob}\left[\overline{C}\right]$.

We conclude that:

$$
\begin{aligned}
f_q(\eta) &\overset{(36)}{\geq} \left|\mathrm{Prob}[(D_q^+ \mid \mathcal{P})(1^\eta) = 1] - \mathrm{Prob}[(D_q^+ \mid \mathcal{Q})(1^\eta) = 1]\right| \\
&\geq \left|\mathrm{Prob}\left[(D_q^+ \mid \mathcal{P})(1^\eta) = 1 \mid \overline{B_p}\right] - \mathrm{Prob}\left[(D_q^+ \mid \mathcal{Q})(1^\eta) = 1 \mid \overline{B_p}\right]\right| - 2 \cdot \mathrm{Prob}\left[B_p\right] \\
&\overset{(41)}{=} \left|\sum_{r=1}^{p_{\mathcal{E}}(\eta)} \mathrm{Prob}\left[\tilde{r}^+ = r \mid \overline{B_p}\right] \cdot \delta'_r\right| - 2 \cdot \mathrm{Prob}\left[B_p\right] \\
&\overset{(42)}{\geq} \left|\sum_{r=1}^{p_{\mathcal{E}}(\eta)} \mathrm{Prob}\left[\tilde{r}^+ = r \mid \overline{B_p}\right] \cdot \delta_r\right| - 2p_{\mathcal{E}}(\eta) \cdot \mathrm{Prob}\left[B_p\right] - 2 \cdot \mathrm{Prob}\left[B_p\right] \\
&= \left|\delta_{r^+} + \sum_{r=1}^{p_{\mathcal{E}}(\eta)} \mathrm{Prob}\left[\tilde{r}^+ = r \mid \overline{B_p}\right] \cdot (\delta_r - \delta_{r^+})\right| - 2p_{\mathcal{E}}(\eta) \cdot \mathrm{Prob}\left[B_p\right] - 2 \cdot \mathrm{Prob}\left[B_p\right] \\
&\geq |\delta_{r^+}| - \left(\sum_{r=1}^{p_{\mathcal{E}}(\eta)} \mathrm{Prob}\left[\tilde{r}^+ = r \mid \overline{B_p}\right] \cdot |\delta_r - \delta_{r^+}|\right) - 2p_{\mathcal{E}}(\eta) \cdot \mathrm{Prob}\left[B_p\right] - 2 \cdot \mathrm{Prob}\left[B_p\right] \\
&\overset{(40)}{>} |\delta_{r^+}| - p_{\mathcal{E}}(\eta) \cdot \frac{2}{p(\eta)} - 2p_{\mathcal{E}}(\eta) \cdot \mathrm{Prob}\left[B_p\right] - 2 \cdot \mathrm{Prob}\left[B_p\right] \\
&\overset{(38)}{\geq} |\delta_{r^+}| - \frac{2p_{\mathcal{E}}^2(\eta) + 4p_{\mathcal{E}}(\eta)}{p(\eta)} \quad .
\end{aligned}
$$

Since this holds for every polynomial p and f_q is negligible (recall that $q = \frac{p^3}{2}$), we conclude that $|\delta_{r^+}|$ is negligible.

To prove that $|\delta_{r^-}|$ is negligible, in the argument above, we simply have to replace D_q^+ by D_q^-, r^+ by r^-, and $\tilde{r}^+$ by $\tilde{r}^-$. D_q^- only differs from D_q^+ in step 3: D_q^- computes $\tilde{r}^-$ such that it minimizes $\tilde{\delta}_r$. □

C Problems with the Composition Theorem in the UC model

This section proves the claim from Section 11.2 that the composition theorem of the 2013 version of the UC model does not hold true even if one were to require different SIDs for every layer of a protocol.

Recall from Section 11.2 that the 2013 version of the UC model disallows the environment from sending any inputs in the name of the challenge session (cf. [4, p. 35]). One of the implications of this restriction is that the environment cannot simulate a higher level protocol in the same session as a subroutine, which in turn invalidates the composition theorem, as illustrated in Section 11.2. A straightforward idea to try to fix this is to limit the composition theorem to protocols that use different SIDs for every protocol layer such that the environment does not need to be able to simulate higher level protocols in the same session. However, the theorem does not work even for this restricted type of protocol, as explained next.

