Universal Composition with Responsive Environments*

Jan Camenisch[1], Robert R. Enderlein[1,2], Stephan Krenn[3], Ralf Küsters[4], Daniel Rausch[4]

[1] IBM Research – Zurich, Rüschlikon, Switzerland
{jca,enr}@zurich.ibm.com
[2] Department of Computer Science, ETH Zürich, Zürich, Switzerland
[3] AIT Austrian Institute of Technology GmbH, Vienna, Austria
stephan.krenn@ait.ac.at
[4] University of Trier, Trier, Germany
{kuesters,rauschd}@uni-trier.de

Abstract. A increasingly popular approach to proving the security of protocols is to define the desired security and functional properties by an ideal functionality and then to prove that a protocol realizes the functionality within a universal composability framework. When specifying such ideal functionalities, one often requires the adversary (or environment) to provide some meta-information, such as cryptographic values of signatures, ciphertexts, and keys. Similarly, when designing protocols, the adversary/environment needs to provide, for example, signaling information and corruption statuses of protocol participants. Intuitively, one would expect that such requests are answered immediately. However, in none of the existing models for universal composability this is guaranteed: adversaries and environments can freely activate protocols and ideal functionalities without answering such requests, resulting in dangling and interleaving requests. We call this issue the non-responsiveness problem. It is typically very cumbersome to properly deal with such intermediate activations and interleaved requests and there is no generally applicable method to handle such activations. If fact, protocol designers often do not even consider this issue and miss to specify the behavior of their protocols and ideal functionalities in these situations. This unfortunately results in undefined or even flawed specifications, making it impossible to use such protocols/ideal functionalities in higher level protocols and carrying out rigorous security proofs. What makes the non-responsiveness problem and its consequences particularly disturbing is that they are merely a modeling artifact: it would be very natural if the mentioned requests were answered immediately by adversaries/environments as they are used for modeling purposes only and allowing adversaries/environments to not answer them immediately does not model any real attack. This paper solves the non-responsiveness problem and its negative consequences by proposing a framework for universal composability with responsive environments and adversaries. In a nutshell, when a protocol or functionality sends what we call a restricting message to the adversary/environment, the latter must provide a valid response before any other protocol/functionality is activated. Hence, protocol designers can declare requests for meta-information to be restricting in order to guarantee that such requests are answered immediately, and hence, they do not have to worry about modeling artefacts resulting from such requests not being answered immediately. Our concepts apply to all existing models for universal composability, we provide formal theorems for the IITM model and discuss it the UC and GNUC models.

Keywords: universal composability; protocol design; cryptographic security proofs; responsive environments

* This project was in part funded by the European Commission through grant agreements n°s 321310 (PERCY) and 644962 (PRISMACLOUD), and by the *Deutsche Forschungsgemeinschaft* (DFG) through Grant KU 1434/9-1.

Table of Contents

1 Introduction

One of the most demanding tasks when designing a cryptographic protocol is to define its intended security guarantees, and to then prove that it indeed satisfies them. In the best case, these proofs should guarantee the security of the protocol in arbitrary contexts, i.e., also when composed with other, potentially insecure, protocols. This would allow one to split complex protocols into smaller components, which can then be separately analyzed one by one and once and for all, thus allowing for a modular security analysis. Over the last two decades, many models to achieve this goal have been proposed [3, 7–10, 18, 21, 22, 25–27].

What all these models have in common is that the protocol designer first needs to specify an *ideal functionality* $\mathcal{F}$ defining the intended security and functional properties of the protocol. Informally, a *real protocol* now realizes $\mathcal{F}$ if no efficient distinguisher (the *environment*) can decide whether it is interacting with the ideal functionality and a *simulator*, or with the real world protocol and an *adversary*.

In the specifications of such real protocols and ideal functionalities, it is often required for the adversary/environment to provide to the protocol or functionality some (meta-)information or to receive some information. Examples for such meta-information are cryptographic values, such as keys, signatures, and ciphertexts, or information about the (initial) corruption status and the existence of a machine. Intuitively, protocol designers would expect the adversary/environment to return answers to such requests immediately. In the sequel, we call such requests *urgent*. However, none of the existing models guarantee immediate answers. We call this the *non-responsiveness* problem.

Urgent request, and hence, the non-responsiveness problem occurs in many functionalities from the literature, see, e.g., [1,4,5,7,11–13,15,16,20,23,24,28]. In order to properly take into account the non-responsiveness to urgent requests, protocol designers have to deal with the problem that protocols/functionalities can be manipulated before the adversary/environment answers the urgent request. In particular, other parties or parts of the protocol/ideal functionality can be activated in the meantime, which might change their state, even their corruption status, and which can result in many dangling and interleaving requests. Furthermore, this unexpected behavior of the adversary also needs to be considered in the design of every higher level protocol using such a protocol as a subroutine. This makes it hard to deal with the non-responsiveness problem, and as explained in more detail in §3, there is no generally applicable way to cope with unanswered urgent requests. Even worse, sometimes protocol designers simply forget about or ignore the non-responsiveness problem altogether, resulting in protocols and functionalities with unspecified and/or ill-defined behavior.

Most disturbingly, the non-responsiveness problem forces protocol designers to take care of adversarial behavior that was unintended in the first place and that does not translate to any concrete attacks in reality, as urgent requests are used for modeling purposes, rather than reflecting real network communication.

Our contribution. In this paper, we propose a universal composability framework with, what we call, *responsive environments* and *adversaries*. This framework completely circumvents the above problem as it guarantees that urgent requests are answered immediately. More specifically, when a protocol/ideal functionality sends what we call a *restricting* message, then the adversary and environment are forced to be responsive, i.e., reply with a valid response before sending any other message to the protocol. This requires careful definitions and non-trivial proofs in order to make sure that all properties and features that are expected in models for universal composition are lifted to the case of responsive environments and adversaries.

By using our framework, protocols and ideal functionalities can be modeled in a very natural way: protocol designers can simply declare urgent requests to be restricting messages. This not only greatly simplifies specifications of protocols and ideal functionalities, but also security proofs as the protocol designer does not have to consider irrelevant execution orders. Of course, the designer must exercise discretion when employing restricting messages: such messages should be employed for meta-information used for modeling purposes only, not for real network traffic, where immediate answers cannot be guaranteed.

We illustrate that our framework and concepts apply to existing models for universal composability. This is exemplified for three prominent models: UC [7, 8], GNUC [18], and IITM [21, 22]. In the appendix, we provide full proofs for the IITM model. In particular, we define all common notions of simulatability, including UC, dummy UC, strong simulatability, and blackbox simulatability with respect to responsive environments and adversaries, and show that all of these notions are equivalent. This result can be seen as a sanity check of

our concepts, as it has been a challenge in previous models (see, e.g., discussions in [19,22]). We also prove in detail that all composition theorems from the original IITM model carry over to the IITM model with our concepts.

Related work. In the first version of his seminal work [7], Canetti introduced *immediate functionalities*. According to the definition (cf. page 35 of the 2001 version), an immediate functionality uses an *immediate adversary* to guarantee that messages are delivered immediately between the functionality and its dummy. To be more precise, an immediate functionality may force an immediate adversary to deliver a message to a specific dummy party within a single activation. This construct was necessary since in the initial version of Canetti's model the ideal functionality could not directly pass an output to its dummy but had to rely on the adversary instead. Clearly, immediate adversaries do not address the non-responsiveness problem. Also, we note that in current versions of UC, the problem addressed by immediate adversaries has vanished completely because ideal functionalities can directly communicate with their dummies. Still, this does not address, let alone solve, the non-responsiveness problem.

In [2], composition for restricted classes of environments is studied, motivated by impossibility results in UC frameworks and to weaken requirements on realizations of ideal functionalities. In this setting, environments are restricted in that they may send only certain sequences of messages to the I/O interfaces of protocols and functionalities. These restrictions cannot express that urgent requests are answered immediately and also these restrictions do not restrict adversaries in any way. Hence, this approach cannot be used to solve the non-responsiveness problem.

Outline. In §2, we briefly recall basic terminology and notation. We observe in §3 that the non-responsiveness problem affects many protocols from the literature, with some papers ignoring the problem altogether, resulting in underspecified, ill-specified, and/or very complex functionalities that cannot easily, if at all, be reused in higher-level protocols. Furthermore, that section shows that properly taking this problem into consideration is quite hard and does not have a simple and generally applicable solution. Our universal composability framework with responsive environment and adversary is then presented in §4. This section is kept quite model independent in order to highlight the main new concepts and the fact that these concepts are not restricted to specific models. §5 then illustrates how our concepts can be implemented in the UC, GNUC, and IITM models. §6 shows how the problems with non-responsive environment and adversaries discussed in §3 can be circumvented elegantly with our restricting messages and responsive environments/adversaries. We conclude in §7. Further details can be found in the appendix. In particular, as mentioned before, for the IITM model with responsive environments and adversaries we provide full details in the appendix.

2 Preliminaries

In this section, we briefly recap the basic concepts of universal composability and fix some notation and terminology. The description is independent of the model being used and can easily be mapped to any concrete model, such as UC, GNUC, or IITM. We, for now, ignore runtime issues as they are handled differently in the models and only implicitly assume that all systems run in polynomial time in the security parameter and the length of the external input (if any). Runtime issues are discussed in detail in §5.

Universal composability models use *machines* to model programs. Each machine may have I/O and network tapes/interfaces. These machines are then used as blueprints to create *instances* which execute the machine code while having their own local state. Machines can be combined to a *system* $\mathcal{S}$. In a run of $\mathcal{S}$, multiple instances of machines may be generated and different instances can communicate by sending messages via I/O or network interfaces. Given two systems $\mathcal{R}$ and $\mathcal{Q}$, we define the system $\{\mathcal{R}, \mathcal{Q}\}$ which contains all machines from $\mathcal{R}$ and $\mathcal{Q}$.

There are three different kinds of entities, which can themselves be considered as systems, and which can be combined to one system: *protocols*, *adversaries*, and *environments*. One distinguishes *real* and *ideal* protocols, where ideal protocols are often called *ideal functionalities*. An ideal protocol can be thought of as the specification of a task, while a real protocol models an actual protocol that is supposed to realize the ideal

protocol (cf. Definition 2.1). These protocols have an I/O interface to communicate with the environment and a network interface to communicate with the adversary. An *adversary* controls the network communication of protocols and can also interact with the environment. *Environments* connect to the I/O interface of protocols and may communicate with the adversary, cf. Figure 1 for an illustration of how environments, adversaries, and protocols are connected.

Environments try to distinguish whether they run with a real protocol and an adversary or an ideal protocol and an adversary (then often called a simulator or ideal adversary). An environment may get some external input to start a run. It is expected to end the run by outputting a single bit.

Given an environment $\mathcal{E}$, an adversary $\mathcal{A}$, and a protocol $\mathcal{P}$, we denote both the combined system and the output distribution of the environment by $\{\mathcal{E}, \mathcal{A}, \mathcal{P}\}$. We use the binary operator $\equiv$ to denote two output distributions that are negligibly close in the security parameter η (and the external input, if any).

Now, in models for universal composability the realization of an ideal protocol by a real protocol is defined as follows.

Definition 2.1 (Realization Relation). *Let $\mathcal{P}$ and $\mathcal{F}$ be protocols, the* real *and* ideal protocol*, respectively. Then, $\mathcal{P}$* realizes *$\mathcal{F}$ ($\mathcal{P} \leq \mathcal{F}$), if for every adversary $\mathcal{A}$ there exists an ideal adversary $\mathcal{S}$ such that $\{\mathcal{E}, \mathcal{A}, \mathcal{P}\} \equiv \{\mathcal{E}, \mathcal{S}, \mathcal{F}\}$ for every environment $\mathcal{E}$.*

We note that, in the definition above and in all reasonable models, instead of quantifying over all adversaries, it suffices to only consider the dummy adversary $\mathcal{A}_D$, which forwards all network messages between $\mathcal{P}$ and $\mathcal{E}$. Intuitively, this is true because $\mathcal{A}$ can be subsumed by $\mathcal{E}$. Hence, we have that $\mathcal{P} \leq \mathcal{F}$ iff there exists an ideal adversary $\mathcal{S}$ such that $\{\mathcal{E}, \mathcal{A}_D, \mathcal{P}\} \equiv \{\mathcal{E}, \mathcal{S}, \mathcal{F}\}$ for every environment $\mathcal{E}$.

The main result in any universal composability model is a composition theorem. Informally, once a protocol $\mathcal{P}$ has been proven to realize an ideal protocol $\mathcal{F}$, one can securely replace $\mathcal{F}$ by $\mathcal{P}$ in arbitrary higher-level systems without affecting the security of the higher-level system.

3 The Non-Responsiveness Problem and its Consequences

As pointed out in the introduction, specifications of protocols and ideal functionalities very often contain what we call *urgent requests*. These are requests sent on the network interface of a protocol/ideal functionality which the protocol designer would want the adversary/simulator to answer immediately. Such requests are used for modeling purposes, rather than reflecting network communication. For example, upon first activation, machines might ask the adversary/simulator whether he wants to corrupt the machine or not. The adversary/simulator might also be asked to provide certain parameters upon initialization, for example, to analyze protocols running in different modes. In ideal functionalities, in particular those that are supposed to model non-interactive tasks, such as digital signatures or public-key encryption, the adversary is typically asked to provide various information right away, such as keys and algorithms upon initialization or signatures for messages given to the adversary.

We emphasize that urgent requests are also used to provide *the adversary* with some information. This is necessary, for example, when modeling honest-but-curious corruption where the adversary is provided with the internal state of the corrupted instance upon every activation, or when the adversary has to be informed about the creation of a fresh instance, as it is required by the current version of the UC model [7] (version 2013) for subroutine respecting protocols.

The problem is that in the existing universal composability models it is not guaranteed that urgent requests are answered immediately, although this would be very desirable and natural; in what follows we refer to this problem as the *problem of non-responsive adversaries/environments* or the *non-responsiveness problem*. Hence, in the existing models, protocol designers have to cope with this problem. In particular, they have to deal with the situation that while a protocol/ideal functionality is waiting for an answer from the adversary to an urgent request the protocol/ideal functionality receives a request itself (e.g., on the I/O interface) that would have to be processed. Can these latter requests simply be ignored? Should one return an error message for such requests? Can they be queued and answered later? As discussed below, there is no general solution. Dealing with the non-responsiveness problem is hard and results in complex protocols and

ideal functionalities. Also, the problem propagates to higher level protocols as they might not get responses from their subprotocols as expected. The security proofs become more complex as well because one has to deal with runs with various dangling and interleaving requests, which, importantly, do not translate to anything in the real world, but are just an artifact of the modeling. Altogether, the non-responsiveness problem often is hard to deal with, results in much more complex protocols, ideal functionalities, and security proofs, and really is an artificial problem.

We note that urgent requests, and hence the non-responsiveness problem, are not just an artifact of a specific modeling choice. They rather seem to be an important mechanism of protocol designs in any UC-like model. Simply getting rid of urgent requests,[5] and by this, solving the non-responsiveness problem, does not seem to be possible at all. Without these requests, the flow of meta-information (i.e., information unrelated to actual network messages) in both directions between the adversary and the protocol would be severely limited. On the one hand, a protocol could no longer ask for any meta-information to which it would expect an immediate response, such as algorithms, cryptographic material, or the corruption status. On the other hand, a protocol also could no longer provide the adversary with meta-information in situations where it does not intend to give control to the adversary, such as certain information leaks (e.g, honest-but-curious corruption). Some models even have urgent requests hardwired into the model itself, such as Canetti's current UC model, which requires new instances of subroutines to inform the adversary of their creation (cf. "subroutine respecting protocols", page 47, 2013 version of [7]).

Looking at the literature, it is apparent that urgent requests and the non-responsiveness problem occur in many protocols and functionalities [1, 4, 5, 7, 11–13, 15, 16, 20, 23, 24, 28]. Even worse, while protocol designers frequently use what we here call urgent requests, they sometimes implicitly assume that they *are* answered immediately. As a result, security proofs may be flawed, protocols and ideal functionalities are underspecified and/or expose unexpected behavior, and thus, are not usable in other (hybrid) protocols.

In the remainder of this section, we first, in §3.1, explain where the misconception that urgent requests are answered immediately might stem from. We discuss in §3.2 that it is very hard to deal with the non-responsiveness problem in specifications of ideal functionalities, with real/hybrid protocols discussed in §3.3. In §3.4, we provide concrete examples from the literature where the non-responsiveness problem led to undesirable complications or was ignored altogether, resulting in underspecified and flawed protocols/functionalities.

3.1 Ideal Functionalities are not Always Protected by a Simulator

The realization relation of an ideal functionality, see Figure 1, and the relation implied by the composition theorem, see Figure 2, both use an ideal functionality $\mathcal{F}$ with a simulator only. This leads to a common misconception where protocol designers simply assume that the ideal functionality's network interface is always controlled by a friendly simulator,[6] which responds to urgent requests immediately.

However, if the ideal functionality is destined to be used as a subroutine of a *hybrid* protocol, then an adversary will have direct access to the functionality, see Figure 3. Although this should be obvious for anyone familiar with universal composability models, many ideal functionalities do not specify their behavior in these cases or behave in a clearly unintended way. In both cases the ideal functionality usually cannot be reused in a hybrid protocol, either because the behavior is not defined, or because the environment might actively trigger ill defined behavior to distinguish hybrid and ideal world, making it impossible to formally prove the realization relation. This is ironic, considering that the whole *raison d'être* of some of those ideal functionalities is to be used in such a hybrid setting.

Similarly, in a joint-state realization relation, where one wishes to realize multiple instances of an ideal functionality via a joint-state protocol that has access to a single instance of a joint-state functionality, the adversary also has direct access to the joint-state ideal functionality. Figure 4 shows such a case.

[5] Recall from above that by this we mean requests sent by a protocol/ideal functionality on the network interface to the adversary/environment to which the protocol designer would expect an immediate answer.

[6] The authors of some papers go as far as writing that the ideal functionality sends messages to the simulator instead of to the adversary or to the network (e.g., [5] writes $\mathcal{SIM}$ and [1] writes "simulator").

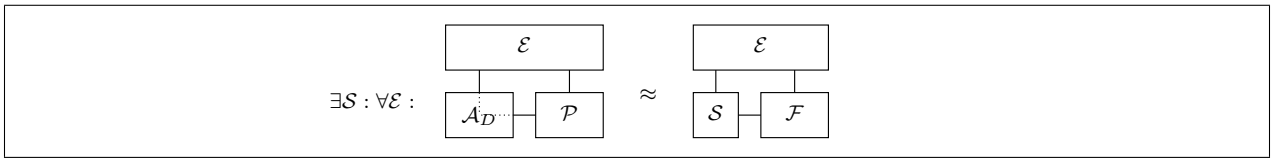

Fig. 1: The realization relation for a real protocol $\mathcal{P}$ realizing an ideal functionality $\mathcal{F}$. Here $\mathcal{A}_D$ denotes the dummy adversary who just forwards messages to and from the environment $\mathcal{E}$.

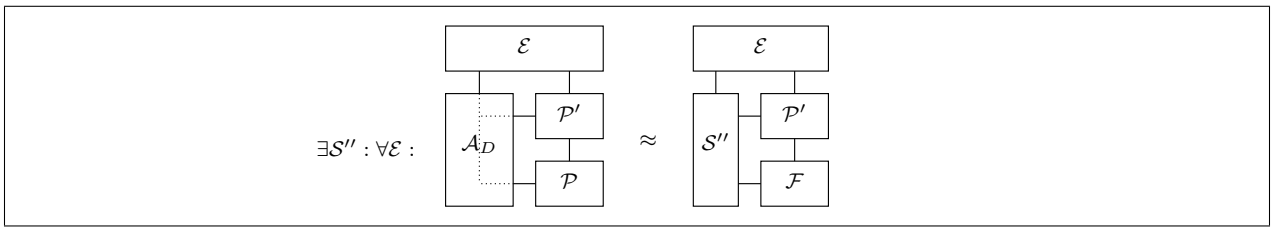

Fig. 2: The composition theorem guarantees that if $\mathcal{P}$ realizes $\mathcal{F}$ (see Figure 1), and that if $\mathcal{P}'$ is a $\mathcal{F}$-hybrid protocol, then $\mathcal{P}'^{\mathcal{F}/\mathcal{P}}$ realizes $\mathcal{P}'$, as shown here. By looking only at this and the previous figure, a protocol designer could mistakenly think that the ideal functionality $\mathcal{F}$ only ever interacts with a (friendly) simulator $\mathcal{S}$ or $\mathcal{S}''$.

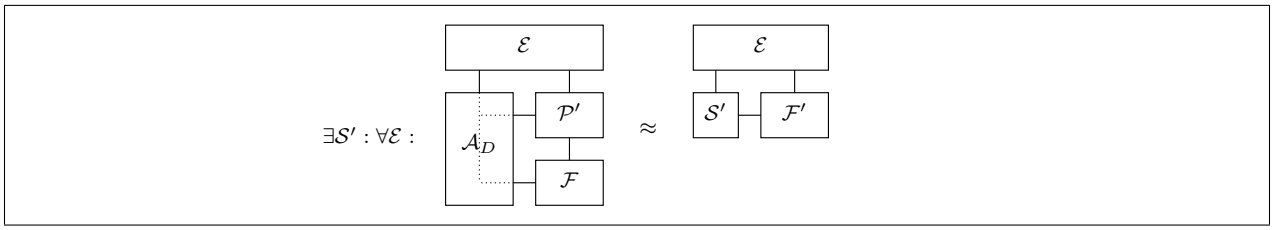

Fig. 3: The realization relation for an $\mathcal{F}$-hybrid protocol $\mathcal{P}'$ realizing some higher-level ideal functionality $\mathcal{F}'$. Notice that the dummy adversary $\mathcal{A}_D$ directly interacts with $\mathcal{F}$, it is thus necessary for the behaviour of $\mathcal{F}$ to be well defined even if urgent requests are not answered immediately.

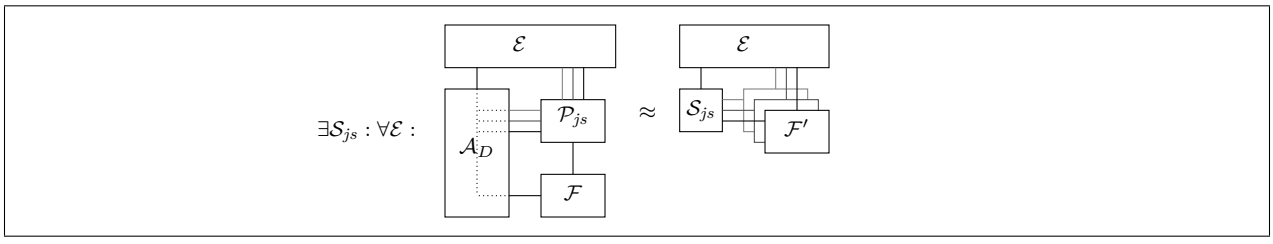

Fig. 4: The realization relation for a joint-state-enabling protocol $\mathcal{P}_{js}$ that realizes multiple instances of $\mathcal{F}'$ from a single joint-state instance of $\mathcal{F}$. Again, notice that the dummy adversary $\mathcal{A}_D$ directly interacts with $\mathcal{F}$, hence also here it is necessary for the behaviour of $\mathcal{F}$ to cover the case that urgent requests are not answered immediately.

3.2 Dealing with the Non-Responsiveness Problem in Ideal Functionalities is Hard

As motivated above, a protocol designer has to specify the behavior of ideal functionalities upon receiving another input (on the I/O interface) while still waiting for a response to an urgent request on the network. In other words, ideal functionalities have to be *reentrant* when waiting for such a response. As explained next, this, however, is hard. Approaches to solve this problem complicate the specification and none of them is sufficiently general to be applied in every case. In fact, a straightforward application of these approaches often yields a functionality that can no longer be realized by a large class of real protocols. Instead one has to adjust these approaches to both the ideal functionality *and* the real protocol, which is not always possible and highly undesirable. By doing this, the definition of the ideal functionality depends on its realization, i.e., if one wants to use a different realization, one also might have to use a different ideal functionality. This limits the usefulness of universal composability.

To illustrate the above points, we consider the following running example for the remainder of this section: Let $\mathcal{F}$ be an ideal functionality which has to send an urgent request to the adversary upon its first creation, say, to retrieve some modeling-related information. This is a common situation. For example, ideal functionalities often require some cryptographic material such as keys and algorithms from the adversary before they can continue their execution. We also assume that $\mathcal{F}$ is meant to be realized by a real protocol consisting of two independent roles A and B. Both of these roles also send an urgent request to the adversary upon their first activation and expect an answer before they can continue with their computation. Again, this is a common situation since, for example, real protocols often ask for their corruption status or notify the adversary of their creation (the latter is required by the definition of "subroutine respecting protocols" in the 2013 version of UC [7]). While the above is only one illustrative example, it already describes a large and common class of real and ideal protocols often encountered in the literature.

We now present several approaches to making $\mathcal{F}$ reentrant in the above sense, i.e., to deal with I/O requests while waiting for a response to an urgent request on the network. We show that the obvious approaches cannot be used in general. In particular, with most of these approaches $\mathcal{F}$ could not even be realized by A and B. This in turn means that solutions tailored to the specific functionality at hand and even the envisioned realization are necessary. This is very unsatisfying, as they lead to more complex and yet less general functionalities and protocols.

Ignore requests. After sending an urgent request to the adversary, the most straightforward approach would be to ignore all incoming messages until a response from the adversary is received. However, this is not only an unexpected behavior in many cases—for example, why should a request silently fail if the ideal functionality models a local computation?—but, as mentioned above, the ideal functionality might not even be realizable by some real protocols anymore.

In our running example, if $\mathcal{F}$ simply ignores incoming messages, an environment can distinguish $\mathcal{F}$ (with a simulator) from the realization A and B (with the dummy adversary). It first sends a message to A which, as we assume, then in turn sends an urgent request to the dummy adversary, and hence, the environment. Now the environment, which does not have to respond to urgent requests immediately, sends a message to B which in turn also sends an urgent request to the adversary, and hence, the environment. Consider the behavior of the ideal world in this case: After receiving the message for A, $\mathcal{F}$ will send an urgent request to the simulator. The simulator, however, cannot answer this urgent request because he has to simulate A by sending an urgent request to the environment. (This might be the case because the simulator has to first consult the environment before answering the urgent request by $\mathcal{F}$ or because $\mathcal{F}$ could not return control to the simulator after receiving an answer to the urgent request.) The environment then sends the second message (for B) to $\mathcal{F}$, which is ignored because $\mathcal{F}$ still waits for an answer to its urgent request. This behavior is different from the real world, and thus, the environment can distinguish the real world from the ideal one.

This illustrates that an ideal functionality that simply blocks *all* requests while waiting for a response to an urgent request can in general not be realized by two or more *independent* roles that also send urgent request to the adversary. This is because a real party cannot know whether it has to block messages because another party is waiting for a response to an urgent request. Often it might be possible to adjust this approach to only block messages that are processed by a single role in the real protocol, while messages for other roles

are still processed. However, this variant is not possible in all cases, for example, if in our running example $\mathcal{F}$ cannot process messages for any party before receiving a response to its urgent request.

Sending an error message. Instead of ignoring and silently dropping subsequent requests, the ideal functionality could instead reply with an error message in an attempt to let the higher-level protocols know that they should retry their request later. This still poses the same problems as ignoring requests. In particular, we can use the exact same distinguishing attack from above. At the end of the attack, in the real world the environment will receive an urgent request from B via the adversary, while in the ideal world it will get an error message from $\mathcal{F}$.

Queuing of intermediate requests. Instead of the above approaches, one might try to store all incoming messages to process them later on. The simplest implementation of this approach would be the following: Upon receiving another input while still waiting for a response to an urgent request, the ideal functionality stores the input in a queue and then ends its activation. After receiving a response from the adversary, the ideal functionality processes the messages stored in the queue.

This approach is vulnerable to the same attack as the previous approaches: if the environment executes this attack in the real world, it will eventually receive an urgent request from B. This, however, cannot be simulated in the ideal world. The simulator does not get control when B is activated since the ideal functionality simply ends its activation after queuing the input for B.

Another problem with this approach is that in all current universal composability models a machine is only allowed to send one message per activation. Hence, the ideal functionality will never be able to catch up with the inputs that have been stored. Every time it is activated by another input, it will have to process both the new input and several older inputs that are still stored in the queue. But it can only answer one of these messages at a time. This observation leads to another approach based on queuing of unanswered requests which we discuss in Appendix A. This approach, which does not seem to have been used in the literature so far, is, however, very complex and weakens the security of the ideal functionality to an extent which for some tasks is unacceptable: it allows the adversary to determine the order in which requests are processed by an ideal functionality.

In Appendix B, we discuss several additional approaches that try to solve at least some part of the problem for ideal functionalities, namely, they seek to address settings where the ideal functionality asks the adversary for some information. However, even for this more specific setting there is no general solution.

3.3 The Non-Responsiveness Problem in Real and Hybrid Protocols

After having demonstrated that dealing with the non-responsiveness problem in ideal functionalities is hard, we now discuss this problem in the context of real and hybrid protocols.

First, observe that the non-responsiveness problem and with this the problem of making ideal functionalities reentrant is propagated from ideal functionalities to all higher level protocols in a hybrid protocol. If a higher level protocol sends some message to an ideal functionality which in turn sends an urgent request on its network interface, the adversary becomes active, and hence, is able to send some input to the higher level protocol, which, however, is still waiting for a response from the ideal functionality. Hence, a protocol designer also has to design higher level protocols to be reentrant, even in cases where intuitively one would expect the ideal functionality to respond immediately for the specific request. This leads to the same problems explained in §3.2.

Furthermore, reentrance of protocols is only one aspect of the non-responsiveness problem. Besides being able to manipulate the control flow in unexpected ways by triggering a protocol waiting for an answer to an urgent request, the adversary can also block subroutines by never answering an urgent request or alter the state of other parts of the protocol, e.g., by corrupting previously honest parties, in the meantime. Even if a protocol designer is able to locally address the non-responsiveness problem by making all parts of the protocol reentrant (which, in itself, is already very difficult as illustrated in §3.2), he still has to deal with this problem on a global scale where data retrieved from subroutines might become outdated because the adversary altered

the state of a subroutine, or expected answers from subroutines are never received. While it might be possible to also address the problem on this global scale, potential solutions (unnecessarily) complicate definitions and proofs, and they will be highly dependent on the concrete protocols at hand.

These problems also exist for real protocols themselves, not only when they use ideal functionalities. Similar to ideal protocols, real protocols often use urgent requests to retrieve or provide some modeling related information. These requests are often part of the underlying model. For example, depending on the way corruption is modeled, a real protocol might have to actively ask the adversary for its corruption status, and some models require protocols to send a notification to the adversary upon their first creation (cf. "subroutine respecting protocols" in the 2013 version of UC [7]). Furthermore, urgent requests are sometimes specified by the protocol designer itself to get a more general modeling. For example, suppose one wants to analyze a multi-party protocol which can run in different modes or using different sets of (pre-shared) keys. One way of modeling this setting is to ask the adversary, by sending (urgent) requests, about the parameters and key IDs to be used when a new session is created.

Finally, for hybrid protocols we want to point out the following paradoxical situation. As illustrated in §3.2, in the attempt to make ideal functionalities reentrant, their security guarantees might be weakened, for example, because the adversary can change the order in which messages are processed or block certain requests, even though this would not be possible in reality. Even worse, the security guarantees might already be weakened by simply using urgent requests at all, in particular when the functionality is supposed to model a non-interactive task; a realization that does not give control to the adversary could yield stronger security. This can then lead to abstruse situations where a hybrid protocol using an ideal functionality cannot be proven secure while the same protocol using the realization of the ideal functionality instead is secure. This stands in contrast to one of the goals of universal composability models, namely, reducing the complexity of security analyses by being able to use conceptually simpler ideal functionalities as subroutines.

3.4 Examples from the Literature

We now illustrate the non-responsiveness problem by examples from the literature. Here we consider two settings where urgent requests occur particularly often: in functionalities modeling local computations and in the setup phases of protocols and functionalities. While these are examples where the adversary is requested to provide information, we emphasize that urgent requests are often also used when ideal functionalities have to provide *the adversary* with information, as outlined at the beginning of §3. Moreover, while the following examples consider urgent requests for ideal functionalities, they also occur in real and hybrid protocols, with the problems outlined in §3.3.

3.4.1 Local Computations

In some ideal functionalities the adversary needs to be contacted for realizing a supposedly local task. For instance, this is the case in several formulations of signature and public-key encryption functionalities, where the adversary is contacted every time a signature/ciphertext has to be created/verified/decrypted (and not only upon setup) [4, 7, 11, 24]. Other examples include functionalities for authentication [13], undeniable signatures [20], non-interactive zero-knowledge proofs of knowledge [23], or the dual-authentication certification functionality $\mathcal{F}_{\text{D-Cert}}$ [28]. The functionality for unlikable redactable signatures [5] contacts the adversary when checking public keys.

In all these functionalities, the adversary needs to be contacted when verifying some token. Since these functionalities are supposed to model local computations, the verification requests are urgent. However, the adversary may not answer immediately resulting in the non-responsiveness problem and its consequences discussed above. This even led to underspecified functionalities or functionalities exhibiting counterintuitive behavior, as illustrated next by $\mathcal{F}_{\text{D-Cert}}$ [28].

Figure 5 shows the Verify instruction of $\mathcal{F}_{\text{D-Cert}}$. Assume now that S' has received a message m and a signature σ for this message which supposedly was created by an honest party P with SID *sid*. Now, if the signature actually was not created by P, the verification should fail since P is not corrupted. However, as the adversary gets activated during this allegedly local task, it could corrupt the signer during the verification

Upon receiving a value $(\texttt{Verify}, sid, m, \sigma)$ from some party S', hand $(\texttt{Verify}, sid, m, \sigma)$ to the adversary. Upon receiving $(\texttt{Verified}, sid, m, \phi)$ from the adversary, do:

1. If $(m, \sigma, 1)$ is recorded then set $f = 1$.
2. Else, if the signer is not corrupted, and no entry $(m, \sigma', 1)$ for any σ' is recorded, then set $f = 0$ and record the entry $(m, \sigma, 0)$.
3. Else, if there is an entry (m, σ, f') recorded, then set $f = f'$.
4. Else, set $f = \phi$, and record the entry (m, σ, ϕ).

Output $(\texttt{Verified}, sid, m, f)$ to S'.

Fig. 5: The Verify instruction of $\mathcal{F}_{\text{D-Cert}}$ from [28].

process, return $\phi = 1$, and therefore let the functionality *accept* σ. This behavior is certainly unexpected and counterintuitive. It also considerably complicates the security analysis of any higher-level application which uses $\mathcal{F}_{\text{D-Cert}}$ as a subroutine, as one has to consider the possiblity of a party getting corrupted also during the invocation of a subroutine modeling a local task.

The following is an example where the authors struggled with the non-responsiveness problem, which finally led to a functionality that, as the authors acknowledge, is not completely satisfying. This functionality, denoted $\mathcal{F}_{\text{NIKE}}$, is supposed to model non-interactive key-exchange and was proposed by Freire et al. [17]. Figure 6 shows a central part of this functionality, namely, the actual key exchange. A party P_i may ask for the key that is shared between the parties P_i and P_j. If this session of P_i and P_j is considered corrupted, namely, because one of the parties is corrupted, and no key has been recorded for this session yet, the adversary is allowed to freely choose the key that is shared between both parties. The functionality uses an urgent request to model this, i.e., it directly sends a message to the adversary if he is allowed to choose a key.

Upon input $(\texttt{init}, P_i, P_j)$ from P_i, if $P_j \notin \Lambda_{\text{ref}}$, return $(P_i, P_j, \perp)$ to P_i. If $P_j \in \Lambda_{\text{reg}}$, we consider two cases:

- Corrupted session mode: if there exists an entry $(\{P_i, P_j\}, K_{i,j})$ in Λ_{keys}, set $key = K_{i,j}$. Else, send $(\texttt{init}, P_i, P_j)$ to the adversary. After receiving $(\{P_i, P_j\}, K_{i,j})$ from the adversary, set $key = K_{i,j}$ and add $(\{P_i, P_j\}, K_{i,j})$ to Λ_{keys}.
- Honest session mode: if there exists an entry $(\{P_i, P_j\}, K_{i,j})$ in Λ_{keys}, set $key = K_{i,j}$, else choose $key \leftarrow \{0,1\}^k$ and add $(\{P_i, P_j\}, K_{i,j})$ to Λ_{keys}.

Return (P_i, P_j, key) to P_i.

Fig. 6: The init instruction of $\mathcal{F}_{\text{NIKE}}$ from [17].

As the authors state, they would have liked to also model "immediateness" of the functionality, i.e., a higher level protocol that requests a key should be able to expect an answer without the adversary being able to interfere with the protocol in between. This indeed would be expected and natural because $\mathcal{F}_{\text{NIKE}}$ models a *non-interactive* key exchange. However, this is in conflict with allowing the adversary to choose the key of a corrupted session. The authors suggest that one option to also model immediateness might be to let the adversary choose an algorithm upon setup, which is then used to compute the keys for corrupt parties. They nevertheless chose the non-immediate modeling because the other solution would lead to "technical complications"; it would also limit the adaptiveness of the adversary and might add different issues as discussed in the next section.

As a consequence of the chosen formulation, the adversary can now, e.g., block requests, which again needs to be considered also in any higher-level protocol using $\mathcal{F}_{\text{NIKE}}$ as a subroutine, even though in the real world the honest party would always obtain some key because of the non-interactivity of the primitive.

3.4.2 Setup Instructions

Some non-interactive functionalities such as the 2005 version of Canetti's signature ideal functionality $\mathcal{F}_{\text{sig}}$, Canetti's public key ideal functionality [6,7], and Chase and Lysyanskaya's $\mathcal{F}_{\text{sok}}(L)$ [14] "signature of knowledge" functionality keep the contact to the adversary to a minimum by having a dedicated setup phase. During setup, these ideal functionalities ask the adversary for all necessary information such that they can operate without any calls to the adversary afterwards. There are many other examples that include a setup phase with urgent requests [1,5,12,15,16]. However, the non-responsiveness problem and its consequences still remain, even in cases where the functionality avoids using urgent requests after setup.

We use the signature of knowledge functionality $\mathcal{F}_{\text{sok}}(L)$ to illustrate this since it is one of those particularly severe examples that ignore the non-responsiveness problem altogether. This functionality consists of a Setup instruction (reproduced in Figure 7), where the adversary provides the keys and algorithms, and signing and verification instructions that then use those keys and algorithms without requiring interaction with the adversary. This functionality is explicitly intended to be used in a hybrid setting to realize delegatable credentials and hence needs to be reentrant. However, this is not the case:

If the adversary does not respond to the first (`Setup`, *sid*) request, all subsequent requests (e.g., a Setup request by a different party) will cause the functionality to use or output the undefined Sign and Verify algorithms. We now show that this is a problem: Chase and Lysyanskaya provide a protocol in the $\mathcal{F}_{\text{sok}}(L)$-hybrid model that can be used for realizing delegated credentials, i.e., an ideal functionality for signatures on a signature. They then prove that this protocol realizes the functionality. They, however, missed the fact that in that proof, $\mathcal{F}_{\text{sok}}(L)$ may interact with a non-responsive adversary in the real world. Such an adversary can force $\mathcal{F}_{\text{sok}}(L)$ to use undefined algorithms, and their simulator does not handle that situation in the ideal world. It is thus easy to distinguish the real from the ideal world, hence their proof is flawed.

Upon receiving a value (`Setup`, *sid*) from any party P, verify that $sid = (M_L, sid')$ for some sid'. If not, then ignore the request. Else, if this is the first time that (`Setup`, *sid*) was received, hand (`Setup`, *sid*) to the adversary; upon receiving (`Algorithms`, *sid*, Verify, Sign, Simsign, Extract) from the adversary, where Sign, Simsign, Extract are descriptions of PPT TMs, and Verify is a description of a deterministic polytime TM, store these algorithms. Output the stored (Algorithms, *sid*, Sign, Verify) to P.

Fig. 7: The Setup instruction of $\mathcal{F}_{\text{sok}}(L)$ from [14].

4 Universal Composability with Responsive Environments

The non-responsiveness problem and the resulting complications shown in §3 are artificial problems. Since urgent requests only exist for modeling purposes but do not model any real network traffic, a real adversary would not be able to use them to carry out attacks. Still, in all current universal composability models the non-responsiveness of adversaries enables attacks that do not correspond to anything in reality. If we could force the adversary to answer urgent requests immediately, which, as already mentioned before, would be the natural and expected behavior, there would not be any need for coming up with workarounds that try to solve the non-responsiveness problem in specifications of protocols and functionalities and one would not have to consider such artificial attacks in security proofs.

In this section, we present our framework which extends universal composability models by allowing protocol designers to specify messages that have to be answered immediately by (responsive) environments and adversaries. We first give a brief overview of our approach, then define in more detail responsive environments, responsive adversaries, the realization relation in this setting, and finally prove that the composition theorems still hold for our extension. Since our framework and concepts can be used by any universal composability model and in order to highlight the new concepts, we keep this section independent from specific models. In particular, we mostly ignore runtime considerations. In §5, we then discuss in detail how our framework can be adapted to specific models.

4.1 Overview

To solve the problems presented in §3, we introduce the concept of *responsive environments* and *responsive adversaries*. In a nutshell, when these environments and adversaries receive specific messages from the network, we call these messages *restricting*, then they have to respond to these messages immediately, i.e., without activating other parts of the protocol before sending an answer. Furthermore, depending on the restricting message, they may send an answer from a specific set of messages only. Restricting messages and the possible answers can be specified by the protocol designer, they are not hardwired into the framework. More specifically, restricting messages and the possible responses are specified by a binary relation $R \subseteq \{0,1\}^+ \times \{0,1\}^+$ over non-empty messages, called a *restriction*. If $(m, m') \in R$, then m is a restricting message and m' a possible answer to m. That is, if an environment/adversary receives m on its network interface, then it has to answer immediately with some m' such that $(m, m') \in R$.

This allows a protocol designer to specify all urgent requests as restricting messages by defining a restriction R appropriately; such requests are then not only answered immediately but also with an expected answer. Therefore the adversary can no longer interfere with the protocol run in an unintended way by activating other parts of the protocol or sending other inputs before answering an urgent request.

Note that this concept is very powerful and needs to be handled with care. While, as motivated above, it does not weaken security results if one models urgent requests as restricting messages, one must not use such messages when modeling real network traffic, as real network messages are not guaranteed to be answered immediately in reality.

4.2 Defining Responsiveness

To define responsive environments and responsive adversaries, we first precisely define the notion of a restriction. As mentioned before, restrictions are used to define both restricting messages, which have to be answered immediately by the environment/adversary, and possible answers to each restricting message.

Definition 4.1. *A* restriction R *is a set of pairs of non-empty messages, i.e.,* $R \subseteq \{0,1\}^+ \times \{0,1\}^+$*, such that, given a pair of messages* (m, m')*, it is efficiently decidable whether* R *allows* m' *as an answer to* m*. We define* $R[0] := \{m | \exists m' : (m, m') \in R\}$*. A message* $m \in R[0]$ *is called a* restricting message*.*

The idea is that if an environment/adversary receives m on the network interface, then there are two cases: If m is not a restricting message, i.e., $m \notin R[0]$, then the environment/adversary is not restricted in any way. Otherwise, if $m \in R[0]$, then the first message (if any) sent back to the protocol (both on the network and I/O interface of the protocol) has to be some message m' with $(m, m') \in R$. This message has to be sent on the network interface of the same machine that issued the request m without sending any other message to another machine of the protocol (see also Definition 4.2).

By requiring efficient decidability we ensure that environments are able to check whether some answer is allowed by the restriction; this is necessary, e.g., for Lemma 4.4. We refer to §5 for the exact definitions of "efficiently decidable", which depend on the runtime definitions of the underlying models.

As mentioned in §4.1, only urgent requests should be defined as restricting messages via a restriction. For example, upon creation of a new instance by receiving a message m, instances of protocols are often expected to first ask the adversary whether they are corrupted before they process the message m. An adversary can be forced to answer such a request immediately by the following restriction:

$$R := \{(m, m') | m = \texttt{AmICorrupted?}, m' = (\texttt{Corruption}, b), b \in \{\mathsf{false}, \mathsf{true}\}\}.$$

We now formalize the responsiveness property of environments and adversaries.

Definition 4.2 (Responsive Environments). *An environment* $\mathcal{E}$ *is called* responsive *for a system of machines* $\mathcal{Q}$ *with respect to a restriction* R *if in an overwhelming set of runs of* $\{\mathcal{E}, \mathcal{Q}\}$ *every restricting message from* $\mathcal{Q}$ *on the network is answered correctly, i.e., for any restricting message* $m \in R[0]$ *sent by* $\mathcal{Q}$ *on the network, the first message* m' *that* $\mathcal{Q}$ *receives afterwards (be it on the network interface or the I/O interface of* $\mathcal{Q}$*), if any, is sent by* $\mathcal{E}$ *on the network interface of* $\mathcal{Q}$ *to the same machine of* $\mathcal{Q}$ *that sent* m *and* m' *satisfies* $(m, m') \in R$*. By* $\mathsf{Env}_R(\mathcal{Q})$ *we denote the set of responsive environments for* $\mathcal{Q}$*.*

In the above definition, "same machine" typically means the same *instance* of a machine. So if an instance of a machine of $\mathcal{Q}$ sent a restricting message m on the network interface to the environment, the first message m' received by any instance of $\mathcal{Q}$ (on the network or I/O interface), including all currently running instances of $\mathcal{Q}$ and an instance that might be created as a result of m', has to be sent back on the network interface to the same instance of $\mathcal{Q}$ which sent m and m' has to satisfy $(m, m') \in R$. The exact definition of "same machine" depends on the model under consideration (see §5).

The system $\mathcal{Q}$ usually is either $\{\mathcal{A}_D, \mathcal{P}\}$, where $\mathcal{P}$ is a real protocol and $\mathcal{A}_D$ is the dummy adversary, or $\{\mathcal{S}, \mathcal{F}\}$, where $\mathcal{S}$ is an ideal adversary and $\mathcal{F}$ is an ideal protocol.

Responsive adversaries have to provide the same guarantees as responsive environments, however, they have to do so only when running in combination with a responsive environment. In other words, they can use the responsiveness property of the environment to ensure their own responsiveness property.

Definition 4.3 (Responsive Adversaries). *Let $\mathcal{Q}$ be a system and let $\mathcal{A}$ be an adversary that controls the network interface of $\mathcal{Q}$. Then, $\mathcal{A}$ is called a* responsive adversary *if, for all $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{A}, \mathcal{Q}\})$, in an overwhelming set of runs of $\{\mathcal{E}, \mathcal{A}, \mathcal{Q}\}$ every restricting message from $\mathcal{Q}$ on the network is immediately answered (in the sense of Definition 4.2). We denote the set of all such adversaries for a protocol $\mathcal{Q}$ by* $\mathrm{Adv}_R(\mathcal{Q})$.

We note that the dummy adversary $\mathcal{A}_D$ is responsive.

Note that both the definitions of responsive environments and responsive adversaries depend on a specific system, i.e., an environment which is responsive for a system $\mathcal{Q}$ is not necessarily responsive for a system $\mathcal{Q}'$. If we required environments to be responsive for *every* system, we would also have to require this from simulators (ideal adversaries). This in turn would needlessly complicate security proofs. Let us elaborate on this. Many theorems and lemmas in UC-like models, such as transitivity of the realization relation (cf. Lemma 4.7) and the composition theorems (cf. Theorem 4.8 and Theorem 4.9), are proven by simulating (some instances of) adversaries/simulators and protocols within the environment. In such proofs, we need to make sure that if an environment is responsive, then it is still responsive if we move a simulator (ideal adversary) into the environment, i.e., run the simulator within the environment. Now, if we require strong responsiveness (i.e., responsiveness for all systems), then moving a simulator into a responsive environment might result in an environment that is not responsive anymore (in the strong sense), unless we require from the simulator that it is responsive in the strong sense as well. However, imposing such a strong requirement on simulators seems unreasonable. Simulators are constructed in security proofs to work with exactly one protocol. So a protocol designer should only have to care about runs with this specific protocol, not with arbitrary systems that might try to actively violate the responsiveness property of the simulator. This is why we require responsiveness for specific systems only and this indeed is sufficient.