On a high level, this is due to the way protocols send outputs to the environment in the 2013 UC model. A protocol is not allowed to directly send output to the environment; instead, if the environment sends some input in the name of an instance, then that instance is registered and every future output for that instance is redirected to the environment instead (cf. [4, p. 35]). An environment can (ab)use this to register subroutine instances of the protocol *as long as they are running in a different session*[44] and then receive all outputs that were supposed to be sent to those subroutines (see Appendix D for more details). Because the challenge session is protected from this attack, the environment might not be able to mount it on some protocol π when it is analyzed in isolation. However, this attack might be possible when using π as a subroutine for another protocol ρ if ρ uses π with a different SID, leading to a potential security breach. Thus, intuitively,

[44] This is due to the restriction imposed on environments that they may not send any inputs in the name of the challenge session. Thus, they cannot register any instances in the challenge session.

the security guarantees obtained when analyzing π in isolation do not necessarily carry over to the setting where π is a subprotocol.

To illustrate this issue, let us give a simple concrete example where the composition theorem fails (cf. [4, p. 37 and p. 48] for the formal definitions of the realization relation and composition theorem): We construct three protocols π, ϕ, and ρ such that π realizes ϕ but ρ^{π} does not realize ρ^{ϕ}, even though all layers of these protocols use different SIDs. The protocol ϕ is defined as follows: Once it receives some input, it activates a subroutine ϕ' with a different SID and input 1 and waits to receive output from ϕ'. Once it has received the output, ϕ outputs 1 to the environment. All other messages are ignored, including all communication on the network. The subroutine ϕ' expects to receive some input and, upon receiving one, immediately returns 1; all other messages are ignored by ϕ'. The protocol π is identical to ϕ except that it uses a subroutine π'. The protocol π' is the same as ϕ' but returns 0 instead of 1. The protocol ρ is just a dummy protocol that expects some input, forwards it to a subroutine instance of π/ϕ (with a different SID), waits for the output, and returns the output to the environment. Again, all other messages are ignored.

We obviously have that π realizes ϕ: both protocols, including their subroutines, are identical in their behavior, except that π' and ϕ' produce different outputs. However, these outputs are never forwarded by π/ϕ and, because π and ϕ run in the challenge session when they are analyzed, the environment also cannot redirect these outputs. So overall, there is no way for the environment to distinguish the two protocols.

However, the situation is quite different when π/ϕ are used as subroutines of ρ. Assume that ρ runs in the challenge session *sid*, its direct subroutine π/ϕ runs in session sid', and the subroutine π'/ϕ' runs in session sid''. In this situation, the environment can send an arbitrary input to ρ in the name of the machine π in session sid'. Note that this is possible because $sid \neq sid'$ by construction, i.e., the environment may act in the name of the subroutine π as it does not run in the challenge session. From that point onwards, every output sent to π in session sid' will be redirected to the environment instead. Now, if the environment is running with ρ^{π}, the following happens: Once ρ receives the initial message, it forwards it to π which will then activate π'. The subroutine π' immediately returns 0 as output to π, however, as mentioned, this output will be redirected to the environment. So the environment receives 0 as a response from some instance in session sid''. In contrast, if the environment is running with ρ^{ϕ}, the following happens: Once ρ receives the initial message, it forwards it to ϕ which will then activate ϕ'. The subroutine ϕ' immediately returns 1 as output to ϕ, which *is not redirected* as the environment has sent an input only in the name of an instance of π, not ϕ. So ϕ will output 1 to ρ, which forwards the output to the environment. Overall, in this setting the environment receives the output 1 from an instance in session *sid*. So the environment can easily distinguish between the real world and the ideal world, which means that the composition fails.

D Model Specific Distinguishing Attacks in the UC Model

As mentioned in Section 11.2, the 2013 version of the UC model [4] (and partly also older versions) allows for several artificial distinguishing attacks that are specific to the model. This section explains these attacks and illustrates why they cause problems. Again, we emphasize that while dealing with distinguishing attacks is an important part of both protocol design and security analysis, the attacks from this section do not relate to any attack scenarios in reality and thus only hinder a protocol designer. They are hard to deal with, lead to unnatural protocol specifications and complex security proofs, and can prevent intuitive realizations, or in fact realizations at all when standard conventions (such as the structure of ideal protocols and corruption mechanisms) from the UC model are used, as explained in what follows.

Exhausting machines. As explained in Section 11.1, the UC model bounds the runtime of instances of machines by a polynomial in the number of bits they receive from higher-level protocols minus the number of bits they send to subroutines (cf. [4, p. 30]). This allows the environment to exhaust instances in the real protocol by sending several messages on the network.

This is quite troublesome, as the real protocol now has to deal with blocked instances and might not be able to do so. In fact, if there is any way for the environment to detect whether a real protocol is blocked (for example, because the protocol usually generates some output after receiving a message from the network),

then such a real protocol *can hardly realize any of the ideal protocols from the literature*. This is because ideal protocols generally do not tell the simulator the number of input bits (including malformed inputs) they have received, so the simulator cannot calculate the runtime bound of the real protocol. One can only fix this issue by leaking the length of all inputs from the ideal functionality to the simulator, which generally is undesirable and has not been done in the literature so far. In particular, leaking the length of the input is not always viable, as some tasks require this information to be secret.