In fact, for security proofs there are two important properties that should be fulfilled and for which we now show that they are. The first says that if an environment is responsive for one system, then it is also responsive for any system indistinguishable from that system. The second property says that a responsive environment can internally simulate a responsive adversary/simulator without losing its responsiveness property. In other words, we can move a responsive adversary/simulator into a responsive environment without losing the responsiveness property of the environment. As mentioned before, this is necessary, for example, for the transitivity of the realization relation and the composition theorems.

Lemma 4.4. *Let R be a restriction. Let $\mathcal{Q}$ and $\mathcal{Q}'$ be two systems of machines such that $\{\mathcal{E}, \mathcal{Q}\} \equiv \{\mathcal{E}, \mathcal{Q}'\}$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q})$. Then, $\mathsf{Env}_R(\mathcal{Q}) = \mathsf{Env}_R(\mathcal{Q}')$.*

Proof. We argue that $\mathsf{Env}_R(\mathcal{Q}) \subseteq \mathsf{Env}_R(\mathcal{Q}')$, the other inclusion follows analogously. Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q})$ and suppose that $\mathcal{E} \notin \mathsf{Env}_R(\mathcal{Q}')$. In other words, unlike the system $\{\mathcal{E}, \mathcal{Q}'\}$, there is only a negligible set of runs of $\{\mathcal{E}, \mathcal{Q}\}$ where restricting messages on the network from $\mathcal{Q}$ are answered incorrectly (i.e., not immediately or with an unexpected answer). We can use this to construct a responsive environment $\mathcal{E}'$ that distinguishes $\mathcal{Q}$ from $\mathcal{Q}'$, in contradiction to the assumption.

The environment $\mathcal{E}'$ internally simulates $\mathcal{E}$ but checks, as soon as $\mathcal{E}$ wants to send a message to $\mathcal{Q}/\mathcal{Q}'$, whether this message violates the responsiveness property. If it does, $\mathcal{E}'$ aborts with output 1 instead of

sending the message; if the run is ended by $\mathcal{E}$ without violating the responsiveness property, $\mathcal{E}'$ outputs 0. This check is possible since restrictions require this problem to be efficiently decidable. Clearly, we have that $\{\mathcal{E}', \mathcal{Q}\} \not\equiv \{\mathcal{E}', \mathcal{Q}'\}$ and that $\mathcal{E}'$ is responsive for $\mathcal{Q}$, since $\mathcal{E}'$ always aborts before sending an incorrect answer. This contradicts the assumption that $\mathcal{Q}$ and $\mathcal{Q}'$ are indistinguishable for all responsive environments for $\mathcal{Q}$. □

Lemma 4.5. *Let R be a restriction. Let $\mathcal{Q}$ be a system, $\mathcal{A} \in \mathrm{Adv}_R(\mathcal{Q})$ be a responsive adversary, and $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{A}, \mathcal{Q}\})$ be a responsive environment. Let $\mathcal{E}'$ denote the environment which internally simulates the system $\{\mathcal{E}, \mathcal{A}\}$. Then, $\mathcal{E}' \in \mathsf{Env}_R(\mathcal{Q})$.*

Proof. Observe that the systems $\{\mathcal{E}, \mathcal{A}, \mathcal{Q}\}$ and $\{\mathcal{E}', \mathcal{Q}\}$ behave exactly the same. Let us assume that $\mathcal{E}' \notin \mathsf{Env}_R(\mathcal{Q})$. Then, there is a non-negligible set of runs of $\{\mathcal{E}', \mathcal{Q}\}$ where a restricting message from $\mathcal{Q}$ on the network is answered incorrectly. Since these messages are handled by the internally simulated adversary $\mathcal{A}$ in $\mathcal{E}'$, there must also be a non-negligible set of runs of $\{\mathcal{E}, \mathcal{A}, \mathcal{Q}\}$ where a restricting message from $\mathcal{Q}$ on the network to $\mathcal{A}$ is answered incorrectly. However, this is a contradiction to the responsiveness property of $\mathcal{A}$. This implies that $\mathcal{E}' \in \mathsf{Env}_R(\mathcal{Q})$. □

4.3 Realization Relation for Responsive Environments

We can now define the realization relation for responsive environments. The definition is analogous to the one for general environments and adversaries (see Definition 2.1), but restricts these entities to being responsive.

Definition 4.6 (Realizing Protocols with Responsive Environments). *Let $\mathcal{P}$ and $\mathcal{F}$ be protocols, the* real *and* ideal protocol*, respectively, and R be a restriction. Then, $\mathcal{P}$* realizes *$\mathcal{F}$ with respect to responsive environments ($\mathcal{P} \leq_R \mathcal{F}$) if for every responsive adversary $\mathcal{A} \in \mathrm{Adv}_R(\mathcal{P})$, there exists an (ideal) responsive adversary $\mathcal{S} \in \mathrm{Adv}_R(\mathcal{F})$ such that $\{\mathcal{E}, \mathcal{A}, \mathcal{P}\} \equiv \{\mathcal{E}, \mathcal{S}, \mathcal{F}\}$ for every environment $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{A}, \mathcal{P}\})$.*

Just as in the case of Definition 2.1, we have that instead of quantifying over all responsive adversaries, it suffices to consider the dummy adversary $\mathcal{A}_D$ only, which forwards all network messages between $\mathcal{P}$ and $\mathcal{E}$ (cf. Theorem C.24). As already mentioned, $\mathcal{A}_D$ is always responsive. This means that in security proofs, one has to construct one responsive simulator $\mathcal{S}$ for $\mathcal{A}_D$ only.

As already mentioned above Lemma 4.5, the responsiveness of $\mathcal{S}$ is necessary for the transitivity of $\leq_R$. While the responsiveness of $\mathcal{S}$ is a property a protocol designer has to make sure, this property is easy to check and guarantee: upon receiving a restricting message from the protocol, it either answers immediately and correctly or sends only restricting messages to the environment until it can provide a correct answer to the original restricting message from the protocol. In such a situation, the simulator should not send a non-restricting message to the environment because then the simulator cannot make sure that it gets back an answer immediately from the environment and that the environment does not invoke the protocol in between. In Appendix C.4, we make this precise and provide a formal proof of this intuition.

We also note that Definition 4.6 is a generalization of Definition 2.1: with $R := \emptyset$ we obtain Definition 2.1.

We now prove that the realization relation with responsive environments is reflexive and transitive. This is crucial for the modular and step-wise design of protocols: ones we have proven $\mathcal{P} \leq_R \mathcal{P}'$ and $\mathcal{P}' \leq_R \mathcal{P}''$, we want to immediately conclude that $\mathcal{P} \leq_R \mathcal{P}''$.

Lemma 4.7. *The $\leq_R$ relation is reflexive and transitive.*

Proof. In what follows, we take the (equivalent) formulation of $\leq_R$ with the dummy adversary $\mathcal{A}_D$.

Now, to prove reflexivity, we define $\mathcal{S} := \mathcal{A}_D$. This trivially implies $\{\mathcal{E}, \mathcal{A}_D, \mathcal{P}\} \equiv \{\mathcal{E}, \mathcal{S}, \mathcal{P}\}$. As already mentioned, $\mathcal{A}_D$ is responsive since it merely forwards messages between $\mathcal{E}$ and $\mathcal{P}$, and $\mathcal{E}$ itself is responsive for $\{\mathcal{A}_D, \mathcal{P}\}$. This shows reflexivity.

To prove transitivity, let us assume that $\mathcal{P} \leq_R \mathcal{P}'$ and $\mathcal{P}' \leq_R \mathcal{P}''$. We have to prove that $\mathcal{P} \leq_R \mathcal{P}''$. Let $\mathcal{S} \in \mathrm{Adv}_R(\mathcal{P}')$ and $\mathcal{S}' \in \mathrm{Adv}_R(\mathcal{P}'')$ be the simulators that yield $\mathcal{P} \leq_R \mathcal{P}'$ and $\mathcal{P}' \leq_R \mathcal{P}''$, respectively. To prove $\mathcal{P} \leq_R \mathcal{P}''$, we build a new simulator $\{\mathcal{S}, \mathcal{S}'\}$ as the combination of $\mathcal{S}$ and $\mathcal{S}'$ where the communication is

as follows: $\mathcal{S}'$ handles the network communication with $\mathcal{P}''$, the network communication of $\mathcal{S}'$ at its interface to the environment is handled by $\mathcal{S}$ at its network interface to $\mathcal{P}'$, and the network communication of $\mathcal{S}$ with the environment is in fact with the environment. Let $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{P}\})$. We have to prove that i) $\{\mathcal{E}, \mathcal{A}_D, \mathcal{P}\} \equiv \{\mathcal{E}, \mathcal{S}, \mathcal{S}', \mathcal{P}''\}$ and that ii) $\{\mathcal{S}, \mathcal{S}'\}$ is responsive for $\mathcal{P}''$.

i) By assumption, we have that $\{\mathcal{E}, \mathcal{A}_D, \mathcal{P}\} \equiv \{\mathcal{E}, \mathcal{S}, \mathcal{P}'\}$. In the second system, we can plug $\mathcal{A}_D$ between $\mathcal{S}$ and $\mathcal{P}'$ without changing the overall behavior of the system since $\mathcal{A}_D$ merely forwards messages between $\mathcal{S}$ and $\mathcal{P}'$. Hence, we have that $\{\mathcal{E}, \mathcal{S}, \mathcal{P}'\} \equiv \{\mathcal{E}, \mathcal{S}, \mathcal{A}_D, \mathcal{P}'\}$. Now suppose that $\{\mathcal{E}, \mathcal{S}, \mathcal{A}_D, \mathcal{P}'\} \not\equiv \{\mathcal{E}, \mathcal{S}, \mathcal{S}', \mathcal{P}''\}$ (otherwise, there is nothing to show). Then, the environment $\mathcal{E}'$ which internally simply simulates $\{\mathcal{E}, \mathcal{S}\}$ obviously distinguishes $\{\mathcal{A}_D, \mathcal{P}'\}$ from $\{\mathcal{S}', \mathcal{P}''\}$. Moreover, this environment is responsive for $\{\mathcal{A}_D, \mathcal{P}'\}$: First observe that $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{S}, \mathcal{A}_D, \mathcal{P}'\})$ by Lemma 4.4. Also observe that $\mathcal{S} \in \mathrm{Adv}_R(\{\mathcal{A}_D, \mathcal{P}'\})$ because $\mathcal{S} \in \mathrm{Adv}_R(\mathcal{P}')$ and $\mathcal{A}_D$ merely forwards messages. Moreover, by applying Lemma 4.5 for $\mathcal{Q} := \{\mathcal{A}_D, \mathcal{P}'\}$, we directly obtain that $\mathcal{E}' \in \mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{P}'\})$. This contradicts the assumption that $\{\mathcal{A}_D, \mathcal{P}'\}$ and $\{\mathcal{S}', \mathcal{P}''\}$ are indistinguishable by every environment in $\mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{P}'\})$. Hence, $\{\mathcal{E}, \mathcal{S}, \mathcal{A}_D, \mathcal{P}'\} \equiv \{\mathcal{E}, \mathcal{S}, \mathcal{S}', \mathcal{P}''\}$ and, by transitivity of the $\equiv$ relation, $\{\mathcal{E}, \mathcal{A}_D, \mathcal{P}\} \equiv \{\mathcal{E}, \mathcal{S}, \mathcal{S}', \mathcal{P}''\}$.

ii) Now let $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{S}, \mathcal{S}', \mathcal{P}''\})$ and define the environment $\mathcal{E}'$ which internally simulates $\{\mathcal{E}, \mathcal{S}\}$ just as in i). We now argue that $\mathcal{E}' \in \mathsf{Env}_R(\{\mathcal{S}', \mathcal{P}''\})$. Since $\mathsf{Env}_R(\{\mathcal{S}, \mathcal{S}', \mathcal{P}''\}) = \mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{P}\})$ by Lemma 4.4 and hence $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{P}\})$, we can reuse the results for $\mathcal{E}'$ from above, i.e., $\mathcal{E}' \in \mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{P}'\})$. Furthermore, because we have $\mathcal{P}' \leq_R \mathcal{P}''$ by assumption and by the definition of $\mathcal{S}'$, we can use Lemma 4.4 to obtain $\mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{P}'\}) = \mathsf{Env}_R(\{\mathcal{S}', \mathcal{P}''\})$. This gives $\mathcal{E}' \in \mathsf{Env}_R(\{\mathcal{S}', \mathcal{P}''\})$. Now observe that, because $\mathcal{S}'$ is a responsive adversary and $\mathcal{E}'$ a responsive environment, there can only be a negligible set of runs of $\{\mathcal{E}', \mathcal{S}', \mathcal{P}''\}$ where a restricting message of $\mathcal{P}''$ on the network is answered incorrectly. Since the systems $\{\mathcal{E}', \mathcal{S}', \mathcal{P}''\}$ and $\{\mathcal{E}, \mathcal{S}, \mathcal{S}', \mathcal{P}''\}$ behave exactly the same, this implies that $\{\mathcal{S}, \mathcal{S}'\}$ is a responsive adversary for $\mathcal{P}''$. □

4.4 Composition Theorems

The core of every universal composability model are the composition theorems. We now present a first composition theorem that handles concurrent composition of any (fixed) number of potentially different protocols.

Theorem 4.8. *Let R be a restriction. Let $k \geq 1$, $\mathcal{Q}$ be a protocol, and $\mathcal{P}_1, \ldots, \mathcal{P}_k$, $\mathcal{F}_1, \ldots, \mathcal{F}_k$ be protocols such that for all $j \leq k$ it holds true that $\mathcal{P}_j \leq_R \mathcal{F}_j$.*
Then, $\{\mathcal{Q}, \mathcal{P}_1, \ldots, \mathcal{P}_k\} \leq_R \{\mathcal{Q}, \mathcal{F}_1, \ldots, \mathcal{F}_k\}$.

Proof. In what follows, we take the (equivalent) formulation of $\leq_R$ with the dummy adversary $\mathcal{A}_D$.

It suffices to prove the theorem for the case $k = 1$. The argument can then be iterated to obtain the theorem for $k > 1$ using transitivity of the $\leq_R$ relation. Let $\mathcal{S} \in \mathrm{Adv}_R(\mathcal{F}_1)$ be the simulator from the definition of $\mathcal{P}_1 \leq_R \mathcal{F}_1$. Define the simulator $\mathcal{S}'$ to forward messages between the environment and $\mathcal{Q}$, while internally simulating $\mathcal{S}$ for messages between the environment and $\mathcal{F}_1$. Now let $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{Q}, \mathcal{P}_1\})$. For convenience, in what follows we split $\mathcal{A}_D$ into $\mathcal{A}_D^{\mathcal{Q}}$ and $\mathcal{A}_D^{\mathcal{P}_1}$ where $\mathcal{A}_D^{\mathcal{Q}}$ forwards all communication betweeen $\mathcal{E}$ and $\mathcal{Q}$ and $\mathcal{A}_D^{\mathcal{P}_1}$ forwards all communication betweeen $\mathcal{E}$ and $\mathcal{P}_1$.

We first prove that $\{\mathcal{E}, \mathcal{A}_D, \mathcal{Q}, \mathcal{P}_1\} \equiv \{\mathcal{E}, \mathcal{S}', \mathcal{Q}, \mathcal{F}_1\}$. Suppose that this is not the case. Then we can define a new environment $\mathcal{E}'$ that distinguishes $\{\mathcal{A}_D^{\mathcal{P}_1}, \mathcal{P}_1\}$ from $\{\mathcal{S}, \mathcal{F}_1\}$. The environment $\mathcal{E}'$ internally simulates $\{\mathcal{E}, \mathcal{A}_D^{\mathcal{Q}}, \mathcal{Q}\}$, and hence, distinguishes with the same probability as $\mathcal{E}$. Now observe that $\mathcal{E}'$ is responsive for $\{\mathcal{A}_D^{\mathcal{P}_1}, \mathcal{P}_1\}$: All network messages from $\{\mathcal{A}_D^{\mathcal{P}_1}, \mathcal{P}_1\}$ in $\{\mathcal{E}, \mathcal{A}_D, \mathcal{Q}, \mathcal{P}_1\}$ are handled by $\mathcal{E}$ only, not by $\mathcal{Q}$. Moreover, since $\mathcal{E}$ is responsive for $\{\mathcal{A}_D, \mathcal{Q}, \mathcal{P}_1\}$, we have that these messages are answered correctly (in the sense of Definition 4.2), implying the responsiveness of $\mathcal{E}'$ for $\{\mathcal{A}_D^{\mathcal{P}_1}, \mathcal{P}_1\}$. This contradicts the assumption that $\mathcal{P}_1 \leq_R \mathcal{F}_1$, and hence, $\{\mathcal{E}, \mathcal{A}_D, \mathcal{Q}, \mathcal{P}_1\} \equiv \{\mathcal{E}, \mathcal{S}', \mathcal{Q}, \mathcal{F}_1\}$ must be true.

We still have to show the responsiveness property of $\mathcal{S}'$, that is, $\mathcal{S}' \in \mathrm{Adv}_R(\{\mathcal{Q}, \mathcal{F}_1\})$. Let $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{S}', \mathcal{Q}, \mathcal{F}_1\})$. We have to show that all restricting network messages from $\mathcal{Q}$ and $\mathcal{F}_1$ to $\mathcal{E}$ and $\mathcal{S}'$ are answered correctly (in the sense of Definition 4.2). Suppose that there is a non-negligible set of runs of $\{\mathcal{E}, \mathcal{S}', \mathcal{Q}, \mathcal{F}_1\}$ where a restricting network message from $\{\mathcal{Q}, \mathcal{F}_1\}$ is not answered correctly. Since $\mathcal{S}'$ only forwards network messages from $\mathcal{Q}$ to the environment and the environment is responsive for $\{\mathcal{S}', \mathcal{Q}, \mathcal{F}_1\}$,

we have that with overwhelming probability these messages are answered correctly. Hence, there must be a non-negligible set of runs where network messages from $\mathcal{F}_1$ are not answered correctly. Now consider $\mathcal{E}'$ from above. Then there also is a non-negligible set of runs of $\{\mathcal{E}', \mathcal{S}, \mathcal{F}_1\}$ where restricting messages on the network from $\mathcal{F}_1$ are answered incorrectly because, by construction of $\mathcal{E}'$, the behavior of the system $\{\mathcal{E}', \mathcal{S}, \mathcal{F}_1\}$ coincides with $\{\mathcal{E}, \mathcal{S}', \mathcal{Q}, \mathcal{F}_1\}$. We already know that $\mathcal{E}' \in \mathsf{Env}_R(\{\mathcal{A}_D^{\mathcal{P}_1}, \mathcal{P}_1\})$ from above. Also, by assumption, we have that $\{\mathcal{E}'', \mathcal{A}_D^{\mathcal{P}_1}, \mathcal{P}_1\} \equiv \{\mathcal{E}'', \mathcal{S}, \mathcal{F}_1\}$ for all $\mathcal{E}'' \in \mathsf{Env}_R(\{\mathcal{A}_D^{\mathcal{P}_1}, \mathcal{P}_1\})$. Now, by Lemma 4.4 it follows that $\mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{P}_1\}) = \mathsf{Env}_R(\{\mathcal{S}, \mathcal{F}_1\})$, and hence, $\mathcal{E}' \in \mathsf{Env}_R(\{\mathcal{S}, \mathcal{F}_1\})$. This contradicts the responsiveness property of $\mathcal{S}$. □

The following composition theorem guarantees the secure composition of an unbounded number of instances of the same protocol system. To state this theorem, we consider single-session (responsive) environments, i.e., environments that invoke a single session of a protocol only. In universal composability models instances of protocol machines have IDs consisting of party IDs and session IDs. Instances with the same session ID form a session. Instances from different sessions may not directly interact with each other. A *single-session environment* may invoke machines with the same session ID only. We denote the set of single-session environments for a system $\mathcal{Q}$ by $\mathsf{Env}_{R,\mathsf{single}}(\mathcal{Q})$. We say that $\mathcal{P}$ *single-session realizes* $\mathcal{F}$ ($\mathcal{P} \leq_{R,\mathsf{single}} \mathcal{F}$) if there exists a simulator $\mathcal{S} \in \mathrm{Adv}_R(\mathcal{F})$ such that $\{\mathcal{E}, \mathcal{A}_D, \mathcal{P}\} \equiv \{\mathcal{E}, \mathcal{S}, \mathcal{F}\}$ for all $\mathcal{E} \in \mathsf{Env}_{R,\mathsf{single}}(\{\mathcal{A}_D, \mathcal{P}\})$. Now, the composition theorem states that if a single session of a real protocol $\mathcal{P}$ realizes a single session of an ideal protocol $\mathcal{F}$, then multiple sessions of $\mathcal{P}$ realize multiple sessions of $\mathcal{F}$.

Theorem 4.9. *Let R be a restriction and let $\mathcal{P}$ and $\mathcal{F}$ be protocols. Then, $\mathcal{P} \leq_{R,\mathsf{single}} \mathcal{F}$ implies $\mathcal{P} \leq_R \mathcal{F}$.*

Proof. Let $\mathcal{S}$ be the simulator for $\mathcal{P} \leq_{R,\mathsf{single}} \mathcal{F}$. A new simulator $\mathcal{S}'$ for arbitrary responsive environments can be constructed just as in the original (non-responsive) composition theorem, i.e., $\mathcal{S}'$ internally keeps one copy of $\mathcal{S}$ per session and uses these copies to answer messages from/to the corresponding sessions.

The proof consists of two main steps: The first step shows indistinguishability of $\{\mathcal{A}_D, \mathcal{P}\}$ and $\{\mathcal{S}', \mathcal{F}\}$ for every responsive environment $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{P}\})$. The second step shows the responsiveness property of the simulator.

The first part uses a hybrid argument where one builds a series of single-session environments $\mathcal{E}_i, i \geq 1$, which internally simulate $\mathcal{E}$ such that all messages to the first $i-1$ sessions are sent to internally simulated instances of $\{\mathcal{S}, \mathcal{F}\}$, messages to the i-th session are sent to the (external) system $\{\mathcal{A}_D, \mathcal{P}\}$ or $\{\mathcal{S}, \mathcal{F}\}$, respectively, and the remaining messages are sent to internally simulated instances of $\{\mathcal{A}_D, \mathcal{P}\}$. Since different sessions of a protocol do not directly interact with each other, it is easy to see that $\{\mathcal{E}_1, \mathcal{A}_D, \mathcal{P}\}$ behaves just as $\{\mathcal{E}, \mathcal{A}_D, \mathcal{P}\}$ (*), and $\{\mathcal{E}_n, \mathcal{S}, \mathcal{F}\}$ behaves just as $\{\mathcal{E}, \mathcal{S}', \mathcal{F}\}$, where $n \in \mathbb{N}$ is an upper bound of the number of sessions created by $\mathcal{E}$ (note that n is a polynomial in the security parameter and the length of the external input given to the environment, if any). Hence, the distinguishing advantage of $\mathcal{E}$ is bounded by the sum of the advantages of $\mathcal{E}_1, \ldots, \mathcal{E}_n$, i.e., it is sufficient to show that the advantages of $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are bounded by the *same* negligible function[7] to show that $\mathcal{E}$ cannot distinguish $\{\mathcal{A}_D, \mathcal{P}\}$ from $\{\mathcal{S}', \mathcal{F}\}$. In what follows, to show the existence of a single negligible function, we consider environments with external input because the argument is simpler in this case. Nevertheless, using sampling of runs, the argument also works without external input, i.e., in the uniform case, as discussed in Appendix C.7.

To show that such a bound exists, it is first necessary to prove that there is a (single) negligible function f that, for every $i \leq n$, bounds the probability of $\mathcal{E}_i$ of violating the responsiveness property in runs of $\{\mathcal{A}_D, \mathcal{P}\}$ or $\{\mathcal{S}, \mathcal{F}\}$, respectively. Let $\hat{C}_i^{\{\mathcal{A}_D, \mathcal{P}\}}$ be the event that in runs of $\{\mathcal{E}_i, \mathcal{A}_D, \mathcal{P}\}$ the environment $\mathcal{E}$, which is internally simulated by $\mathcal{E}_i$, answers a restricting message of the external system $\{\mathcal{A}_D, \mathcal{P}\}$ *or* one of the internally simulated instances of $\{\mathcal{A}_D, \mathcal{P}\}$ and $\{\mathcal{S}, \mathcal{F}\}$ incorrectly; $\hat{C}_i^{\{\mathcal{S}, \mathcal{F}\}}$ is defined analogously. Because $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{P}\})$ and because of (*), we have that $\hat{C}_1^{\{\mathcal{A}_D, \mathcal{P}\}}$ is negligible. It also holds true that (**) there exists a single negligible function that bounds $|\Pr\left[\hat{C}_i^{\{\mathcal{A}_D, \mathcal{P}\}}\right] - \Pr\left[\hat{C}_i^{\{\mathcal{S}, \mathcal{F}\}}\right]|$ for all $i \geq 1$. This

[7] It is not sufficient to show that the advantage of every environment $\mathcal{E}_i$ is bounded by a negligible function f_i, which is actually rather easy to show. The negligible functions f_i might be different and then their sum $f_1 + \cdots + f_n$ might not be negligible.

is because one can define a single session responsive environment $\mathcal{E}'$ that gets i as external input and then simulates $\mathcal{E}_i$; $\mathcal{E}'$ aborts and outputs 1 as soon as a restricting message is about to be answered incorrectly, while it outputs 0 otherwise. Note that because the restriction R can be decided efficiently, $\mathcal{E}'$ can perform the described task. Also, by construction, $\mathcal{E}'$ is a single-session environment (it invokes a single external session only) and it is responsive (it stops the execution before the responsiveness requirement would be violated). Since $\mathcal{E}'$ distinguishes $\{\mathcal{A}_D, \mathcal{P}\}$ and $\{\mathcal{S}, \mathcal{F}\}$ only based on the events $\hat{C}_i^{\{\mathcal{A}_D,\mathcal{P}\}}$ and $\hat{C}_i^{\{\mathcal{S},\mathcal{F}\}}$, and both systems are indistinguishable for every single session responsive environment, statement (**) holds true. Finally observe that, for all $i \geq 2$, the systems $\{\mathcal{E}_{i-1}, \mathcal{S}, \mathcal{F}\}$ and $\{\mathcal{E}_i, \mathcal{A}_D, \mathcal{P}\}$ behave exactly the same, and hence, $\Pr\left[\hat{C}_i^{\{\mathcal{A}_D,\mathcal{P}\}}\right] = \Pr\left[\hat{C}_{i-1}^{\{\mathcal{S},\mathcal{F}\}}\right]$. This implies that there is a single negligible function that bounds $\Pr\left[\hat{C}_i^{\{\mathcal{A}_D,\mathcal{P}\}}\right]$ for all $1 \leq i \leq n$ (here we need that n is polynomially bounded).[8] In particular, we have that the probability that $\mathcal{E}_i$ is not responsive for the system $\{\mathcal{A}_D, \mathcal{P}\}$ is bounded by a single negligible function independently of $i \leq n$.

We can now conclude the indistinguishability argument by showing that the advantages of $\mathcal{E}_i$, $1 \leq i \leq n$, in distinguishing $\{\mathcal{A}_D, \mathcal{P}\}$ from $\{\mathcal{S}, \mathcal{F}\}$ are bounded by the same negligible function. For this, we construct another single session responsive environment $\mathcal{E}''$ analogously to $\mathcal{E}'$. The system $\mathcal{E}''$ expects $1 \leq i \leq n$ as external input (and otherwise stops) and then exactly simulates $\mathcal{E}_i$. Importantly, $\mathcal{E}''$ is responsive for $\{\mathcal{A}_D, \mathcal{P}\}$ because we have shown that every $\mathcal{E}_i$ violates responsiveness with at most the same negligible probability, i.e., the same bound also holds for $\mathcal{E}''$ for every input. Since $\mathcal{E}''$ is a single session responsive environment, its distinguishing advantage for the systems $\{\mathcal{A}_D, \mathcal{P}\}$ or $\{\mathcal{S}, \mathcal{F}\}$ is negligible for every possible input. Moreover, with external input i its distinguishing advantage is the same as the one for $\mathcal{E}_i$. Hence, the same negligible function that bounds the advantage of $\mathcal{E}''$ also bounds all advantages of $\mathcal{E}_i, i \leq n$. As mentioned at the beginning of the proof, this implies that indistinguishability of $\{\mathcal{A}_D, \mathcal{P}\}$ and $\{\mathcal{S}', \mathcal{F}\}$ for every responsive environment $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{P}\})$.

Having proved indistinguishability, it remains to show that $\mathcal{S}'$ is responsive, i.e., $\mathcal{S}' \in \mathrm{Adv}_R(\mathcal{F})$. Let $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{S}', \mathcal{F}\})$. We have to show that the probability that all restricted messages from $\mathcal{F}$ in runs of $\{\mathcal{E}, \mathcal{S}', \mathcal{F}\}$ are answered correctly (in the sense of Definition 4.2) is overwhelming. For this, consider the following single session environment $\mathcal{E}'$ that is meant to run with $\{\mathcal{S}, \mathcal{F}\}$: The system $\mathcal{E}'$ first flips $r \leq n$, with n as above, and then internally simulates $\mathcal{E}$ and several sessions of $\{\mathcal{S}, \mathcal{F}\}$ such that messages from $\mathcal{E}$ to the r-th session are sent to the external session, while all other messages are processed by the internally simulated sessions. Note that $\{\mathcal{E}', \mathcal{S}, \mathcal{F}\}$ behaves just as $\{\mathcal{E}, \mathcal{S}', \mathcal{F}\}$, and hence, since $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{S}', \mathcal{F}\})$, by Lemma 4.4 we have that $\mathcal{E}'$ is responsive for $\{\mathcal{S}, \mathcal{F}\}$. Because $\mathcal{S}$ is a responsive adversary, this implies that there is only a negligible set of runs of $\{\mathcal{E}', \mathcal{S}, \mathcal{F}\}$ where a restricting message of $\mathcal{F}$ is answered incorrectly (by $\mathcal{E}'$ or $\mathcal{S}$). Hence, the probability for this to happen is bounded by some negligible function f. From this and the fact that there are only polynomial many sessions, it follows that the probability that a restricting message from some session of $\mathcal{F}$ is answered incorrectly is negligible. Hence, $\mathcal{S}'$ is a responsive adversary. □

We note that Theorems 4.8 and 4.9 can be combined to obtain more and more complex protocols. For example, one can first show that a single session of a real protocol $\mathcal{P}$ realizes a single session of an ideal protocol $\mathcal{F}$. Using the two theorems it, for example, then follows that a protocol $\mathcal{Q}$ using multiple sessions of $\mathcal{P}$ realizes $\mathcal{Q}$ using multiple sessions of $\mathcal{F}$.

4.5 Retaining Security Results for Different Restrictions

We note that all of our lemmas and theorems have been proven using a single restriction R. Hence, formally, a protocol designer would have to use the same restriction in all of her security proofs in order to be able to use our results. However, sometimes it is desirable to extend a restriction, for example, because one wants to introduce a new type of restricting message that is necessary to model a specific protocol, or to combine the results of two different security analyses that used different restrictions. This section first shows that

[8] Note that it also follows that $\Pr\left[\hat{C}_i^{\{\mathcal{S},\mathcal{F}\}}\right]$ is bounded for all $1 \leq i \leq n$. We, however, do not need this result in the following.

extending restrictions is possible in principle, and then shows in particular how results of two different security analyses can be combined, even if they used different restrictions. Also note, as discussed later on in §6.2, that there in fact is one generic restriction that would suffice for all purposes.

The following lemma allows us to extend restrictions while still retaining security results. The general idea is as follows: Observe that there are two major constraints in the security notion (cf. Definition 4.6), namely, indistinguishability for all responsive environments and responsiveness of the simulator. To be able to use a restriction R' instead of R while still retaining security results, we have to preserve both constraints. The first can be preserved if R' only further restricts responsive environments and adversaries, hence yielding a subset of responsive environments and adversaries for R. The latter can be preserved if the ideal protocol simply never sends one of the new restricting messages that were added by R', since then there are no runs where the simulator answers one of these new messages incorrectly.

Lemma 4.10. *Let R be a restriction and let $\mathcal{P}$ and $\mathcal{F}$ be protocols such that $\mathcal{P} \leq_R \mathcal{F}$. Let R' be an extension of R, i.e., $R \subseteq R'$. Then we have that $\mathcal{P} \leq_{R'} \mathcal{F}$ if the following holds true:*

- *$\{(m, m') | m \in R[0], (m, m') \in R' \backslash R\} = \emptyset$, that is, R' does not add new allowed answers to any restricting message of $R[0]$, and*
- *for all systems $\mathcal{Q}$, the protocol $\mathcal{F}$ in runs of $\{\mathcal{Q}, \mathcal{F}\}$ never outputs a message $m \in R'[0] \backslash R[0]$ on the network interface.*

Proof. Let $\mathcal{S}$ be the simulator that yields $\mathcal{P} \leq_R \mathcal{F}$. We have to show that (1) $\{\mathcal{E}, \mathcal{A}_D, \mathcal{P}\} \equiv \{\mathcal{E}, \mathcal{S}, \mathcal{F}\}$ for every environment $\mathcal{E} \in \mathsf{Env}_{R'}(\{\mathcal{A}_D, \mathcal{P}\})$ and (2) that $\mathcal{S} \in \mathrm{Adv}_{R'}(\mathcal{F})$.

For (1), observe that by assumption we have $R[0] \subseteq R'[0]$, i.e., R' only adds additional restricting messages. Furthermore, also by assumption, R' does not add new possible answers to any restricting message already defined in $R[0]$. Therefore the restriction R' actually restricts responsive environments and adversaries more than R, i.e., any environment in $\mathsf{Env}_{R'}(\{\mathcal{A}_D, \mathcal{P}\})$ also is in $\mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{P}\})$. Because we have $\{\mathcal{E}, \mathcal{A}_D, \mathcal{P}\} \equiv \{\mathcal{E}, \mathcal{S}, \mathcal{F}\}$ for every environment $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{A}_D, \mathcal{P}\})$ by the definition of $\mathcal{P} \leq_R \mathcal{F}$, (1) directly follows.

For (2), let $\mathcal{E} \in \mathsf{Env}_{R'}(\{\mathcal{S}, \mathcal{F}\})$. Observe that $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{S}, \mathcal{F}\})$ by the same reasoning as in the above paragraph. Now first suppose there was a non-negligible set of runs of $\{\mathcal{E}, \mathcal{S}, \mathcal{F}\}$ where $\mathcal{S}$ answers a restricting message $m \in R[0]$ incorrectly by the definition of R' (note that this does not consider messages $m \in R'[0] \backslash R[0]$). Because $R \subseteq R'$, these answers would also be incorrect by the definition of R. This contradicts the responsiveness of $\mathcal{S}$ (for R) since $\mathcal{E} \in \mathsf{Env}_R(\{\mathcal{S}, \mathcal{F}\})$.

Finally observe that $\mathcal{F}$ will never send a restricting message $m \in R'[0] \backslash R[0]$ in runs of $\{\mathcal{E}, \mathcal{S}, \mathcal{F}\}$ by assumption. Therefore $\mathcal{S}$ also will never answer such a message incorrectly. Combined with the results from the previous paragraph, this implies $\mathcal{S} \in \mathrm{Adv}_{R'}(\mathcal{F})$. □

Using Lemma 4.10, we directly obtain that security results carry over if one extends a restriction R by adding some *new* restricting message not used by $\mathcal{F}$ with arbitrary sets of allowed answers. This allows a protocol designer to simply add new messages each time she analyzes a different protocol, without having to re-prove all previous results. Should $\mathcal{F}$ use some restricting messages one wants to add, then one could slightly redefine $\mathcal{F}$ (e.g., by prefixing some messages) to avoid overlap.

We now explain how Lemma 4.10 even allows us to combine security results for two different restrictions R_1 and R_2 where one is not necessarily an extension of the other as required by the lemma.

To combine two (or more) security results for different restrictions, one has to define a new restriction R that combines all restrictions. As a running example illustrating the exact procedure assume two protocols $\mathcal{P}_1, \mathcal{P}_2$ that realize ideal functionalities $\mathcal{F}_1, \mathcal{F}_2$ for restrictions R_1, R_2, i.e., $\mathcal{P}_1 \leq_{R_1} \mathcal{F}_1$ and $\mathcal{P} \leq_{R_2} \mathcal{F}_2$. To combine both results, e.g., via a composition theorem (cf. Theorem 4.8), we need a restriction R such that $\mathcal{P}_1 \leq_R \mathcal{F}_1$ and $\mathcal{P}_2 \leq_R \mathcal{F}_2$. Generally speaking, to construct such a restriction, one syntactically changes both the protocols and the restrictions such that protocols using different restrictions only send the same restricting message if both restrictions agree on the allowed answers. It is then simple to combine both restrictions.

To be more precise, first define $R_{1 \cap 2}$ to be the binary set that contains all pairs $(m, m') \in R_1$ such that $m \in R_1[0] \cap R_2[0]$ and both R_1 and R_2 define the same allowed answers for m. In other words, $R_{1 \cap 2}$ contains

those message pairs on which both restrictions coincide. Now, syntactically change R_1 and R_2 in the following way such that there is no restricting message with $m \in R_1[0] \cap R_2[0]$ but $m \notin R_{1\cap 2}[0]$. In other words, for every restricting message from the resulting restrictions, say R'_1 and R'_2, they either agree in the allowed answers or the restricting message is only defined in one of these restrictions. This can easily be established, e.g., by prefixing every restricting message where both restrictions do not agree with different strings.

Now observe that we can modify the protocols (and simulators) to adjust them to the new restrictions by simply renaming messages in the same way. This yields new protocols $\mathcal{P}'_1, \mathcal{P}'_2, \mathcal{F}'_1, \mathcal{F}'_2$ such that $\mathcal{P}'_1 \leq_{R'_1} \mathcal{F}'_1$ and $\mathcal{P}'_2 \leq_{R'_2} \mathcal{F}'_2$. Note that these protocols are "as good as" the original protocols for a security analysis, since their behavior is still exactly the same.

Now define $R := R'_1 \cup R'_2$.[9] By construction, we can use Lemma 4.10 to obtain $\mathcal{P}'_1 \leq_R \mathcal{F}'_1$ and $\mathcal{P}'_2 \leq_R \mathcal{F}'_2$. This yields realizations for the same restriction and thus it is now possible to apply all lemmas and theorems from our framework.

5 Responsive Environments in Concrete Models

In the previous section, we have presented our universal composability framework with responsive environments in a rather model independent way. In this section, we outline how to implement this framework in the prominent UC, GNUC, and IITM models, respectively, to exemplify that our framework and concepts are sufficiently general to be applicable to any universal composability model. While these three models follow the same general idea, they differ in several details which affect the concrete implementation of our concepts in these models (see, e.g., [18,22] for a discussion of these differences). The main differences and details to be considered concern runtime definitions and mechanism for addressing (instances of) machines.

To instantiate our universal composability framework with responsive environments for the mentioned models, we mainly have to concretize the definitions in §4.2, that is, the definitions of restrictions as well as responsive environments and adversaries for these models. For some models we also have to adjust their runtime notions a bit. Before presenting the details for the specific models, in what follows we briefly explain the central points to be taken care of:

Runtime. In the GNUC and IITM model, the runtime of systems/protocols is polynomially bounded only for a certain class of environments. Since we now want to consider only responsive environments, we should restrict the class of environments to those that are responsive. This also has some technical advantages. To see this, let $\mathcal{R}$ and $\mathcal{R}'$ be two systems/protocols. For example, $\mathcal{R}$ and $\mathcal{R}'$ could be the systems $\{\mathcal{E}, \mathcal{A}_D, \mathcal{Q}, \mathcal{P}\}$ and $\{\mathcal{E}, \mathcal{S}, \mathcal{Q}, \mathcal{I}\}$ as considered in the composition theorem (Theorem 4.8) when we want to prove that $\{\mathcal{Q}, \mathcal{P}\}$ realizes $\{\mathcal{Q}, \mathcal{I}\}$. We often face the situation that we know that say $\mathcal{R}$ satisfies the model's runtime bound for all environments in a certain class and that $\mathcal{R}$ and $\mathcal{R}'$ are indistinguishable for every *responsive* environment $\mathcal{E}$ (in this class). This implies that $\mathcal{R}'$ has to satisfy the runtime notion as well but only for all responsive environments of the class. Hence, one cannot necessarily use $\mathcal{R}'$, with any environment, in another system as it does not satisfy the model's runtime notion (for non-responsive environments $\mathcal{E}$ the runtime of $\mathcal{R}'$ might not be polynomial). Hence, also from a technical point of view, it makes sense to relax the runtime notions in these models in that the runtime of systems/protocols should be required to be polynomially bounded for responsive environments only.

Definition of restrictions. According to Definition 4.1, we require that restrictions are "efficiently decidable". As mentioned, the exact definition depends on the model at hand. The important property that this definition should satisfy is the following. An environment $\mathcal{E}'$ that internally simulates another environment $\mathcal{E}$ should be able to decide that when the internal environment $\mathcal{E}$ gets a restricting message as input whether the output it produces is a correct answer (according to the restriction). Often we use such a simulation to prove that $\mathcal{E}$ is responsive. But at this point we do not know this yet, and therefore, depending on the model under consideration, do not necessarily have guarantees about the length of the restricting message sent to $\mathcal{E}$ (a polynomially bound on this length might be guaranteed for responsive

[9] Formally, one still has to make sure that R actually is a restriction, i.e., fulfills Definition 4.1. This, however, should be the case for natural definitions of R_1 and R_2.

environments only, as already explained above). A model dependent definition of an efficiently decidable restriction should take this into account.

Definition of responsive environments. In the definition of responsive environments (Definition 4.2), we require that an answer to a restricting message is sent back to the same machine and we already explained that "same machine" typically means the same instance from which the restricting message has been received from. This has to be made precise for the different models.

Definition of responsive adversaries. Depending on the restriction R considered, in some models, in particular UC and GNUC, Definition 4.3 can be too restrictive, and for example, the dummy adversary in these models might not satisfy the definition. The dummy adversary in these models is required to perform multiplexing. When it receives a message from an instance of the protocol and forwards this message to the environment, it has to prefix the message with the ID of that instance in order tell the environment where the message came from. This alters the message and the resulting message might no longer be restricting, depending on the definition of the restriction R. Hence, the environment would not be obliged anymore to answer directly, and thus, the (dummy) adversary would not be responsive. One way to fix this is to require a certain closure property of restrictions, namely adding IDs at the beginning of restricting messages still yields restricting messages and these message permit the same answers. But this is quite cumbersome. For example, by recursively applying this constraint one would have to require that R is closed under arbitrarily long prefixes of sequences of IDs. A more elegant solution that would still allow for simple and natural restrictions would redefine what it means for a message from an adversary to the environment to be restricting. This is what we suggest for the UC and GNUC models (see below).

In what follows, we sketch how to adjust and concretize the runtime notions and the mentioned definitions for the UC, GNUC, and IITM models. As already mentioned in the introduction, for the IITM model we have carried out the implementation of responsive environments in this model in full detail.

5.1 UC

For the UC model, we do not have to change the runtime definition since the runtime of a protocol is not defined w.r.t. a class of environments but simply bounded by a fixed polynomial (see also below).

Definition of restrictions. For UC we require both R and $R[0]$ to be decidable in polynomial time in the length of the input. Due to UC's strict runtime definition, this is sufficient to satisfy the requirement mentioned above, namely, that an environment $\mathcal{E}'$ simulating another environment $\mathcal{E}$ can check whether a restricting message received by $\mathcal{E}$ is answered correctly by $\mathcal{E}$. To see this, recall that every machine in UC is required to be parameterized with a polynomial. At every point in the run, the runtime of every instance of a machine is bounded by this polynomial, where the polynomial is in $n := n_I - n_O$, with n_I being the number of bits received so far on the I/O interface from higher level machines and n_O being the number of bits sent on the I/O interface to lower level machines. Environment machines have to satisfy this condition as well, where n_I is the number of bits of the external input (which contains the security parameter η). Hence, since protocols will receive only a polynomial number of input bits from the environment, they can send messages of polynomial length in the length of the external input plus η only. Hence, given some message m that was received by an environment and a response m' to this message, the message pair (m, m') has at most polynomial length in the external input plus η, and therefore an environment is able to decide within its runtime bound whether m' is a correct answer to m, if we use the above definition of effectively decidable restrictions.

Definition of responsive environments. We require that a response to a restricting message is sent back to the *instance* of the machine that sent the restricting message. This is possible since every instance in UC is assigned a globally unique ID, which is then used to specify sender and recipient of a message.

Definition of responsive adversaries. As already explained above, messages from the adversary to the environment and vice versa may contain a prefix (typically an ID). For reasons already explained above, in UC we say that such a prefix is ignored for the sake of checking whether a message is restricting and

whether the answer is correct. To be more specific, a message $m = (pre, \bar{m})$ from the adversary to the environment is restricting iff $\bar{m} \in R[0]$. Also, if m is restricting (in this sense), an answer $m' = (pre', \bar{m}')$ from the environment is allowed if $(\bar{m}, \bar{m}') \in R$. Using this definition, it is easy to see that the dummy adversary in UC, which adds some prefix to messages from a protocol to the environment and strips off a prefix from messages from the environment to a protocol, is responsive.

5.2 GNUC

The changes necessary for the GNUC model are similar to those for the UC model. However, the runtime notion has to be modified as explained.

Runtime. Let us first recall the relevant parts of the runtime definition of GNUC.[10] In this model, the runtime definition depends on the entity considered. For an environment $\mathcal{E}$, there has to exist a polynomial p that bounds the runtime of the environment in runs with *every* system, where p gets as input the number of bits of all messages that have been received by the environment during the run, including the external input, plus the security parameter η. For a protocol $\mathcal{P}$ there has to exist a polynomial q such that the runtime of $\mathcal{P}$ is bounded by q in runs with any environment and the dummy adversary, where q gets as input the number of bits that are output by the environment (to both the adversary and the protocol). As motivated at the beginning of this section, this definition has to be changed such that the runtime of a protocol needs to be bounded only for all environments (in the sense of GNUC) that in addition are responsive.

Definition of restrictions. Analogously to UC, we require R and $R[0]$ to be decidable in polynomial time in the length of the input. This is sufficient to satisfy the described requirement ($\mathcal{E}'$ simulating $\mathcal{E}$) since the runtime of environments in GNUC depends on the number of bits received from a protocol. Hence, an environment is always able to read a potentially restricting message m entirely, while the length of an answer m' is bounded by the runtime bound of the environment.

Definition of responsive environments. Just as for UC, we require that responses to restricting messages are sent to the same *instance* of a machine. This is possible in GNUC since, again, all machines have globally unique IDs to address instances.

Definition of responsive adversaries. Since, just as for UC, the adversary in GNUC might (have to) add IDs as prefixes or remove such prefixes, these prefixes are ignored in the definition of responsive adversaries.

5.3 IITM

Just as for the other models, we now outline how to adjust and concretize the runtime notion and the definitions from §4 for the IITM model. As already mentioned, in Appendix C we provide full details for the IITM model with responsive environments, with a brief summary of the results presented at the end of this subsection.

Runtime. In the IITM model, the runtime depends on the type of entity. For an environment $\mathcal{E}$ it is required that there exists a polynomial p (in the length of the external input, if any, plus the security parameter) such that for *every* system running with $\mathcal{E}$ the runtime of $\mathcal{E}$ with this system is bounded by p. For a protocol $\mathcal{P}$ it is merely required that it is environmentally bounded, i.e., for every environment $\mathcal{E}$ there is a polynomial q (again, in the length of the external input plus the security parameter) that bounds the overall runtime of runs of $\{\mathcal{E}, \mathcal{P}\}$ (except for at most a negligible set of runs).[11] Given a protocol $\mathcal{P}$, for an adversary $\mathcal{A}$ for $\mathcal{P}$ it

[10] Note that there are several additional requirements, such as bounds on the number of bits that are sent by the environment as well as so-called invited messages. These details, however, are not relevant here.