Preventing interaction with subroutines. The environment can block a real protocol from interacting with some of its subroutines. The severity of this problem strongly depends on the protocols at hand. While a protocol designer might be able to deal with this problem for some protocols, in other cases one might not be able to change a protocol such that it can handle missing subroutines. Either way, at the very least, this problem drastically complicates protocol specifications and security analyses as protocol designers have to consider and deal with many annoying details. There are two different ways for the environment to prevent interaction with subroutines.

The first possibility is due to the runtime notion. Recall that in the UC model the runtime of an instance is bounded by a polynomial $p(n)$ where $n = n_I - n_O$, n_I is the number of bits received from higher-level protocols, and n_O is the number of bits sent to subroutines. Also recall that higher level protocols have to send the security parameter to new instances of subroutines (cf. [4, p. 30f]). In other words, the security parameter for each subroutine instance is included in n_O. Thus, if the environment sends a very short input to the challenge protocol, say the security parameter 1^η and some constant number of bits, then the challenge protocol can invoke at most one subroutine instance: in order to invoke a second subroutine, the challenge protocol would have to provide this subroutine with at least 1^η again, which, however, is not possible anymore. Clearly, the environment can use this to influence the number of subroutines a protocol can create.[45]

The second possibility is due to the ability of the adversary to create new machines and determine their machine code. Recall that the adversary specifies both the ID and the machine code of the receiver to send a message. If there is some instance with that ID already, then the message will be delivered to that instance regardless of its machine code. However, if there is no instance with that ID yet, then a new one will be created that executes the code provided by the adversary (cf. [4, p. 19f, 35]). Thus, the adversary can, at the start of the run before any instances of the challenge protocol exist, create instances with IDs that belong to subroutines that will be called by the challenge protocol. While the challenge protocol will notice that its subroutines do not exist (it receives a special error when its subroutine uses the wrong machine code), the protocol execution is disrupted as now the protocol has to deal with a missing subroutine.

Intercepting outputs from subroutines. The environment can intercept the outputs from subroutines even though these outputs are meant to be sent directly on a protected channel to a higher-level protocol.

For this attack, recall from the UC model (see also Section 11.2) that an environment can send inputs to the highest level of the challenge protocol and claim that it was sent by an arbitrary sender, as long as that sender has a different SID than the challenge protocol. The identities of those senders are stored and if some instance of the protocol sends some output to those identities, the output is redirected to the environment instead (cf. [4, p. 35]). This is a problem if the challenge protocol uses subroutines that have a different SID than the highest-level instances, as the environment can redirect all outputs sent to these subroutines to itself.[46] To illustrate this, consider the following example: The challenge protocol π uses two

[45] We note that one mechanism to try to deal with this issue is to expect a sufficiently long (padded) initial input in the real protocol such that all subroutines are accessible (and block the real protocol until sufficiently many bits have been received). However, this approach is not generally applicable as it works only if (i) the number of subroutine instances is known at the start of a run and (ii) the simulator is able to block an uncorrupted party in the ideal functionality if the initial input was too short. Furthermore, padding of inputs is usually not done in the literature as it yields very unnatural protocol specifications and is an annoying technical detail.

[46] The following attack requires the environment to send some input in the name of a subroutine. As the environment may only do so for instances that have a different SID as the challenge session, this attack does not work for subroutines with the same SID. Note, however, that as explained in Section 11.2 this restriction imposed on the environment actually causes the composition theorem to fail. If this restriction is removed to fix the composition theorem, then the environment will be able to intercept outputs even from subroutines with the same SID as the challenge session.

ideal protocols ϕ and ϕ' (which consist of a dummy and an ideal functionality) as subroutines. By convention, these subroutines have different SIDs as otherwise the IDs of the ideal functionalities would coincide; w.l.o.g. let π and ϕ have SID sid, and let ϕ' have SID sid'. Now, the environment can send an input in the name of a dummy instance of ϕ' to the challenge protocol to register this ID. Any future output sent by the ideal functionality of ϕ' to the dummy instance will then be redirected to the environment instead.

Again, the severity of this attack depends on the protocols at hand. In many cases, the integrity of the real protocol π is disrupted if one can eavesdrop on direct connections between this protocols and its subroutines, allowing the environment to distinguish the real protocol from the ideal one. For example, if the subroutine ϕ' from above is a secure channel, then the environment might be able to acquire a secret key sent via this channel and use this key to distinguish the real from the ideal world. Even if eavesdropping is not a concern, the higher-level protocol still has to deal with the possibility that it might not receive any outputs from subroutines, even when they are supposed to immediately and always provide one.