[11] Here $\mathcal{E}$ may directly connect to $\mathcal{P}$'s network interface. Equivalently one could have $\mathcal{E}$ communicate with $\mathcal{P}$ on the network interface via a dummy adversary.

is required only that $\{\mathcal{A}, \mathcal{P}\}$ is environmentally bounded. (Clearly, the dummy adversary is environmentally bounded.) To adjust the runtime notions for the setting with responsive environments, in the definition of environmentally bounded protocols/adversaries, instead of quantifying over all environments, one should now quantify over responsive environments only, as motivated at the beginning of §5.

Definition of restrictions. We require that a restriction R is *efficiently decidable in the second component*, i.e., there is an algorithm A which expects pairs (m, m') of messages as input and which runs in polynomial time in $|m'|$ in order to decide whether m' is a correct answer to m according to R (cf. Definition C.9). This stronger definition is necessary to obtain the described property, namely that an environment $\mathcal{E}'$ internally simulating another environment $\mathcal{E}$ can check that answers of $\mathcal{E}$ to restricting messages are correct. Due to the very liberal runtime notion for protocols employed in the IITM model, in proofs (e.g., of the composition theorem) we sometimes have to establish that a system is environmentally bounded. Therefore, we do not know a prior that the length of the message m is polynomially bounded. Hence, the environment might not be able to read m completely. Conversely, the length of m' is guaranteed to be polynomially bounded as it is output by the environment $\mathcal{E}$, which, by definition, is polynomially bounded. With R being efficiently decidable in the second component, $\mathcal{E}'$ can then efficiently decide whether m' is a correct answer to m. Compared to the definition of restrictions for the UC and GNUC model presented above, this formally is more restricted. It is, however, sufficient for all practical purposed, as discussed in §6.2.

Definition of responsive environments. Unlike the UC and GNUC model, the IITM model does not hardwire a specific addressing mechanism for instances of machines and specific IDs for such instances into the model. It rather supports a flexible addressing mechanism which allows a protocol designer to specify how machine instances are addressed and what they consider to be their ID. More specifically, the IITM model allows a protocol designer to specify an algorithm run by machine instances that decides whether the message received is accepted by the instance or not. Therefore, in the IITM model we can require only that responses to restricting messages are sent to the same machine, but not necessarily the same machine *instance*. This, however, is indeed sufficient. A protocol designer, as typically done in the IITM model, can specify that a (protocol) machine accepts a message iff it is prefixed by a certain ID (the one seen in the first activation of the instance). This ID can then be considered to be the ID of this machine instance, and messages output by this machine would also be prefixed by this ID. Now, a protocol designer can use restrictions to manually enforce that the same instance receives a response. Such a restriction would contain message pairs of the form $((id, m), (id, m'))$. By this it is guaranteed that if a restricting message was sent by a protocol machine instance with ID id, then the response is returned to this instance, as the response is prefixed with id. Appendix C.2 shows an example of such a restriction.

Definition of responsive adversaries. For the IITM model, we do not have to change the definition of responsive adversaries. Adversaries in the IITM model do not have to add prefixes to messages, and hence, do not have to modify restricting messages. In particular, the dummy adversary simply forwards messages between the environment and the protocol without changing messages (see also the definition of dummy adversaries in Appendix C.5).

Detailed Results for the IITM Model. In Appendix C we provide full details of the IITM model with responsive environments. That is, we adjust the runtime notion of the IITM model accordingly, provide full definitions of restrictions, responsive environments and adversaries. Based on these definitions we define the various security notions for realization relations considered in the literature (now with responsive environments), namely, (dummy) UC, black-box simulatability, strong simulatability, and reactive simulatability. These new and adjusted notions have been carefully developed in order to be general and preserve central properties. In particular, we show that all the mentioned notions for realization relations are equivalent (for reactive simulatability this requires environments with external input). We also prove that these relations are reflexive and transitive. We finally prove the composition theorems for responsive environments. As should be clear from the proof sketches in §4, the proofs are more involved than those without responsive environments since one always has to make sure that the constructed environments and simulators are responsive. The full proofs

in Appendix C are even more intricate and non-trivial because they take all model specific details, such as the liberal runtime notions, into account. We note, however, that this is a once and for all effort. Protocol designers do not have to perform such proofs anymore. They can simply use the results. That is, responsive environments do not put any burden on the protocol designer. On the contrary, they, as explained, greatly simply the specification and analysis of protocols.

6 Applying Our Concepts to the Literature

In this section, we briefly show how our concepts can be used to elegantly resolve the issues discussed in §3. We first provide concrete solutions to the examples from §3.4, which do not require any changes to the ideal functionalities and their realizations, but require some attention when combining protocols. Subsequently, we give a more generic solution, which does not require to define individual restricting messages for every functionality.

6.1 Concrete Solutions

One possible solution for the issues discussed in §3.4 is to define individual sets of restricting messages for each functionality, forcing the adversary to immediately reply to all urgent requests of the respective functionality.

Local computation. As a concrete example for this approach, consider the case of $\mathcal{F}_{\text{D-Cert}}$. There, the binary relation of restricting messages could be defined as follows:

$$\left\{\left((\texttt{Verify}, sid, m, \sigma), (\texttt{Verified}, sid, m, \phi)\right) : sid, m, \sigma \in \{0,1\}^*, \phi \in \{0,1\}\right\}$$

Now, whenever the adversary is asked to verify some σ, the next message sent to the ideal functionality is guaranteed to be the expected response, and the adversary is no longer able to deny verification requests or to change the corruption state of the signer during verification. This directly resolves the issues discussed in §3.4.1, and similar solutions are possible for $\mathcal{F}_{\text{NIKE}}$ and $\mathcal{F}_{\text{sok}}$.

Note that in order to show that the respective real protocols realize their ideal functionalities, according to Definition 4.6, one needs to prove that there exists a *responsive* simulator. However, it is easy to verify that the simulators constructed in [14, 17, 28] are already responsive, and thus these realizations can be unalteredly used also in a responsive setting.

We also note that the above approach of defining a separate restriction for each protocol is general in the sense that it can be used independently of the underlying model for universal composition, and is thus applicable, e.g., to the UC, GNUC, and IITM models. Furthermore, this solution allows one to fix many ideal functionalities and their realizations found in the literature without any modifications to the specifications, in particular all examples mentioned in this document. However, since the composition theorems and the transitivity property assume one restriction, different restrictions have to be combined to one. This is always possible as explained in §4.5. Nevertheless, the following solution seems preferable.

6.2 Generic Restriction

Alternatively to individual restrictions, one can use the following generic restriction:

$$R_G := \{(m, m') | m = (\texttt{Respond}, m''), m', m'' \in \{0,1\}^*\}.$$

This means that messages prefixed with `Respond` are considered to be restricting, and hence, protocol designers can declare a message to be restricting by simply prefixing it by `Respond`. While, according to the definition of R_G, the adversary/environment can respond with any message to these messages, protocols or ideal functionalities can be defined in such a way that they repeat their requests until they receive the expected answer: for instance, in the case of $\mathcal{F}_{\text{sok}}$ it can repeatedly send $m'' = (\texttt{Setup}, sid)$ to the adversary

until it receives the expected algorithms. In this way, the adversary is forced to eventually provide an expected answer if it wants the protocol run to continue.

Using this fixed multi-purpose restriction has the advantage that, in contrast to the former approach, there is no need to combine different restrictions. Also, in protocols specifications the prefixing makes immediately clear which messages are considered to be restricting.

The main reasons why we did not hardwire the generic restriction into our framework are twofold. First, this is not required for our results to go through, making our framework only more general, and this flexibility might become useful in some situations. Second, as protocols and ideal functionalities have to send several requests until they get the expected answer, depending on the runtime notions employed, they might run out of resources. Unfortunately, this is the case in the UC model, where the runtime of every machine is fixed by a polynomial and does not depend on the input received on the network interface. In the GNUC and IITM models this exhaustion is not possible, and hence, the generic restriction can be employed in these models (see also Appendix C.2 for the IITM model).

7 Conclusion

Urgent requests, i.e., requests where the adversary/environment is asked for or provided with (meta-)information, frequently occur in specifications of protocols and ideal functionalities. In none of the previous models for universal composition immediate responses to such requests are guaranteed, and hence, protocol designers have to specify what happens when the protocol/ideal functionality is activated before urgent requests are answered. As pointed out in this paper, dealing with this problem, which really is a modeling artifact, is hard and resulted in underspecified, less secure, flawed, and/or unnecessarily complex specifications, with the problem propagating to higher-level protocols.

Our framework completely circumvents the problem. It enables protocol designers to declare urgent requests to be restricting messages, and hence, such requests are answered immediately by (responsive) environments/adversaries, allowing for specifying protocols and ideal functionalities in the expected and natural way. We discussed how our concepts can be adopted by existing models for universal composition, as exemplified in this work by the UC, GNUC, and IITM models. For the IITM model, we have provided full details, showing that our concepts can seamlessly be integrated into the existing model without losing any of the properties compared to the setting without responsive environments: all security notions for the realization relations are formulated, shown to be (still) equivalent, and enjoy reflexivity and transitivity; the composition theorems carry over to the setting with responsive environments as well.

References

1. Abe, M., Ohkubo, M.: A framework for universally composable non-committing blind signatures. In: ASIACRYPT 2009. LNCS, vol. 5912, pp. 435–450. Springer (2009)
2. Backes, M., Dürmuth, M., Hofheinz, D., Küsters, R.: Conditional Reactive Simulatability. International Journal of Information Security (IJIS) 7(2), 155–169 (April 2008)
3. Backes, M., Pfitzmann, B., Waidner, M.: The reactive simulatability (RSIM) framework for asynchronous systems. Information and Computation 205(12), 1685–1720 (2007)
4. Backes, M., Hofheinz, D.: How to break and repair a universally composable signature functionality. In: ISC 2004. LNCS, vol. 3225, pp. 61–72. Springer (2004)
5. Camenisch, J., Dubovitskaya, M., Haralambiev, K., Kohlweiss, M.: Composable & modular anonymous credentials: Definitions and practical constructions. ASIACRYPT 2015 (2015)
6. Canetti, R.: Universally Composable Signature, Certification, and Authentication. In: CSFW 2004. pp. 219–233. IEEE (2004)
7. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (2000)
8. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001)

9. Canetti, R., Cheung, L., Kaynar, D.K., Liskov, M., Lynch, N.A., Pereira, O., Segala, R.: Time-Bounded Task-PIOAs: A Framework for Analyzing Security Protocols. In: DISC 2006. LNCS, vol. 4167, pp. 238–253. Springer (2006)
10. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer (2007)
11. Canetti, R., Halevi, S., Katz, J.: Adaptively-secure, non-interactive public-key encryption. In: TCC 2005. LNCS, vol. 3378, pp. 150–168. Springer (2005)
12. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer (2003)
13. Canetti, R., Shahaf, D., Vald, M.: Universally composable authentication and key-exchange with global PKI. Cryptology ePrint Archive, Report 2014/432 (2014)
14. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer (2006)
15. Damgård, I., Hofheinz, D., Kiltz, E., Thorbek, R.: Public-key encryption with non-interactive opening. In: CT-RSA 2008. LNCS, vol. 4964, pp. 239–255. Springer (2008)
16. Dowsley, R., Müller-Quade, J., Otsuka, A., Hanaoka, G., Imai, H., Nascimento, A.C.A.: Universally composable and statistically secure verifiable secret sharing scheme based on pre-distributed data. IEICE Transactions 94-A(2), 725–734 (2011)
17. Freire, E.S.V., Hesse, J., Hofheinz, D.: Universally composable non-interactive key exchange. In: SCN 14. pp. 1–20. LNCS, Springer (2014)
18. Hofheinz, D., Shoup, V.: GNUC: A new universal composability framework. Cryptology ePrint Archive, Report 2011/303 (2011)
19. Hofheinz, D., Unruh, D., Müller-Quade, J.: Polynomial runtime and composability. Journal of Cryptology 26(3), 375–441 (2013)
20. Kurosawa, K., Furukawa, J.: Universally composable undeniable signature. In: ICALP 2008, Part II. LNCS, vol. 5126, pp. 524–535. Springer (2008)
21. Küsters, R.: Simulation-Based Security with Inexhaustible Interactive Turing Machines. In: CSFW '06. pp. 309–320. IEEE (2006)
22. Küsters, R., Tuengerthal, M.: The IITM model: a simple and expressive model for universal composability. Cryptology ePrint Archive, Report 2013/025 (2013)
23. Laud, P., Ngo, L.: Threshold Homomorphic Encryption in the Universally Composable Cryptographic Library. Cryptology ePrint Archive, Report 2008/367 (2008)
24. Matsuo, T., Matsuo, S.: On universal composable security of time-stamping protocols. In: IWAP 2005. pp. 169–181 (2005)
25. Maurer, U.: Constructive Cryptography - A New Paradigm for Security Definitions and Proofs. In: TOSCA 2011. LNCS, vol. 6993, pp. 33–56. Springer (2011)
26. Maurer, U., Renner, R.: Abstract cryptography. In: ICS 2011. pp. 1–21. Tsinghua University Press (2011)
27. Pfitzmann, B., Waidner, M.: Composition and integrity preservation of secure reactive systems. In: ACM CCS 00. pp. 245–254. ACM Press (2000)
28. Zhao, S., Zhang, Q., Qin, Y., Feng, D.: Universally composable secure TNC protocol based on IF-T binding to TLS. In: NSS 2014. LNCS, vol. 8792, pp. 110–123. Springer (2014)

A Queuing of Intermediate Requests with Notifications to the Adversary

In the situation described in §3.2, and in particular the queuing approach described there, instead of simply ending the activation of the ideal functionality after queuing an input, one could also choose to send a notification to the adversary for each message that is stored this way. The adversary would then be expected to send responses to each notification such that the ideal functionality is activated sufficiently often to process every message. To illustrate the complexity and see the disadvantages of this approach, let us first be more precise about the exact implementation. If the ideal functionality, upon receiving some input, wants to send an urgent request, the input is first stored (in some set). Then the ideal functionality not only sends the urgent request itself, but also a notification consisting of the ID of the sender of the original input message and a token associated with this message. Now, every time the ideal functionality receives another input before receiving a response to its urgent request, it will again store this input and send another notification

(ID and token) to the adversary. As soon as the ideal functionality receives a response to its urgent request, it stores the data from this response and returns control to the adversary. Now, the adversary is allowed to send responses to the notifications he received, which can be uniquely identified by the token in the notification. As soon as the ideal functionality receives such a response, it will process and then delete the input message associated with the token in the response. Note that it should now be possible to process these inputs because the ideal functionality received the answer to its urgent request before. However, while processing one of these stored messages, the ideal functionality might have to send a second urgent request, leading to yet another set of queued input messages, and so on.

Observe that this gives a lot of additional power to the adversary by allowing him to see the original sender of an input, blocking certain requests by never responding to a notification, and being able to change the order in which messages are processed. These abilities are necessary to prevent the same kind of artificial distinguishing attack presented for the previous approaches. In fact, to prevent the specific attack in our running example, the simulator must know whether he has to simulate an instance of A or B upon input (which is why we have to include the ID), and he has to be able to tell the ideal functionality whether it should first process the stored message for A or the stored message for B. If the ideal functionality would not allow the latter and instead process the messages in the order they arrived, an environment could still distinguish: if in the above attack the environment first answers the urgent request of B, then the simulator must be able to tell $\mathcal{F}$ to process this request first, although A sent the first request.

Although this approach prevents the simple distinguishing attack from before, it still has three severe drawbacks: First, it is very complex and not very intuitive, especially if ideal functionalities send more than one urgent request. Second, this approach prevents some artificial distinguishing attacks by giving a lot of additional power to the adversary. This weakens the overall security guarantees provided by an ideal functionality. The adversary gains more information and is now able to use new attacks to potentially distinguish the real and ideal worlds. Third, this approach is still not generally applicable to any ideal functionality. This is the case, for example, if the ideal functionality has to preserve the order of execution to model its intended task.

The complexity and the weakened security guarantees are probably the reasons why this approach does not seem to have been followed in the literature so far.

B Solutions for Providing Data for Ideal Functionalities

This section discusses more potential approaches on how to deal with urgent requests in the specification of ideal functionalities. However, unlike those presented in §3.2, these approaches try to adress only a specific subset of urgent requests, namely those that try to retrieve some meta information from the adversary. Nevertheless, as we show in the following, while they might be applicable in some cases, they are also not sufficiently general to be used for any ideal functionality, and again, they in any case tackle only a specific subset of urgent requests, ignoring, for example, urgent requests in real/hybrid protocols and cases where protocols/functionalities want to provide the adversary with information. Altogether, this emphasizes the difficulty of solving the non-responsiveness problem in a satisfying way in the existing models, i.e., those without responsive environments and adversaries.

Using a default. To solve the non-responsiveness of the adversary when an ideal functionality asks for some algorithms, one might be tempted to add a default to the functionality such that this default will be used if the next message received is not from the adversary or does not contain the expected information.

While this definition seems straightforward at first, it actually is much more complex as soon as one tries to implement it. Note that after the ideal functionality sent its urgent request, the next activation may be due to some input on the I/O interface. The ideal functionality would then have to process two requests at the same time (while using the default), which is not possible in current UC-like models. Dropping one of the two requests generally is not an option, since this leads to exactly the same problems as described for the *ignoring requests* approach. Hence, a queuing approach is necessary where messages are processed one after another. As discussed in §3.2, the simplest form of a queue does not work in general either, while a much

more complex form (cf. Appendix A) severly weakens security guarantees of ideal functionalities and hence is usually unacceptable.

Besides the problems mentioned above, there are further issues with this approach. First, it might not always be possible to find a sensible default. Second, in some settings the environment can force the usage of the default algorithm and thus distinguish real and ideal world. To see this, consider the running example of §3.2 and the example attack presented in the *ignoring requests* paragraph. In this setting $\mathcal{F}$ would be forced to use the default; the simulator has no way to first provide other values since he has to simulate the network traffic of A before $\mathcal{F}$ produces output on the I/O interface (which happens as soon as the simulator provides the algorithms). Hence, any simulation that depends on the simulator being able to choose the information that is used by $\mathcal{F}$ will fail.

Not asking for information at all (code upload constructs). One could try to eliminate urgent requests from the ideal functionality (at least those that ask for some information from the adversary) by requiring that the adversary actively sends the required information before it is needed. Let us be more precise: Such an ideal functionality accepts a special message containing some data from the adversary at any point in time. Then, as soon as the ideal functionality has to process a request from the I/O interface, it uses the data it has previously received. If it has not received any data prior to the request, it resorts to a default (and stays with this default even if the adversary later on provides different data). One extreme form of this variant are code upload constructs, where the adversary first provides some code which is then internally simulated by the ideal functionality to generate answers to any (urgent) request it might have sent otherwise.

However, this approach, which tries to eliminate one type of urgent requests altogether, cannot be used in general. In particular, an environment might again be able to force usage of the default to be able to distinguish between the real and ideal world. To see this, consider an ideal functionality $\mathcal{F}$ that requires some information at the beginning (e.g., signing and verification algorithms in case of $\mathcal{F}_{\text{sig}}$) and otherwise does not contact the adversary/simulator at all because it models a local computation. Let P be a realization of $\mathcal{F}$ that uses a specific algorithm. To simulate P the simulator has to provide the same algorithm (or a variant of it) to $\mathcal{F}$. Since $\mathcal{F}$ may not send an urgent request to the simulator, the simulator now has to somehow figure out when and to which instance of $\mathcal{F}$ to send the required algorithm. The simulator should send this information before (the instance of) $\mathcal{F}$ is used for the first time. Even if it is guaranteed that the simulator is activated before $\mathcal{F}$ (such as in the UC model, where the adversary/simulator is always activated directly after the environment), the simulator would still have to know which session of $\mathcal{F}$ is going to be used by the environment. However, the environment is not forced to tell the simulator which session ID it is going to use for the challenge session and thus a simulator can only guess at this point (note that the simulator will be wrong most of the time since he can only guess a polynomial number of sessions, while the environment is free to choose from an exponential number). It is also not possible to provide the algorithm later on since $\mathcal{F}$ using this approach does not contact the simulator at all. Hence, as soon as the environment regains control, it can access $\mathcal{F}$ directly for some random session ID, and thus, force (the instance of) $\mathcal{F}$ to use the default algorithm (with overwhelming probability).

Even if the simulator somehow gets to know the ID of the challenge session, the same issue as above reoccurs as soon as one tries to find a joint state realization of $\mathcal{F}$. A distinguishing environment for the joint state realization is not bound to only use a single session but may access an arbitrary number of different sessions, so a simulator has again no means to provide algorithms "in time" for all possible sessions of $\mathcal{F}$.

A variant of this approach defines a special instance with a fixed ID which gets all data from the adversary/simulator. This instance is then used as a subroutine by all ideal functionalities that need to access this data. While this fixes the above issue by defining a single, known ID for the instance to which the simulator has to send its data, this comes at a hefty price: It is no longer possible to use the composition theorem for multiple sessions. In this setting, all sessions access *the same* subroutine instance and hence sessions are no longer disjoint, as required by the theorem.

All of the above approaches also have an additional disadvantage. As mentioned at the beginning, they only seek to address urgent requests where the ideal functionality wants to retrieve information *from* the adversary, but not urgent requests that are meant to provide information *to* the adversary. Also, urgent requests for

real/hybrid protocols are not addressed either. So even if one of the above approaches can be applied, which depends on the concrete functionalities and realizations, it only fixes part of the problem and this partial solution leads to artificially complex ideal functionalities.

C The IITM Model with Responsive Environments

In this section, we provide detailed definitions and proofs of our responsive environments framework in the IITM model. In particular, we precisely define the terms *restriction*, *responsive environment*, and *responsive adversary* from §4.2 now in the IITM model. Furthermore, we define the different security notions (unlike in §4.3, we not only define UC and dummy UC but also strong simulatability, black-box simulatability, and reactive simulatability) w.r.t. responsive environments and prove that, just as in the original IITM model with general (i.e., non-responsive) environments, they are still equivalent. We also prove the composition theorems (cf. §4.4) in the IITM model now with responsive environments.

More specifically, the structure of this section is as follows. In Appendix C.1 we recall the original IITM model briefly. Appendix C.2 formally defines restrictions in the IITM model and Appendix C.3 defines responsive environments, responsive adversaries, and the security notion of strong simulatability with responsive environments. In Appendix C.4 we show that the responsiveness property for simulators is easy to check. Appendix C.5 defines further security notions and proves that they are still equivalent to strong simulatability. Finally, Appendix C.6 and Appendix C.7 state and prove both composition theorems.

C.1 Recalling Notions from the IITM Model

Before we can extend the IITM model to work with responsive environments, we first briefly recall the original IITM model from [22]. Note that in the following we will introduce and use the original notiation of the IITM model instead of using the model independent notation from §2. The general structure of this section is as follows: First, we recall the definitions of IITMs (the machines in the IITM model) and systems of IITMs, including runs of systems. Then we introduce I/O and network interfaces of machines and systems. Finally, we recall the original runtime definitions of the IITM model in detail and formally specify environments, adversaries, and protocols.

Before we start, we have to recall the formal definition of negligible functions with external input, cf. Canetti [8]:

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

C.1.1 IITMs and Systems of IITMs

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), and only input tapes may be named start and only output tapes may be named decision. An IITM runs in one of two modes, **CheckAddress** and **Compute**. The **CheckAddress** mode is used as a generic mechanism for addressing copies of IITMs in a system of IITMs, as explained in Appendix C.1.2. 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 **CheckAddress** mode cannot alter the state of the IITM, that is, after this mode finishes the local state is reset to the state before running the **CheckAddress** mode. The actual processing of the message, if accepted, is done in mode **Compute**. In mode **Compute**, a machine may only output 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, in Appendix C.1.4, 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. 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.

A *system* $\mathcal{Q}$ of IITMs is of the form $\mathcal{Q} = M_1 \mid \cdots \mid M_k \mid !\, M'_1 \mid \cdots \mid !\, M'_{k'}$ 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) copies of a machine can be generated. Conversely, for machines which are not in the scope of a bang operator there exists at most a single copy in a run.