Prevent interaction with the ideal functionality The environment can essentially "delete" the ideal functionality, thus forcing the simulation to fail as the simulator receives no inputs and cannot corrupt any parties. One version of this attack, which we describe in the following, can even be used to *distinguish any real protocol from any ideal protocol* if standard UC conventions (structure of ideal functionalities and in particular corruption mechanisms) are used.

On a high level, the attack works as follows. Recall that the environment can send inputs to arbitrary challenge protocol instances with some ID (pid, sid). If no instance with that ID exists yet, a new instance of the challenge protocol is created (cf. [4, p. 19f, 35]). Thus, if the environment sends its first input to the instance with ID $(\bot, sid)$, then this will create a new dummy with this ID in the ideal world. As this ID is now bound to the instance of a dummy, the dummy cannot access its ideal functionality: this is because in the UC model the convention is that for an instance of a dummy with an ID of the form (pid, sid) the corresponding ideal functionality has ID $(\bot, sid)$, which now, however, is taken by the dummy instance. Hence, an input sent by the dummy to the ideal functionality would result in an error of the write command, which is then forwarded to the environment (cf. [4, p. 40]). This already allows the environment to distinguish many real protocols from ideal protocols. Some real protocols might also produce an error in the same situation, though. We can distinguish even these protocols by using the standard corruption mechanism to let the real protocol send a notification to the environment, which cannot be done by the simulator. In what follows, this is made more precise.

We first define the behavior of the environment when running with the dummy adversary and then show that no simulator can simulate a real protocol that uses any of the corruption mechanisms (static, adaptive, honest-but-curious) defined in the 2013 version of the UC model. Let the environment $\mathcal{E}$ be such that it activates the adversary at the start of the run (this is required by the model, cf. [4, p. 35]). As soon as $\mathcal{E}$ regains control, it sends some input to the (ideal/real) challenge protocol instance with ID $(\bot, sid)$ for some random sid of length η (where η is the security parameter). When $\mathcal{E}$ regains control, it corrupts the instance $(\bot, sid)$ (an instance with an ID of the form (pid, sid) for any pid would work as well) and waits to receive a confirmation (recall that, by convention, real protocols send a confirmation to all higher level protocols upon corruption; cf. [4, p. 58]). If the whole run proceeds as expected, $\mathcal{E}$ outputs 0; otherwise, it outputs 1.

First note that in the real world, $\mathcal{E}$ will always output 0. Now, assume, by contradiction, that there is a simulator $\mathcal{S}$ such that the real protocol is indistinguishable from the ideal protocol. First observe that $\mathcal{S}$ does not know the SID sid that $\mathcal{E}$ will use at the start of the run, so there is at most a negligible probability that $\mathcal{S}$ has already activated any instances in session sid before $\mathcal{E}$ sends the first input. Thus, we can ignore this negligible set of runs in the following. When $\mathcal{E}$ sends the first input to the ideal challenge protocol, a fresh dummy instance with ID $(\bot, sid)$ will be created. By definition, the dummy will try to forward its input to its ideal functionality with ID $(\bot, sid)$, which fails since the instance with this ID is a dummy instance, not an instance of the ideal functionality. The dummy returns an error to the environment in this case. Now the environment tries to corrupt the instance $(\bot, sid)$ and expects to receive a confirmation. The simulator cannot simulate this in the ideal world: by convention, dummies only forward output from the ideal functionality with ID $(\bot, sid)$ while ignoring corruption messages from the simulator. Instead, the instance $(\bot, sid)$ of the

ideal functionality is supposed to handle corruption (cf. [4, p. 61]). However, such an instance does not and cannot exist. Thus, the simulator cannot send any messages via the dummy and is unable to simulate the behavior of the real world, allowing the environment to distinguish the real from the ideal world.

Note that this argument does not assume anything other than that standard UC conventions are used and thus affects virtually all of the current UC literature. We also note that this attack cannot be prevented by restricting the environment to never send input to an instance with ID $(\perp, sid)$. Such a restricted environment can, at the start of the run, advise the dummy adversary to create an instance with $(\perp, sid)$ but different machine code, similar to what is described in the paragraph "prevent interaction with subroutines". In order to simulate the behavior of this instance (with a machine code that is determined by the environment) in the ideal world, the simulator will generally be forced to also create a similar instance with the same ID, thus overwriting the ideal functionality. The same attack from above then still works.