C.1.2 Running a System

In a run of a system $\mathcal{Q}$ at any time only one IITM is active and all other IITMs wait for new input; the first IITM to be activated in a run of $\mathcal{Q}$ is the so-called master IITM, of which a system has at most one and which may get external input. The master IITM is identified by its input tape named start on which the external input of a run is written. 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 $\mathcal{Q} = M_1 \mid !\, M_2$ 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 $\mathcal{Q}$ executed so far, two copies 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. If M'''_2 accepts m, then M'''_2 gets to process m. Otherwise, M'''_2 is removed again, the message m is dropped, and the master IITM is activated, in this case M_1, and so on. The master IITM is also activated if the currently active IITM does not produce output, i.e., stops in this activation without writing to any output tape (sometimes also called *empty output*). A run stops if the master IITM, after being activated, does not produce output (and hence, does not trigger another machine) or an IITM outputs a message on a tape named decision. Such a message is considered to be the *(overall) output of the system*. If no message is written on decision, the overall output of a (finite) run is ϵ.

If two systems output 1 with the same probability (except for a negligible difference), these systems are called indistinguishable. More formally:

Definition C.2 (Equivalence/Indistinguishability of Systems). *Two systems $\mathcal{Q}$ and $\mathcal{R}$ are called* equivalent *or* indistinguishable *($\mathcal{Q} \equiv \mathcal{R}$) if and only if there exists a negligible function $f : \mathbb{N} \times \{0,1\}^* \to \mathbb{R}_{\geq 0}$ such that for every security parameter η and external input $a \in \{0,1\}^*$ the following holds true:*

$$|\Pr[\mathcal{Q}(1^\eta, a) = 1] - \Pr[\mathcal{R}(1^\eta, a) = 1]| \leq f(\eta, a)$$

C.1.3 Interfaces and Connecting Systems

Given a system $\mathcal{Q}$, an input/output tape of an IITM in $\mathcal{Q}$ is called *external* iff there is no corresponding output/input tape with the same name in $\mathcal{Q}$; non-external tapes are called *internal*. External tapes can be used to connect other systems to $\mathcal{Q}$. A system $\mathcal{R}$ can be connected to $\mathcal{Q}$ if the external interfaces of both systems agree, that is, if both systems have external tapes with the same name then these tapes have

opposite directions. Systems $\mathcal{Q}$ and $\mathcal{R}$ with this property are called *connectable* and the combination of both systems $\mathcal{Q} \mid \mathcal{R}$ is well-defined (internal tapes are implicitly renamed such that both systems only connect via external tapes). We also say that $\mathcal{Q}$ can be *connected* to $\mathcal{R}$ (and vice versa). Given multiple systems $\mathcal{Q}_1, \ldots, \mathcal{Q}_n$ they are called *connectable* iff they are pairwise connectable. Again, in this setting the combined system $\mathcal{Q}_1 \mid \ldots \mid \mathcal{Q}_n$ is well-defined.

In the following, we group tapes into I/O and network tapes. This is to model either a secure direct connection or a network connection controlled by an adversary. For brevity, we define NET to be the set of all network tapes. Sometimes it is needed that two (or more) connectable systems $\mathcal{Q}$ and $\mathcal{R}$ only connect via external I/O tapes but not via external network tapes; such systems are called *I/O-connectable*.

Finally, two systems $\mathcal{Q}$ and $\mathcal{R}$ are called *compatible* if they have the same sets of (names of) external interfaces with the same directions.

C.1.4 Runtime Definitions

As already mentioned in §2, the IITM model, just as other universal composability models, ensures that systems run in polynomial time in the security parameter η and the length of the external input. This section formalizes this intuition by recalling the runtime definitions of the IITM model.

The IITM model ensures this with the three runtime notions of *almost bounded*, *universally bounded*, and *environmentally bounded* systems. These definitions differ in the setting they consider: A system $\mathcal{Q}$ is almost bounded if its runtime is bounded by a polynomial, it is universally bounded if its runtime is still bounded by a polynomial even when running with an arbitrary system $\mathcal{R}$, and it is environmentally bounded if the combined runtime of $\mathcal{Q}$ and any universally bounded system $\mathcal{R}$ is still bounded by a polynomial. Note that all of these notions consider runtime in mode **Compute** only, the **CheckAddress** mode is already bounded by the polynomial associated with the IITM. More formally:

Definition C.3. *A system $\mathcal{Q}$ is called* almost bounded *if there exists a polynomial p such that*

$$f(\eta, a) = \Pr\left[\mathit{Time}(\mathcal{Q}(1^\eta, a)) > p(\eta, |a|)\right] \text{for all } \eta \in \mathbb{N} \text{ and } a \in \{0,1\}^*$$

is negligible, where $\mathit{Time}(\mathcal{Q}(1^\eta, a))$ is the random variable which denotes the combined runtime of all instances of $\mathcal{Q}$ (in a run of $\mathcal{Q}(1^\eta, a)$) in mode **Compute**.
A system $\mathcal{Q}$ is called strictly bounded *if it is almost bounded for $f(\eta, a) = 0$.*

Definition C.4. *A system $\mathcal{E}$ is called* universally bounded *if there exists a polynomial p such that for every security paramater $\eta \in \mathbb{N}$, external input $a \in \{0,1\}^*$, any system $\mathcal{Q}$ that can be connected to $\mathcal{E}$ it holds true that the combined runtime in mode* **Compute** *of all instances of $\mathcal{E}$ in every run of $(\mathcal{E} \mid \mathcal{Q})(1^\eta, a)$ is at most $p(\eta, |a|)$.*

Definition C.5. *A system $\mathcal{P}$ is called* environmentally bounded *if for every universally bounded $\mathcal{E}$ that can be connected to $\mathcal{P}$ the system $\mathcal{E} \mid \mathcal{P}$ is almost bounded. A system $\mathcal{P}$ is called* environmentally strictly bounded *if for every universally bounded $\mathcal{E}$ that can be connected to $\mathcal{P}$ the system $\mathcal{E} \mid \mathcal{P}$ is strictly bounded.*

C.1.5 Environments, Adversaries, Protocols

To define security notions for universal composability, recall from §2 that there are three types of entities, namely environments, adversaries, and protocols, which can be modeled via systems of machines. This section precisely defines the properties of a system modeling one of these entities. This includes both runtime and connectivity requirements: Environments are required to be universally bounded, i.e., they will not exceed a polynomial runtime no matter in which system they are used. Protocols and adversaries only have to be polynomial when running with an (universally bounded) environment and hence when receiving inputs of at most polynomial length. Furthermore, these entities are not allowed to start or end a run since the environment is responsible for controlling the run.

Definition C.6. *A system $\mathcal{E}$ is called* environmental system *or just* environment *if it is universally bounded. For some system $\mathcal{Q}$, we denote by* $\mathsf{Env}(\mathcal{Q})$ *the set of all environments $\mathcal{E}$ that can be connected to $\mathcal{Q}$.*

Definition C.7. *A system $\mathcal{P}$ is called* protocol system *if (i) no tape in $\mathcal{P}$ is named* start *or* decision*, (ii) $\mathcal{P}$ is environmentally bounded and (iii) for every IITM M occuring 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**.[12]

Definition C.8. *A system $\mathcal{A}$ is called* adversarial system *if no tape of $\mathcal{A}$ is named* start *or* decision.[13]

Note that, as discussed in [22], the runtime notions of the IITM model allow for a very natural modeling of protocols without having to resort to workarounds such as (artificial) padding of messages, or adding new messages that only serve the purpose of transferring runtime to different machines. Furthermore, the runtime notion is very easy to check and it seems that all protocols seen in applications satisfy it.

C.2 Restricting Messages

We now formally define the term *restriction* from §4.2 in the IITM model. Recall that a *restriction* is a set of pairs of messages. The first component of a pair is a message that, when sent on a network tape by a protocol, has to be answered "immediately", i.e., an answer should be received on the corresponding network tape before any other machine of the protocol is activated on a different tape. The second component of a pair describes one possible answer. Additionally, we require that it is possible to efficiently decide whether an environment violates the restriction. More formally:

Definition C.9 (Restriction). *Let $R \subseteq \{0,1\}^+ \times \{0,1\}^+$ be a set of tuples of non-empty messages. Recall that $R[0] := \{m | \exists m' : (m,m') \in R\}$.*

The set R is called a restriction *if and only if the following holds true: There exists an algorithm A which for all inputs of the form (m,m') runs in at most polynomial time in the length of m' and outputs 1 iff $m \in R[0]$ and $(m,m') \notin R$, outputs 2 iff $(m,m') \in R$, and otherwise outputs 0. A message $m \in R[0]$ is called a* restricting message*.*

In the following, we will say that R is *decidable in polynomial time in the second component* or *efficiently decidable in the second component* when we refer to the property defined in the above definition. Note that this is different from the usual meaning of *decidable in polynomial time*: First, the algorithm A does not decide whether $(m,m') \in R$ but it decides whether, given some initial message m, a response m' is allowed by the restriction: If A outputs 0, the answer is allowed since $m \notin R[0]$, if it outputs 1, the answer is not allowed, and if it outputs 2, the answer is allowed iff it is sent on the correct tape. Second, A does not run in polynomial time in the length of (m,m') but only in the length of m'. This property is needed in the proof of Lemma C.11. Technically, in Lemma C.11 we can guarantee only that the length of the second component m' is polynomially bounded in the security parameter η and the external input a (if considered) as it is produced by the (universally bounded) environment; the first component m might not be as it is generated by the protocol, which in Lemma C.11 we do not require to be bounded in any way. Still the environment needs to be able to decide in polynomial time in η and a whether $m \in R[0]$ and whether $(m,m') \in R$. This property does not seem to be a strong requirement for practical purposes, as can be seen by the following example restriction:

[12] The intuition behind property (iii) is that the **CheckAddress** mode is meant to be used to address one of multiple instances, so if there is only a single instance of a machine in every run, this mode should not influence runs at all. This property is required in the proof of Theorem C.32.

[13] Note that, similar to environmental systems and protocol systems, there are also runtime conditions imposed on adversarial systems. However, these conditions depend on the system that is connected to $\mathcal{A}$ and thus will be presented in later definitions. Informally, if $\mathcal{A}$ is connected to another system $\mathcal{Q}$, then $\mathcal{A} \mid \mathcal{Q}$ will be required to be environmentally bounded.

$$
\begin{aligned}
R := \quad & \{(m, m') | m = (id, \texttt{AmICorrupted?}), \\
& \qquad m' = (id, \texttt{Corruption}, b), \\
& \qquad id \in \{0,1\}^*, b \in \{\mathsf{false}, \mathsf{true}\}\} \\
& \cup \{(m, m') | m = (id, \texttt{Info}, m''), \\
& \qquad m' = (id, \texttt{OK}), \\
& \qquad id, m'' \in \{0,1\}^*\} \\
& \cup \{(m, m') | m = (id, \texttt{Respond}, m''), \\
& \qquad m' = (id, m'''), \\
& \qquad id, m'', m''' \in \{0,1\}^*\},
\end{aligned}
$$

This restriction consists of three types of restricting messages, each identified by a fixed bit string. Each of the three types of restricting messages is prefixed with some identifier id, which has to be repeated in the answer. This can be used to ensure that the same instance which sent a message will also receive the answer, e.g., by defining mode **CheckAddress** such that every instance only accepts messages that are prefixed with its own (unique) ID. In fact, one of the main reasons why we did not simply hardwire a specific restriction into our framework is that one can adjust restrictions to different definitions of the **CheckAddress** mode. In the above example, the first type of restricting messages can be used to ask the adversary for the corruption state of a fresh instance upon its first activation, the adversary then has to provide a bit b immediately. The second type can be used to provide the adversary with some information such that the adversary has to give back control immediately by sending an acknowledgement. The third type is meant to exchange arbitrary data between the protocol and the adversary, i.e., it could be used to ask the adversary for some initialization such as keys and algorithms (the string m'' can be used to distinguish different requests), while the adversary can provide an arbitrary answer m''' but has to do so immediately. Note that this type is a variant of the generic restriction introduced in §6.2 and hence is actually sufficient to model all cases where restricting messages are needed. In particular, one can omit the first two types and only use the third type instead. We expect that restrictions will usually be variants of the generic restriction which have been adapted to the definition of the **CheckAddress** mode such that responses will be received by the correct instance.

This relation is in fact a restriction as defined in Definition C.9 for an appropriate encoding.[14] Given some initial message m and an answer m', it is possible to decide in polynomial time (in the length of m') whether m' is an incorrect response to a restricting message. The algorithm A first checks whether m is a restricting message by checking the number of components in tuples and reading components with a fixed length. Note that this is possible in constant time, since we assumed an appropriate encoding. If $m \notin R[0]$, then A outputs 0. Otherwise, A reads m' and checks whether m' is a possible answer to m, i.e., whether $(m, m') \in R$. Depending on the result, A outputs either 1 or 2. It is easy to see that A can do this in polynomial time in the length of m'.

In the following we assume that every IITM has a corresponding output tape for every input tape and vice versa (except for tapes named start and $\mathsf{decision}$), i.e., we actually consider pairs of tapes. For an input/output tape $t \notin \{\mathsf{start}, \mathsf{decision}\}$ of a machine, we denote by t^{-1} the corresponding output/input tape of the same machine such that (t, t^{-1}) is a pair of tapes. We require that two machines connect via tape pairs only. That is, if (t, t^{-1}) is a tape pair of a machine M, with t being an input tape and t^{-1} the corresponding output tape, and M connects to other machines, then if M connects to an output (input) tape of another machine M' with t (t^{-1}), then M also connects to the corresponding input (output) tape of M' with t^{-1} (t). The intuition behind this is that machines, after having received some input from a machine M, usually send a response back to M at some point, so they need a corresponding output tape. In particular there has to be a corresponding output tape for every input tape of the environment such that the environment is able to send responses to restricting messages to the correct machine.

[14] In particular, we need an encoding such that it is possible to check the number of components in a tuple and read single components without having to read the full string. Such an encoding can easily be established, so we will keep it implicit for simplicity of presentation.

Note that this is only a formal requirement, since every system $\mathcal{Q}$ that does not fulfill this requirement can easily be adjusted by adding some additional tapes. The behavior of $\mathcal{Q}$ does not change since these new tapes will never be used.

C.3 Strong Simulatability for Responsive Environments

We now define *responsive environments*, *protocols for responsive environments*, and *responsive simulators* from §4.2 in the IITM model. We also define the security notion *strong simulatability for responsive environments* in the IITM model with responsive environments.

Definition C.10 (Responsive environments). *Let R be a restriction. Let $\mathcal{P}$ be a system of IITMs and let $\mathcal{E} \in \mathsf{Env}(\mathcal{P})$ be an environment. We define the event E to be the set of all runs of $\mathcal{E} \mid \mathcal{P}$ where the following holds true: If $\mathcal{P}$ sends a restricting message $m \in R[0]$ on an external tape $t \in \mathsf{NET}$ and if there exists a message m' such that m' is the first message received on an external tape t' (not necessarily in NET) of $\mathcal{P}$ after m was sent, then $t' = t^{-1}$ and $(m, m') \in R$.*

The environment $\mathcal{E}$ is called a responsive environment *for $\mathcal{P}$ (with respect to R) if and only if $\Pr[E]$ is overwhelming. We denote the set of all responsive environments for an IITM system $\mathcal{P}$ by $\mathsf{Env}_R(\mathcal{P})$.*

Note that if a responsive environment receives a restricting message, it may still perform arbitrary computations and even send messages on tapes that are not connected to $\mathcal{P}$ (e.g. $\mathsf{decision}$ or any internal tape). Note also that this definition does not imply that $\mathcal{E}$ is actually connected to the tape t on which $\mathcal{P}$ sends a restricting message; it is entirely possible that a restricting message activates a master IITM with an empty string on tape start. Nevertheless, the definition guarantees that the next message to $\mathcal{P}$ will be on the corresponding answer tape and will be of a form that satisfies R (except with negligible probability). Typically, however, $\mathcal{E}$ will be connected to all external tapes of $\mathcal{P}$, where $\mathcal{P}$ could be a real protocol or a simulator connected to an ideal protocol.

In our proofs, we will often say that *"in a run of the system $\mathcal{E} \mid \mathcal{P}$ restricting messages from $\mathcal{P}$ are answered correctly"* to say that such a run belongs to the event E specified in Defintion C.10, i.e., after $\mathcal{P}$ sends a restricting message there is either no answer or the answer is of an expected form on the correct tape. Analogously we say that *"in a run of the system $\mathcal{E} \mid \mathcal{P}$ a restricting message from $\mathcal{P}$ is answered incorrectly"* to say that such a run does not belong to E, i.e., at least one restricting message was answered with an unexpected answer or on the wrong tape.

While responsive environments are defined with respect to a specific system $\mathcal{P}$, the following lemma shows that responsive environments are also responsive for systems that are indistinguishable from $\mathcal{P}$.

Lemma C.11. *Let R be a restriction. Let $\mathcal{P}$ and $\mathcal{P}'$ be two systems of IITMs such that neither of them has a start or $\mathsf{decision}$ tape, both systems have the same external interface, and $\mathcal{E} \mid \mathcal{P} \equiv \mathcal{E} \mid \mathcal{P}'$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$. Then, $\mathsf{Env}_R(\mathcal{P}) = \mathsf{Env}_R(\mathcal{P}')$.*[15]

Proof. Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$. We can assume that start is a tape of $\mathcal{E}$; otherwise there would be no master IITM and every run would directly terminate with empty output, which implies $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P}')$. If $\mathcal{E} \notin \mathsf{Env}_R(\mathcal{P}')$, then there must be a non-negligible set of runs of $\mathcal{E} \mid \mathcal{P}'$ where a restricting message of $\mathcal{P}'$ is not answered correctly. Let $\mathcal{E}' \in \mathsf{Env}(\mathcal{P})$ be the single IITM that accepts all messages in mode **CheckAddress** and then simulates $\mathcal{E}$. Note that this is possible by Lemma 6 from [22] which states that we can simulate every system with a single IITM that has the same external interface and accepts all messages in mode **CheckAddress**. Furthermore, we add additional tapes c to $\mathcal{E}'$ for each external network tape of $\mathcal{P}$ that does not connect to $\mathcal{E}$. The system $\mathcal{E}'$ also has a $\mathsf{decision}$ tape even if $\mathcal{E}$ does not have one. Note that these additional tapes do not interfere with the simulation: If $\mathcal{P}$ in a run of $\mathcal{E}|\mathcal{P}$ sends a message on an external network tape that is not

[15] Note that this lemma does not impose any runtime requirements on the systems $\mathcal{P}$ and $\mathcal{P}'$. We need such a general version of this lemma since there are often cases where we have constructed a new system $\mathcal{P}'$ from a system $\mathcal{P}$ and still have to prove that the runtime of $\mathcal{P}'$ is bounded when running with a responsive environment. To prove this, we first need the results of this lemma.

connected to $\mathcal{E}$, the master IITM (which is part of $\mathcal{E}$) will be activated with empty input. The system $\mathcal{E}'$ can easily simulate this as soon as a message on one of these new tapes is received.

Every time a message m' is about to be sent by $\mathcal{E}$ to $\mathcal{P}$ (or $\mathcal{P}'$), $\mathcal{E}'$ checks whether the condition of responsive environments is about to be violated by this message and, if that is the case, outputs 1 on decision instead. Note that this is easy to check: First, $\mathcal{E}'$ connects to all external network tapes of $\mathcal{P}$, i.e. $\mathcal{E}'$ "sees" all restricting messages m from $\mathcal{P}$. Second, because R is decidable in polynomial time in the second component, $\mathcal{E}'$ can use the algorithm A (see Definition C.9) to decide in polynomial time in $|m'|$ whether m' is an allowed answer: If A outputs 0, m' is allowed because $m \notin R[0]$. If A outputs 1, m' is an incorrect answer since $m \in R[0] \wedge (m, m') \notin R$. If A outputs 2, m' is an allowed answer iff it is sent on the correct tape, because $m \in R[0] \wedge (m, m') \in R$. Since m' was outputed by the simulated $\mathcal{E}$ and since $\mathcal{E}$ is universally bounded, $|m'|$ is bounded by a polynomial in the security parameter η and the length of the external input a (if any). So, overall A runs in polynomial time in η and $|a|$. If $\mathcal{E}$ wants to output something on decision or the master IITM of $\mathcal{E}$ stops with empty output, $\mathcal{E}'$ outputs 0 on decision instead. Now it is easy to see that $\mathcal{E}'$ is universally bounded since $\mathcal{E}'$ only simulates the universally bounded system $\mathcal{E}$ and additionally performs some checks which can be carried out in polynomial time. In particular, since $\mathcal{E}'$ uses the algorithm A to decide whether it must ouput 1, it does not have to read the entire messages m that are sent by $\mathcal{P}$ but only polynomial many bits (as many as required by the polynomial time algorithm A and the universally bounded system $\mathcal{E}$).

Clearly, we have that $\mathcal{E}' \in \mathsf{Env}_R(\mathcal{P})$ since $\mathcal{E}'$ ends the run before a restricting message from $\mathcal{P}$ is answered incorrectly. Furthermore, since every restricting message from $\mathcal{P}$ (or $\mathcal{P}'$) will activate the simulated $\mathcal{E}$ (either on a connected tape or on tape start), it is easy to see that $\mathcal{E}'$ will output 1 if and only if a restricting message from $\mathcal{P}$ or $\mathcal{P}'$ in the simulated run of $\mathcal{E} \,|\, \mathcal{P}$ or $\mathcal{E} \,|\, \mathcal{P}'$, respectively, is answered incorrectly. In other words, $\mathcal{E}'$ is able to "see" all incorrect answers since $\mathcal{E}$ must have been the system that sent the incorrect answer. Overall, because $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ and $\mathcal{E} \notin \mathsf{Env}_R(\mathcal{P}')$, we have that $\mathcal{E}' \,|\, \mathcal{P} \not\equiv \mathcal{E}' \,|\, \mathcal{P}'$ in contradiction to the assumption that $\mathcal{E} \,|\, \mathcal{P} \equiv \mathcal{E} \,|\, \mathcal{P}'$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ and the fact that $\mathcal{E}' \in \mathsf{Env}_R(\mathcal{P})$. The other inclusion follows analogously. □

Since we will consider responsive environments only in this work, it is not necessary (and generally not even possible) to make any runtime guarantees for a system $\mathcal{P}$ if combined with an arbitrary environment (i.e. any universally bounded system that can be connected to $\mathcal{P}$). This is why we need a weaker notion of environmentally bounded systems.

Definition C.12 (R-environmentally bounded systems). *Let $\mathcal{P}$ be a system of IITMs. Then $\mathcal{P}$ is called* environmentally bounded for responsive environments *or simply* R-environmentally bounded *if and only if $\mathcal{E} \,|\, \mathcal{P}$ is almost bounded for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$.*
The system $\mathcal{P}$ is called environmentally strictly bounded for responsive environments *or simply* R-environmentally strictly bounded *if and only if $\mathcal{E} \,|\, \mathcal{P}$ is strictly bounded for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$.*

Definition C.13 (Protocol systems for responsive environments). *Let R be a restriction. Let $\mathcal{P}$ be a system of IITMs. Then $\mathcal{P}$ is called a* protocol system for responsive environments *if and only if $\mathcal{P}$ has no tapes named* start *or* decision*, $\mathcal{P}$ is R-environmentally bounded, and every IITM in $\mathcal{P}$ that is not in the scope of a bang accepts all messages in mode* **CheckAddress**.

The only difference between *protocol systems* and *protocol systems for responsive environments* is that the runtime condition of the latter is relaxed, i.e. the set of all protocol systems for responsive environments is a superset of the set of all protocol systems. From now on, we will say *environmentally bounded protocol system* to denote a protocol system in the original sense and we will simply say *protocol system* to denote a protocol system for responsive environments since the latter systems will be the default from now on.

Definition C.14 (Responsive simulators). *Let R be a restriction. Let $\mathcal{P}, \mathcal{F}$ be protocol systems. Let $\mathcal{S}$ be an adversarial system 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 $\mathcal{F}$, $\mathcal{S} \,|\, \mathcal{F}$ and $\mathcal{P}$ have the same external interface, and $\mathcal{S} \,|\, \mathcal{F}$ is R-environmentally bounded.*

Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ be a responsive environment. We define the event E to be the set of all runs of $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}$ where the following holds true: If $\mathcal{F}$ sends a restricting message $m \in R[0]$ on an external tape $t \in \mathsf{NET}$ and if

there exists a message m' such that m' is the first message received on an external tape t' of $\mathcal{F}$ after m was sent, then $t' = t^{-1}$ and $(m, m') \in R$.

The simulator $\mathcal{S}$ is called a responsive simulator *(with respect to R) if and only if $\Pr[E]$ is overwhelming for all responsive environments $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$. We denote the set of all responsive simulators for protocol systems $\mathcal{P}$ and $\mathcal{F}$ by $\mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})$.*

This definition ensures that restricting messages from $\mathcal{F}$ are answered without activating another machine of $\mathcal{F}$ (and with an expected response), even if $\mathcal{F}$ is connected to a simulator (on its network interface). Analogous to responsive environments, the terms *restricting messages are answered correctly* and *a restricting message is answered incorrectly* can also be defined for responsive simulators.

Note that the definition of the event E uses $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ instead of $\mathcal{E} \in \mathsf{Env}_R(\mathcal{S} \mid \mathcal{F})$. This is motivated by the fact that $\mathcal{S}$ should be responsive for all responsive environments of $\mathcal{P}$, which is supposed to realize $\mathcal{F}$. The simulator $\mathcal{S}$ should not restrict this set. In fact, good simulators do not restrict this set: We will use responsive simulators for settings where we have/require $\mathcal{E} \mid \mathcal{P} \equiv \mathcal{E} \mid \mathcal{S} \mid \mathcal{F}$ for every $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$. Thus, by Lemma C.11, in these settings it holds true that $\mathsf{Env}_R(\mathcal{P}) = \mathsf{Env}_R(\mathcal{S} \mid \mathcal{F})$. This also motivates why we could use $\mathcal{E} \in \mathsf{Env}_R(\mathcal{S} \mid \mathcal{F})$ instead of $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ for simulators in Definition 4.6 in the main body. Since both sets are the same in this setting, it was simpler to reuse the definition of responsive adversaries instead of separately defining responsive simulators.

Now we can define strong simulatability for responsive environments.

Definition C.15 (Strong Simulatability with responsive environments). *Let R be a restriction. Let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems, the* real *and* ideal protocol*, respectively. Then, $\mathcal{P}$* realizes *$\mathcal{F}$ with respect to* responsive environments *($\mathcal{P} \leq_R \mathcal{F}$) if and only if there exists a responsive simulator $\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})$ such that $\mathcal{E} \mid \mathcal{P} \equiv \mathcal{E} \mid \mathcal{S} \mid \mathcal{F}$ for every responsive environment $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$.*

As shown in Section C.4, being responsive is typically a property that is easy to see/prove for simulators. First, we will prove some fundamental properties of $\leq_R$, namely reflexivity and transitivity. For this purpose, we first have to recall two lemmas that were originally shown in [22]. Since in [22] the lemmas were not defined with respect to responsive environments, we adjust them and say where the original proofs have to be changed. We also prove two new lemmas that are based on the other lemmas.

In the following lemma, for a system $\mathcal{R} \mid \mathcal{Q}$ where $\mathcal{R}$ contains a master IITM, we consider an IITM, denoted by $[\mathcal{R}]_{\mathcal{Q}}$, which simulates $\mathcal{R}$ up to the runtime bound of $\mathcal{R} \mid \mathcal{Q}$ and then, if the runtime bound is exceeded, terminates the run with empty output.

Lemma C.16 (Lemma 7 from [22]). *Let $\mathcal{R}$ and $\mathcal{Q}$ be connectable systems such that $\mathcal{R} \mid \mathcal{Q}$ is almost bounded and* start *is a tape of $\mathcal{R}$ (i.e. $\mathcal{R}$ contains a master IITM, and hence, $\mathcal{Q}$ does not). Then there exists an IITM $[\mathcal{R}]_{\mathcal{Q}}$ such that the following conditions are satisfied:*

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

Lemma C.17 in combination with Lemma C.19 says that $\mathcal{E} \mid \mathcal{S}$ is a responsive environment for $\mathcal{F}$. We require simulators to be responsive in order to get this property. Without requiring simulators to be responsive, one could not move a simulator into the environment and still hope to obtain a responsive environment, a property, which, however, is important in many places, such as the transitivity of $\leq_R$ and the composition theorems.

Lemma C.17. *Let R be a restriction. Let $\mathcal{P}, \mathcal{F}$ be protocol systems with $\mathcal{P} \leq_R \mathcal{F}$. Let $\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})$ be the responsive simulator that is used in the definition of $\mathcal{P} \leq_R \mathcal{F}$. Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ be a responsive environment such that* start *is a tape of $\mathcal{E}$. Then $[\mathcal{E} \mid \mathcal{S}]_{\mathcal{F}} \in \mathsf{Env}_R(\mathcal{F})$.*

Proof. Let $\mathcal{P}, \mathcal{F}, \mathcal{S}, \mathcal{E}$ be systems as required by the lemma. First, we observe that Lemma C.16 can be applied to $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}$: Since $\mathcal{S} \,|\, \mathcal{F}$ is R-environmentally bounded and $\mathcal{E} \in \mathsf{Env}_R(\mathcal{S} \,|\, \mathcal{F})$ by Lemma C.11, we have that $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}$ is almost bounded and $\mathcal{E} \,|\, \mathcal{S}$ contains a master IITM.
By Lemma C.16 we know that $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{F}}$ is universally bounded, and hence, $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{F}} \in \mathsf{Env}(\mathcal{F})$. Suppose there was a non-negligible set of runs of $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{F}} \,|\, \mathcal{F}$ where a restricting message of $\mathcal{F}$ is not answered correctly. Since $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{F}} \,|\, \mathcal{F}$ and $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}$ are the same except for a negligible set of runs (where a certain runtime is reached), it follows that there must also be a non-negligible set of runs of $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}$ where a restricting message from $\mathcal{F}$ is not answered correctly. However, this is a contradiction to $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_R(\mathcal{F})$. This implies that restricting messages from $\mathcal{F}$ are answered correctly in an overwhelming set of runs of $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{F}} \,|\, \mathcal{F}$, i.e. $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{F}} \in \mathsf{Env}_R(\mathcal{F})$. □

Note that Lemma C.17 is a special case of the following more general lemma.

Lemma C.18. *Let R be a restriction and let $\mathcal{R}, \mathcal{Q}$ be connectable systems of IITMs such that $\mathcal{R} \,|\, \mathcal{Q}$ is almost bounded and* start *is a tape of $\mathcal{R}$, then $[\mathcal{R}]_{\mathcal{Q}} \in \mathsf{Env}_R(\mathcal{Q})$ if restricting messages from $\mathcal{Q}$ are answered correctly in an overwhelming set of runs of $\mathcal{R} \,|\, \mathcal{Q}$.*

Proof. The proof is analogous to the proof of Lemma C.17. □

The following lemma corresponds to Lemma 8 in [22].

Lemma C.19. *Let R be a restriction. Let $\mathcal{R}$ and $\mathcal{Q}$ be connectable systems such that $\mathcal{R} \,|\, \mathcal{Q}$ is almost bounded,* start *is a tape of $\mathcal{R}$ (i.e. $\mathcal{R}$ contains a master IITM, and hence, $\mathcal{Q}$ does not), and* decision *is not a tape of $\mathcal{Q}$. Furthermore, let $\mathcal{Q}'$ be a system which is compatible with $\mathcal{Q}$ and satisfies the following condition: $\mathcal{E} \,|\, \mathcal{Q} \equiv \mathcal{E} \,|\, \mathcal{Q}'$ for every $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q})$ such that $\mathcal{E} \,|\, \mathcal{Q}$ is almost bounded. If $[\mathcal{R}]_{\mathcal{Q}} \in \mathsf{Env}_R(\mathcal{Q})$, then*

$$[\mathcal{R}]_{\mathcal{Q}} \,|\, \mathcal{Q}' \equiv \mathcal{R} \,|\, \mathcal{Q}' \ .$$

Moreover, if $[\mathcal{R}]_{\mathcal{Q}} \in \mathsf{Env}_R(\mathcal{Q})$ and $[\mathcal{R}]_{\mathcal{Q}} \,|\, \mathcal{Q}'$ is almost bounded, then $\mathcal{R} \,|\, \mathcal{Q}'$ is almost bounded too.

Proof. We may assume that $\mathcal{R}$ is a single IITM that accepts every message in mode **CheckAddress** (by Lemma 6 from [22]). Now, recall that by definition, $[\mathcal{R}]_{\mathcal{Q}}$ exactly simulates all transitions of $\mathcal{R}$ up to a certain polynomial bound and that when running $[\mathcal{R}]_{\mathcal{Q}}|\mathcal{Q}$ this bound is exceeded with negligible probability only. If $[\mathcal{R}]_{\mathcal{Q}} \in \mathsf{Env}_R(\mathcal{Q})$, then the probability that this bound is exceeded when running the system $[\mathcal{R}]_{\mathcal{Q}}|\mathcal{Q}'$ is negligible as well. Otherwise, one could easily construct a system $\mathcal{E}$ such that $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q})$, $\mathcal{E}|\mathcal{Q}$ is almost bounded, and $\mathcal{E}|\mathcal{Q} \not\equiv \mathcal{E}|\mathcal{Q}'$, in contradiction to the assumption: The system $\mathcal{E}$ is defined to simulate $[\mathcal{R}]_{\mathcal{Q}}$ and output 1 on decision if and only if the bound is exceeded (note that this is only possible since decision is not a tape of $\mathcal{Q}$ and $\mathcal{Q}'$, respectively). Since we assume $[\mathcal{R}]_{\mathcal{Q}} \in \mathsf{Env}_R(\mathcal{Q})$, we directly obtain that $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q})$. Furthermore, since $[\mathcal{R}]_{\mathcal{Q}}|\mathcal{Q}$ is almost bounded, it is easy to see that $\mathcal{E}|\mathcal{Q}$ is also almost bounded. Observe that 1 is output on decision in runs of $\mathcal{E}|\mathcal{Q}$ and $\mathcal{E}|\mathcal{Q}'$, respectively, if and only if the bound is exceeded.

It follows that with overwhelming probability $[\mathcal{R}]_{\mathcal{Q}}$ exactly simulates $\mathcal{R}$ in the system $[\mathcal{R}]_{\mathcal{Q}}|\mathcal{Q}'$. Thus, we obtain $[\mathcal{R}]_{\mathcal{Q}}|\mathcal{Q}' \equiv \mathcal{R}|\mathcal{Q}'$. Similarly, it is easy to see that if $[\mathcal{R}]_{\mathcal{Q}}|\mathcal{Q}'$ is almost bounded (and $[\mathcal{R}]_{\mathcal{Q}} \in \mathsf{Env}_R(\mathcal{Q})$), then $\mathcal{R}|\mathcal{Q}'$ is almost bounded too.

□

We can now proof reflexivity and transitivity of $\leq_R$.

Lemma C.20. *$\leq_R$ is reflexive and transitive.*

Proof. Reflexivity: Let $\mathcal{P}$ be a protocol system. Let $\mathcal{S}$ be a single IITM without external tapes that does nothing. This directly implies that $\mathcal{S}$ fulfills the conditions of responsive simulators regarding the interfaces and runtime. Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ be a responsive environment. We have $\mathcal{E} \,|\, \mathcal{P} \equiv \mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{P}$ because $\mathcal{S}$ cannot interact with the other machines, i.e. $\mathcal{S}$ does not influence a run in any way. Moreover, since $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ and $\mathcal{S}$ is not involved in the interaction between $\mathcal{E}$ and $\mathcal{P}$, we have that restricting messages sent from $\mathcal{P}$ to $\mathcal{E}$ are answered correctly by $\mathcal{E}$. Hence, we obtain $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_R(\mathcal{P})$. Overall, we have that $\mathcal{P} \leq_R \mathcal{P}$.

Transitivity: Let $\mathcal{P}, \mathcal{P}', \mathcal{P}''$ be protocol systems with $\mathcal{P} \leq_R \mathcal{P}'$ and $\mathcal{P}' \leq_R \mathcal{P}''$. Without loss of generality, we can assume that $\mathcal{P}, \mathcal{P}'$ and $\mathcal{P}''$ have pairwise disjoint sets of (external) network tapes. This is possible since, if a simulator exists for two systems which share some network tape names, we can rename the tapes of one system and define a new simulator that behaves just as the original one but additionally forwards messages between the renamed tapes and the original tapes. Let $\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{P}')$ and $\mathcal{S}' \in \mathrm{Sim}_R^{\mathcal{P}'}(\mathcal{P}'')$ be the simulators that are used in the definition of $\mathcal{P} \leq_R \mathcal{P}'$ and $\mathcal{P}' \leq_R \mathcal{P}''$, respectively. Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ be a responsive environment. We may assume that $\mathcal{E}$ has a start tape; otherwise there would be no master IITM and the run would always directly terminate with empty output, in which case transitivity is trivially fulfilled. Since $\mathcal{E}$ contains a master IITM we can use Lemma C.16 to get $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{P}' \equiv [\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}'} \,|\, \mathcal{P}'$. By Lemma C.17 we have $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}'} \in \mathsf{Env}_R(\mathcal{P}')$ and thus $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}'} \,|\, \mathcal{P}' \equiv [\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}'} \,|\, \mathcal{S}' \,|\, \mathcal{P}''$. Lemma C.19 implies $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}'} \,|\, \mathcal{S}' \,|\, \mathcal{P}'' \equiv \mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{S}' \,|\, \mathcal{P}''$. Note that $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{S}' \,|\, \mathcal{P}''$ is well defined, i.e., all systems are actually connectable by the initial assumption that the sets of external network tapes of $\mathcal{P}, \mathcal{P}'$, and $\mathcal{P}''$ are pairwise disjoint. Overall we have that $\mathcal{E} \,|\, \mathcal{P} \equiv \mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{S}' \,|\, \mathcal{P}''$.

It remains to show that $\mathcal{S} \,|\, \mathcal{S}' \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{P}'')$. It is easy to see that the conditions regarding interfaces of responsive simulators are fulfilled. Since $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}'} \in \mathsf{Env}_R(\mathcal{S}' \,|\, \mathcal{P}'')$ by Lemma C.11, we have that $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}'} \,|\, \mathcal{S}' \,|\, \mathcal{P}''$ is almost bounded, and thus, by Lemma C.19, $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{S}' \,|\, \mathcal{P}''$ is also almost bounded. This holds for every $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ and thus for every $\mathcal{E} \in \mathsf{Env}_R(\mathcal{S} \,|\, \mathcal{S}' \,|\, \mathcal{P}'')$ by Lemma C.11, so $\mathcal{S} \,|\, \mathcal{S}' \,|\, \mathcal{P}''$ is R-environmentally bounded. Let the event E be the set of all runs of $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{S}' \,|\, \mathcal{P}''$ where a restricting message from $\mathcal{P}''$ is answered on the wrong tape or with an unexpected response. If $\Pr[E]$ is non-negligible, then there must also be a non-negligible set of runs of $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}'} \,|\, \mathcal{S}' \,|\, \mathcal{P}''$ where a restricting message from $\mathcal{P}$ is not answered correctly.[16] However, this is a contradiction to $\mathcal{S}'$ being a responsive simulator, since $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}'}$ is a responsive environment of $\mathcal{P}'$. Altogether, this implies $\mathcal{S} \,|\, \mathcal{S}' \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{P}'')$, and thus, $\mathcal{P} \leq_R \mathcal{P}''$.

□

C.4 Proving Responsiveness of Simulators

According to the definition of strong simulatability (Definition C.15), for $\mathcal{P} \leq_R \mathcal{F}$ to hold one has to prove that there exists a simulator $\mathcal{S}$ which fulfills i) certain restrictions concerning connectivity and interfaces, ii) certain runtime conditions, iii) the responsiveness condition (i.e., restricting messages from $\mathcal{F}$ are answered correctly by $\mathcal{S}$), and most importantly iv) the equivalence $\mathcal{E} \,|\, \mathcal{P} \equiv \mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$. Conditions i) and ii) are easy to check. The following lemma shows that iii) can be checked easily as well. According to the definition, one has to show that if $\mathcal{S}$ receives a restricting message from $\mathcal{F}$, then $\mathcal{S}$ has to make sure that this message is answered correctly according to the restriction R. So if $\mathcal{S}$ sends out a message on an external tape, then there are two options: either this message is sent to $\mathcal{F}$, and in this case, this has to be the correct response according to R, or this message is sent to $\mathcal{E}$, but then this message has to be a restricting message for $\mathcal{E}$ in order to make sure that $\mathcal{E}$ answers directly (to $\mathcal{S}$). If $\mathcal{S}$ sent a non-restricting message to $\mathcal{E}$, then $\mathcal{E}$ would be free to send a message to $\mathcal{F}$, which would violate the responsiveness property that $\mathcal{S}$ has to fulfill. Whether or not $\mathcal{S}$ always sends such restricting messages (either to $\mathcal{F}$ or to $\mathcal{E}$) should be easy to check given the specification of $\mathcal{S}$. Therefore, condition iii) should be easy to check. This is made precise in the following lemma.

Lemma C.21. *Let R be a restriction. Let $\mathcal{P}, \mathcal{F}$ be protocol systems. Let $\mathcal{S}$ be an adversarial system 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 $\mathcal{F}$, $\mathcal{S} \,|\, \mathcal{F}$ and $\mathcal{P}$ have the same external interface, and $\mathcal{S} \,|\, \mathcal{F}$ is R-environmentally bounded.*

Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ be a responsive environment. We define the event $E_{\mathcal{E}}$ to be the set of all runs of $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}$ where the following holds true: The system $\mathcal{S}$ accepts all restricting messages from $\mathcal{F}$ in mode **CheckAddress**. *If $\mathcal{S}$ receives a restricting message from $\mathcal{F}$, it does not produce empty output until it has sent a message to $\mathcal{F}$. Furthermore, if $\mathcal{S}$ sends a message on an external tape after having received a restricting message m on tape t from $\mathcal{F}$ and before having sent an answer to $\mathcal{F}$, the message is of one of two types: either a restricting*

[16] Per construction of $[\mathcal{E} \,|\, \mathcal{S}]_{\mathcal{P}'}$ it works just like $\mathcal{E} \,|\, \mathcal{S}$ except for cases where a certain runtime is reached. However, this runtime is only reached by $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{S}' \,|\, \mathcal{P}''$ in a negligible set of runs (see the proof of Lemma C.19). Since both systems only differ in a negligible set of runs, the statement follows.

message n on an external tape of $\mathcal{S} \,|\, \mathcal{F}$ (i.e. to the environment) where all possible answers n' with $(n, n') \in R$ will be accepted by $\mathcal{S}$ in mode **CheckAddress**, *or a message m' on tape t^{-1} with $(m, m') \in R$.*

Then, $\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})$ and $\mathcal{P} \leq_R \mathcal{F}$ if, for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$, we have that (I) $\mathcal{E} \,|\, \mathcal{P} \equiv \mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}$ and (II) the event $E_{\mathcal{E}}$ is overwhelming.

Proof. Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ be a responsive environment and let $\mathcal{S}$ be given as above such that (I) and (II) are satisfied. In order to show $\mathcal{P} \leq_R \mathcal{F}$, we have to prove only that $\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})$. For this, it remains to show that $\mathcal{S}$ fulfills the responsiveness condition, i.e., restricting messages from $\mathcal{F}$ are answered correctly in an overwhelming set of runs of $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}$. First, by Lemma C.11 we have $\mathcal{E} \in \mathsf{Env}_R(\mathcal{S} \,|\, \mathcal{F})$. Let E' be the set of all runs of $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}$ where a restricting message from $\mathcal{F}$ is answered incorrectly, let E'' be the subset of E' where the first restricting message from $\mathcal{F}$ that is not answered correctly is sent on a network tape that is not connected to $\mathcal{S}$, and let E''' be the subset of E' where the first restricting message from $\mathcal{F}$ that is not answered correctly is sent on a network tape that is connected to $\mathcal{S}$. Obviously, E'' and E''' are disjoint, $E' = E'' \cup E'''$, and hence, $\Pr[E'] = \Pr[E''] + \Pr[E''']$.

The event E'' is a subset of all runs of $\mathcal{E} \,|\, \mathcal{S} \,|\, \mathcal{F}$ where a restricting message from $\mathcal{S} \,|\, \mathcal{F}$ is answered incorrectly. Since $\mathcal{E} \in \mathsf{Env}_R(\mathcal{S} \,|\, \mathcal{F})$, this set of runs is negligible, and thus, $\Pr[E'']$ must also be negligible.

The event E''' can be devided into two (distinct) subsets E'''_0 and E'''_1, where we define E'''_0 to contain only those runs of E''' which belong to $E_{\mathcal{E}}$ (as defined in the lemma) and E'''_1 contains the remaining runs of E'''. Since, by assumption, $E_{\mathcal{E}}$ is overwhelming, we have that E'''_1 is negligible. Now, we further divide E'''_0 into two (distinct) subsets E'''_{0a} and E'''_{0b}, where we define E'''_{0a} to contain only those runs of E'''_0 where restricting messages from $\mathcal{S}$ on an external tape of $\mathcal{S} \,|\, \mathcal{F}$ are answered correctly and E'''_{0b} contains the remaining runs of E'''_0. Again, we directly obtain that E'''_{0b} is negligible since $\mathcal{E} \in \mathsf{Env}_R(\mathcal{S} \,|\, \mathcal{F})$.

We now show that E'''_{0a} is the empty set, and hence, $\Pr[E'''_{0a}] = 0$. Assume that E'''_{0a} contains a run. Consider the first restricting message from $\mathcal{F}$ sent to $\mathcal{S}$ that is answered incorrectly. Such a message exists since $E'''_{0a} \subseteq E'''$. Since $E'''_{0a} \subseteq E_{\mathcal{E}}$, $\mathcal{S}$ accepts this message in mode **CheckAddress** and thus gets activated. Since $\mathcal{S}$ may not produce empty output, it can only end its activation by sending one of two kinds of messages. If it sends a restricting message to $\mathcal{E}$, the next message to $\mathcal{S} \,|\, \mathcal{F}$ will be on a tape of $\mathcal{S}$ (otherwise the run would be in E'''_{0b}). Since $\mathcal{S}$ must accept this message in mode **CheckAddress** (by definition of $E_{\mathcal{E}}$) the system $\mathcal{S}$ will get activated and, again, it may only end this activation by sending one of two kinds of messages. If $\mathcal{S}$ sends a message to $\mathcal{F}$, it will be of such a form that the answer to the restricting message is correct; otherwise this run would be in E'''_1. Hence, the restricting message from $\mathcal{F}$ is actually answered correctly, a contradiction. This implies that $E'''_{0a} = \emptyset$.

Altogether, we have that $\Pr[E'] = \Pr[E''] + \Pr[E'''_1] + \Pr[E'''_{0a}] + \Pr[E'''_{0b}]$, where all probabilities on the right-hand side of the equation are negligible. Hence, the claim follows. □

We note that $\mathcal{S}$ might send several restricting messages to $\mathcal{E}$ until it sends an answer to $\mathcal{F}$.

C.5 Notions of Universal Composability

In the literature, besides strong simulatability, often the security notions UC, dummy UC, and black-box simulatability are used. In [22], it was shown that these notions are equivalent to strong simulatability, which justifies the use of the coneptually simpler notion of strong simulatability. Furthermore, in [22], equivalence to reactive simulatability, a notion which is not used in the literature, was shown as well; however, this equivalence holds true only with respect to non-uniform environments. In this section, we show that these notions are still equivalent with respect to responsive environments. Hence, again, it suffices to consider strong simulatability only. First, we have to define the other security notions. For this purpose, we have to introduce responsive adversaries and the (responsive) dummy adversary.

Definition C.22 (Responsive adversaries). *Let R be a restriction. Let $\mathcal{P}$ be a protocol system. Let $\mathcal{A}$ be an adversarial system such that $\mathcal{A}$ can be connected to $\mathcal{P}$, the set of external tapes of $\mathcal{A}$ is disjoint from the set of I/O-tapes of $\mathcal{P}$, $\mathcal{A}$ connects to all external network tapes of $\mathcal{P}$, and $\mathcal{A}|\mathcal{P}$ is R-environmentally bounded.*

Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{A}|\mathcal{P})$ be a responsive environment. Define the event E to be the set of all runs of $\mathcal{E}|\mathcal{A}|\mathcal{P}$ where the following holds true: If $\mathcal{P}$ sends a restricting message $m \in R[0]$ on an external tape $t \in \mathsf{NET}$ and if

there exists a message m' such that m' is the first message received on an external tape t' of $\mathcal{P}$ after m was sent, then $t' = t^{-1}$ and $(m, m') \in R$.

The adversarial system $\mathcal{A}$ is called a responsive adversary *(with respect to R) if and only if $\Pr[E]$ is overwhelming for all responsive environments $\mathcal{E} \in \mathsf{Env}_R(\mathcal{A}|\mathcal{P})$. We denote the set of all responsive adversaries for a protocol system $\mathcal{P}$ by $\mathrm{Adv}_R(\mathcal{P})$.*

Note that responsive adversaries are quite similar to responsive simulators since both definitions only differ in the requirements for the external interface of $\mathcal{A}|\mathcal{P}$ and $\mathcal{S}|\mathcal{F}$, respectively. The idea behind this definition is that an adversary should also answer restricting messages from $\mathcal{P}$ correctly, i.e. he should not be more powerful than a responsive environment.

In the following, we will also need a special responsive adversary: Given a protocol system $\mathcal{P}$, by $D_{\mathcal{P}}$ we denote the *dummy adversary for* $\mathcal{P}$ which connects to all network tapes of $\mathcal{P}$ and has one corresponding network tape per network tape of $\mathcal{P}$. The dummy adversary simply forwards all messages between a tape of $\mathcal{P}$ and the corresponding tape. Note that the connectivity conditions of responsive adversaries are fulfilled and that $D_{\mathcal{P}}|\mathcal{P}$ is R-environmentally bounded since $\mathcal{P}$ is R-environmentally bounded. Furthermore, it is easy to see that restricting messages of $\mathcal{P}$ are answered correctly in an overwhelming set of runs, since the dummy adversary forwards the same restricting message to a responsive environment on a network tape. Thus, we have $D_{\mathcal{P}} \in \mathrm{Adv}_R(\mathcal{P})$.

Definition C.23. *Let R be a restriction. Let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems, the* real *and* ideal protocol.[17]

1. Strong Simulatability with responsive environments (SS_R):
   $\mathcal{P} \leq_R^{SS} \mathcal{F}$ *iff* $\exists\, \mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})\ \forall\, \mathcal{E} \in \mathsf{Env}_R(\mathcal{A}|\mathcal{P})$*:*
   $$\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{S}|\mathcal{F} \ .$$
2. Universal Simulatability/Composability with responsive environments (UC_R):
   $\mathcal{P} \leq_R^{UC} \mathcal{F}$ *iff* $\forall\, \mathcal{A} \in \mathrm{Adv}_R(\mathcal{P})\ \exists\, \mathcal{I} \in \mathrm{Sim}_R^{\mathcal{A}|\mathcal{P}}(\mathcal{F})\ \forall\, \mathcal{E} \in \mathsf{Env}_R(\mathcal{A}|\mathcal{P})$*:*
   $$\mathcal{E}|\mathcal{A}|\mathcal{P} \equiv \mathcal{E}|\mathcal{I}|\mathcal{F} \ .$$
3. Dummy Version of UC with responsive environments ($\mathrm{dummyUC}_R$):
   $\mathcal{P} \leq_R^{dumUC} \mathcal{F}$ *iff* $\exists\, \mathcal{I} \in \mathrm{Sim}_R^{D_{\mathcal{P}}|\mathcal{P}}(\mathcal{F})\ \forall\, \mathcal{E} \in \mathsf{Env}_R(D_{\mathcal{P}}|\mathcal{P})$*:*
   $$\mathcal{E}|D_{\mathcal{P}}|\mathcal{P} \equiv \mathcal{E}|\mathcal{I}|\mathcal{F} \ .$$
4. Black-box Simulatability with responsive environments (BB_R):
   $\mathcal{P} \leq_R^{BB} \mathcal{F}$ *iff* $\exists\, \mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})\ \forall\, \mathcal{A} \in \mathrm{Adv}_R(\mathcal{P})\ \forall\, \mathcal{E} \in \mathsf{Env}_R(\mathcal{A}|\mathcal{P})$*:*
   $$\mathcal{E}|\mathcal{A}|\mathcal{P} \equiv \mathcal{E}|\mathcal{A}|\mathcal{S}|\mathcal{F} \ .$$
5. Reactive Simulatability with responsive environments (RS_R):
   $\mathcal{P} \leq_R^{RS} \mathcal{F}$ *iff* $\forall\, \mathcal{A} \in \mathrm{Adv}_R(\mathcal{P})\ \forall\, \mathcal{E} \in \mathsf{Env}_R(\mathcal{A}|\mathcal{P})\ \exists\, \mathcal{I} \in \mathrm{Sim}_R^{\mathcal{A}|\mathcal{P}}(\mathcal{F})$*:*
   $$\mathcal{E}|\mathcal{A}|\mathcal{P} \equiv \mathcal{E}|\mathcal{I}|\mathcal{F} \ .$$

For strong and black-box simulatability, if $\mathcal{P}$ and $\mathcal{F}$ do not have disjoint network tapes, there might not exist a simulator such that $\mathcal{P}$ and $\mathcal{S} \,|\, \mathcal{F}$ are indistinguishable since $\mathcal{S}$ cannot connect to all network tapes of $\mathcal{F}$ in this case. In what follows, 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. Note that this is only a formal requirement. It does not influence the behavior of $\mathcal{P}$ and $\mathcal{F}$ in any way.

We will now prove that strong simulatability with responsive environments is equivalent to the other notions of Definition C.23. We note that, except for the equivalence of strong simulatability and reactive simulatability, this theorem holds true also for uniform environments, i.e., environments that do not obtain external input. As mentioned before, an analogous result was originally proven in [22] for arbitrary environments.

[17] Strong simulatability was defined in Definition C.15 already. We here list strong simulatability again to have a complete list of the security notions we consider.

Theorem C.24. *Let R be a restriction. Let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems. Then:*

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

Proof. Let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems.

$\mathcal{P} \leq_R^{SS} \mathcal{F}$ *iff* $\mathcal{P} \leq_R^{UC} \mathcal{F}$ *iff* $\mathcal{P} \leq_R^{dumUC} \mathcal{F}$: We will first show the equivalence of SS_R, UC_R, and $\mathrm{dummyUC}_R$. Since UC_R trivially implies $\mathrm{dummyUC}_R$, we only have to show that SS_R implies UC_R and that $\mathrm{dummyUC}_R$ implies SS_R.

Let $\mathcal{P} \leq_R^{SS} \mathcal{F}$. Then there exists a responsive simulator $\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})$ such that $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{S}|\mathcal{F}$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$. Let $\mathcal{A} \in \mathrm{Adv}_R(\mathcal{P})$ be a responsive adversary and let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{A}|\mathcal{P})$ be a responsive environment. We define $\mathcal{I} := \mathcal{A}|\mathcal{S}$. We may assume that start is a tape of $\mathcal{E}$; if this was not the case, then there would be no master IITM in $\mathcal{E}|\mathcal{A}|\mathcal{P}$ and $\mathcal{E}|\mathcal{I}|\mathcal{F}$, i.e. every run directly terminates with empty output, which trivially fulfills the following statements. Since $\mathcal{A}|\mathcal{P}$ is R-environmentally bounded we have that $\mathcal{E}|\mathcal{A}|\mathcal{P}$ is almost bounded. Furthermore, by the definition of responsive adversaries, we know that restricting messages of $\mathcal{P}$ are answered correctly in an overwhelming set of runs of $\mathcal{E}|\mathcal{A}|\mathcal{P}$. This gives the following equation:

$$\begin{aligned}
\mathcal{E}|\mathcal{A}|\mathcal{P} &\equiv [\mathcal{E}|\mathcal{A}]_{\mathcal{P}}|\mathcal{P} && \text{(Lemma C.16)}\\
&\equiv [\mathcal{E}|\mathcal{A}]_{\mathcal{P}}|\mathcal{S}|\mathcal{F} && \text{(Lemma C.18 and } \mathcal{P} \leq_R^{SS} \mathcal{F}\text{)}\\
&\equiv \mathcal{E}|\mathcal{A}|\mathcal{S}|\mathcal{F} && \text{(Lemma C.19)}\\
&= \mathcal{E}|\mathcal{I}|\mathcal{F}
\end{aligned}$$

We also obtain that $\mathcal{E}|\mathcal{I}|\mathcal{F}$ is almost bounded by Lemma C.19. Since this holds for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{I}|\mathcal{F})$ by Lemma C.11, we have that $\mathcal{I}|\mathcal{F}$ is R-environmentally bounded.

Now observe that there is only a negligible set of runs of $[\mathcal{E}|\mathcal{A}]_{\mathcal{P}}|\mathcal{S}|\mathcal{F}$ where a restricting message from $\mathcal{F}$ is answered incorrectly because $\mathcal{S}$ is a responsive simulator. Since $[\mathcal{E}|\mathcal{A}]_{\mathcal{P}}|\mathcal{S}|\mathcal{F}$ and $\mathcal{E}|\mathcal{A}|\mathcal{S}|\mathcal{F}$ are the same except for a negligible set of runs (see the proof of Lemma C.19), the set of runs of $\mathcal{E}|\mathcal{I}|\mathcal{F}$ where a restricting message from $\mathcal{F}$ is answered incorrectly must also be negligible. Overall we have $\mathcal{I} \in \mathrm{Sim}_R^{\mathcal{A}|\mathcal{P}}(\mathcal{F})$, which shows that $\mathcal{P} \leq_R^{UC} \mathcal{F}$.

For the other implication, let $\mathcal{P} \leq_R^{dumUC} \mathcal{F}$. Then there exists a responsive simulator $\mathcal{I} \in \mathrm{Sim}_R^{D_{\mathcal{P}}|\mathcal{P}}(\mathcal{F})$ such that $\mathcal{E}|D_{\mathcal{P}}|\mathcal{P} \equiv \mathcal{E}|\mathcal{I}|\mathcal{F}$ for all $\mathcal{E} \in \mathsf{Env}_R(D_{\mathcal{P}}|\mathcal{P})$. Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$. We can construct a new responsive environment $\mathcal{E}' \in \mathsf{Env}_R(D_{\mathcal{P}}|\mathcal{P})$ that works exactly as $\mathcal{E}$ but for all network tapes c that connect between $\mathcal{E}$ and $\mathcal{P}$, $\mathcal{E}'$ instead has a network tape c' that connects to the tape of $D_{\mathcal{P}}$ that corresponds to c. Note that $\mathcal{E}|\mathcal{P}$ and $\mathcal{E}'|D_{\mathcal{P}}|\mathcal{P}$ work in exactly the same way since we can rename tapes and insert the dummy adversary without changing the behavior of the system by Lemma 4 and 5 from [22]. This implies $\mathcal{E}' \in \mathsf{Env}_R(D_{\mathcal{P}}|\mathcal{P})$ since restricting messages from $\mathcal{P}$ are answered correctly in an overwhelming set of runs of $\mathcal{E}|\mathcal{P}$ and thus restricting messages of $D_{\mathcal{P}}|\mathcal{P}$ must also be answered correctly in an overwhelming set of runs of $\mathcal{E}'|D_{\mathcal{P}}|\mathcal{P}$.

Similarly, we can construct $\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})$ that works just as $\mathcal{I}$ but for every external network tape c' that does not connect to $\mathcal{F}$, $\mathcal{S}$ instead has a network tape c that corresponds to c'.[18] Now we have:

$$\begin{aligned}
\mathcal{E}|\mathcal{P} &\equiv \mathcal{E}'|D_{\mathcal{P}}|\mathcal{P} && \text{(Lemma 4 and 5 from [22])}\\
&\equiv \mathcal{E}'|\mathcal{I}|\mathcal{F} && (\mathcal{P} \leq_R^{dumUC} \mathcal{F})\\
&\equiv \mathcal{E}|\mathcal{S}|\mathcal{F} && \text{(Lemma 4 from [22])}
\end{aligned}$$

It also holds true that $\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{F}}(\mathcal{P})$: The conditions regarding external interfaces are obviously fulfilled. Since $\mathcal{E}'|\mathcal{I}|\mathcal{F}$ and $\mathcal{E}|\mathcal{S}|\mathcal{F}$ are the same and $\mathcal{E}'|\mathcal{I}|\mathcal{F}$ is almost bounded, we have that $\mathcal{E}|\mathcal{S}|\mathcal{F}$ is almost bounded. This holds for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{S}|\mathcal{F})$ by Lemma C.11, and thus, $\mathcal{S}|\mathcal{F}$ is R-environmentally bounded. Finally, observe that restricting messages from $\mathcal{F}$ are answered correctly in an overwhelming set of runs of $\mathcal{E}'|\mathcal{I}|\mathcal{F}$ because $\mathcal{I}$

[18] With *corresponds* we mean that $\mathcal{P}$ has a network tape c, which is mapped to a network tape c' by the dummy adversary.

is a responsive simulator. This implies that restricting messages from $\mathcal{F}$ are also answered correctly in an overwhelming set of runs of $\mathcal{E}|\mathcal{S}|\mathcal{F}$, which shows $\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{F}}(\mathcal{P})$, and thus, $\mathcal{P} \leq_R^{SS} \mathcal{F}$.

$\mathcal{P} \leq_R^{SS} \mathcal{F}$ *iff* $\mathcal{P} \leq_R^{BB} \mathcal{F}$: We first show $\mathrm{SS}_R \Rightarrow \mathrm{BB}_R$. Let $\mathcal{P} \leq_R^{SS} \mathcal{F}$. Then there exists $\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})$ with $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{S}|\mathcal{F}$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$. Let $\mathcal{A} \in \mathrm{Adv}_R(\mathcal{P})$ and $\mathcal{E} \in \mathsf{Env}_R(\mathcal{A}|\mathcal{P})$. Analogously to the proof of "$\mathrm{SS}_R \Rightarrow \mathrm{UC}_R$" we obtain that $\mathcal{E}|\mathcal{A}|\mathcal{P} \equiv \mathcal{E}|\mathcal{A}|\mathcal{S}|\mathcal{F}$,which directly implies $\mathcal{P} \leq_R^{BB} \mathcal{F}$ since $\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})$.

For the other implication let $\mathcal{P} \leq_R^{BB} \mathcal{F}$. Then there exists $\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})$ with $\mathcal{E}|\mathcal{A}|\mathcal{P} \equiv \mathcal{E}|\mathcal{A}|\mathcal{S}|\mathcal{F}$ for all $\mathcal{A} \in \mathrm{Adv}_R(\mathcal{P})$ and $\mathcal{E} \in \mathsf{Env}_R(\mathcal{A}|\mathcal{P})$. In the following let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$. Analogous to the proof of "$\mathrm{dummyUC}_R \Rightarrow \mathrm{SS}_R$", we can construct a new environment $\mathcal{E}' \in \mathsf{Env}_R(D_{\mathcal{P}}|\mathcal{P})$ by renaming some tapes of $\mathcal{E}$ such that $\mathcal{E}'$ connects to the dummy adversary $D_{\mathcal{P}}$ instead of the network tapes of $\mathcal{P}$. We obtain the following equation:

$$\begin{aligned}
\mathcal{E}|\mathcal{P} &\equiv \mathcal{E}'|D_{\mathcal{P}}|\mathcal{P} && \text{(Lemma 4 and 5 from [22])}\\
&\equiv \mathcal{E}'|D_{\mathcal{P}}|\mathcal{S}|\mathcal{F} && (D_{\mathcal{P}} \in \mathrm{Adv}_R(\mathcal{P}) \text{ and } \mathcal{P} \leq_R^{BB} \mathcal{F})\\
&\equiv \mathcal{E}|\mathcal{S}|\mathcal{F} && \text{(Lemma 4 and 5 from [22])}
\end{aligned}$$

Since $\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})$ we directly obtain $\mathcal{P} \leq_R^{SS} \mathcal{F}$.

$\mathcal{P} \leq_R^{SS} \mathcal{F}$ *iff* $\mathcal{P} \leq_R^{RS} \mathcal{F}$: Because $\mathrm{dummyUC}_R$, UC_R and SS_R are equivalent and UC_R trivially implies RS_R, we have to show only that RS_R implies $\mathrm{dummyUC}_R$. The main argument is similar to the one presented in [8].

Let $\mathcal{P} \leq_R^{RS} \mathcal{F}$, then for all $\mathcal{A} \in \mathrm{Adv}_R(\mathcal{P})$ and all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{A}|\mathcal{P})$ there exists $\mathcal{I} \in \mathrm{Sim}_R^{\mathcal{A}|\mathcal{P}}(\mathcal{F})$ such that:

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

We choose $\mathcal{A} = D_{\mathcal{P}}$ to be the dummy adversary for $\mathcal{P}$. We also choose $\mathcal{E} \in \mathsf{Env}_R(D_{\mathcal{P}}|\mathcal{P})$ 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 to be external input to $\mathcal{E}'$, and t is interpreted to be runtime (by Lemma 6 from [22], 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. During this simulation, $\mathcal{E}$ always checks whether $\mathcal{E}'$ is about to send an incorrect answer to a restricting message of $D_{\mathcal{P}}|\mathcal{P}$ and, if that is the case, stops the run with empty output instead. This check is possible in polynomial time, since R is decidable in polynomial time in the second component (also see the proof of Lemma C.11 for a more detailed discussion). Clearly, $\mathcal{E}$ is universally bounded because its runtime is polynomial in the security parameter plus the length of the external input. Furthermore, $\mathcal{E}$ is responsive (for $D_{\mathcal{P}}|\mathcal{P}$) since it never answers a restricting message incorrectly. This implies that there exists $\mathcal{I}_{\mathcal{E}} \in \mathrm{Sim}_R^{D_{\mathcal{P}}|\mathcal{P}}(\mathcal{F})$ such that $\mathcal{E}|D_{\mathcal{P}}|\mathcal{P} \equiv \mathcal{E}|\mathcal{I}_{\mathcal{E}}|\mathcal{F}$.

We observe that 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 $\mathcal{E}'$ except for runs where a restricting message is answered incorrectly by $\mathcal{E}'$. We will refer to this observation by (*).

Now let $\mathcal{E}' \in \mathsf{Env}_R(D_{\mathcal{P}}|\mathcal{P})$ be a responsive environment with representation e. We first show that $\mathcal{E}' \in \mathsf{Env}_R(\mathcal{I}_{\mathcal{E}}|\mathcal{F})$. For this, let $\mathcal{E}'' \in \mathsf{Env}_R(D_{\mathcal{P}}|\mathcal{P})$ be the universally bounded IITM that simulates $\mathcal{E}'$, checks whether a restricting message is about to be answered incorrectly and outputs 1 if this is about to happen. In all other cases 0 is output (again, see the proof of Lemma C.11 which includes a more detailed construction of such an IITM). Let e' be the representation of $\mathcal{E}''$. Since $\mathcal{E}''$ never answers a responsive message incorrectly, we can use (*) to obtain that for all $\eta \in \mathbb{N}$ and $a \in \{0,1\}^*$ there exists a tuple $(a, e', 1^t)$ such that $\mathcal{E}''$ is perfectly simulated by $\mathcal{E}$ with external input $(a, e', 1^t)$. Since $\mathcal{E}|D_{\mathcal{P}}|\mathcal{P} \equiv \mathcal{E}|\mathcal{I}_{\mathcal{E}}|\mathcal{F}$, i.e. both systems differ only with negligible probability, the length of $(a, e', 1^t)$ is polynomial in η and $|a|$, and $\mathcal{E}''$ outputs 1 with negligible probability in runs of $\mathcal{E}''|D_{\mathcal{P}}|\mathcal{P}$, we obtain that $\mathcal{E}''$ also outputs 1 with negligible probability in runs of $\mathcal{E}''|\mathcal{I}_{\mathcal{E}}|\mathcal{F}$. This implies that $\mathcal{E}' \in \mathsf{Env}_R(\mathcal{I}_{\mathcal{E}}|\mathcal{F})$.

In the following, let g be the negligible function that bounds the probability of an incorrectly answered message of $D_{\mathcal{P}}|\mathcal{P}$ in runs of $\mathcal{E}'|D_{\mathcal{P}}|\mathcal{P}$ and let g' be the negligible function that bounds the probability of an incorrectly answered message of $\mathcal{I}_{\mathcal{E}}|\mathcal{F}$ in runs of $\mathcal{E}'|\mathcal{I}_{\mathcal{E}}|\mathcal{F}$. For all $\eta \in \mathbb{N}$ and $a \in \{0,1\}^*$, let $(a, e, 1^t)$ be the tuple from (*) such that $\mathcal{E}$ simulates $\mathcal{E}'$ perfectly in runs where all restricting messages are answered correctly. We obtain the following equation, where f is a negligible function:

$$\begin{aligned}
&|\Pr[\mathcal{E}'|D_{\mathcal{P}}|\mathcal{P}(1^\eta, a) = 1] - \Pr[\mathcal{E}'|I_{\mathcal{E}}|\mathcal{F}(1^\eta, a) = 1]| \\
\leq\ &|\Pr\left[\mathcal{E}|D_{\mathcal{P}}|\mathcal{P}(1^\eta, (a, e, 1^t)) = 1\right] - \Pr\left[\mathcal{E}|I_{\mathcal{E}}|\mathcal{F}(1^\eta, (a, e, 1^t)) = 1\right]| + g(\eta, a) + g'(\eta, a) \\
\leq\ &f(\eta, (a, e, 1^t)) + g(\eta, a) + g'(\eta, a)
\end{aligned}$$

Since the length of $(a, e, 1^t)$ is polynomial in $\eta + |a|$, with e being a fixed bit string and t being a polynomial in η and a, we have that $f'(\eta, a) := f(\eta, (a, e, 1^t))$ is a negligible function. Now, since the sum of three negligible functions is still negligible, we have that there is a negligible function (in η and a) that bounds $|\Pr[\mathcal{E}'|D_{\mathcal{P}}|\mathcal{P}(1^\eta, a) = 1] - \Pr[\mathcal{E}'|I_{\mathcal{E}}|\mathcal{F}(1^\eta, a) = 1]|$. Hence, we have that $\mathcal{E}'|D_{\mathcal{P}}|\mathcal{P} \equiv \mathcal{E}'|\mathcal{I}_{\mathcal{E}}|\mathcal{F}$ for all $\mathcal{E}' \in \mathsf{Env}_R(D_{\mathcal{P}}|\mathcal{P})$, which together with $\mathcal{I}_{\mathcal{E}} \in \mathrm{Sim}_R^{D_{\mathcal{P}}|\mathcal{P}}(\mathcal{F})$ implies $\mathcal{P} \leq_R^{dumUC} \mathcal{F}$.

□

C.6 Composition of a Constant Number of Protocol Systems

In this section we formally define and prove Theorem 4.8 in the IITM model. Analogously to [22], in order to prove a theorem for composing a constant number of protocol systems, we first prove a theorem which talks about computational indistinguishability. This theorem corresponds to Theorem 4 in [22], which we adjust to the case of responsive environments.

Theorem C.25. *Let R be a restriction. Let $k \geq 1$, $\mathcal{Q}, \mathcal{P}_1, \ldots, \mathcal{P}_k, \mathcal{R}_1, \ldots, \mathcal{R}_k$ be systems of IITMs without* start *and* decision *tapes such that the following conditions are satisfied:*

1. *For all $j \leq k$: $\mathcal{P}_j$ and $\mathcal{R}_j$ are R-environmentally bounded, $\mathcal{P}_j$ and $\mathcal{R}_j$ have the same (external) interface, and $\mathcal{E} \,|\, \mathcal{P}_j \equiv \mathcal{E} \,|\, \mathcal{R}_j$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P}_j)$.*
2. *$\mathcal{Q}, \mathcal{P}_1, \ldots, \mathcal{P}_k$ are I/O-connectable (hence, $\mathcal{Q}, \mathcal{R}_1, \ldots, \mathcal{R}_k$ are I/O-connectable) and $\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k$ is R-environmentally bounded.*

Then, $\mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k \equiv \mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{R}_1 \,|\, \ldots \,|\, \mathcal{R}_k$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k)$ and $\mathcal{Q} \,|\, \mathcal{R}_1 \,|\, \ldots \,|\, \mathcal{R}_k$ is R-environmentally bounded.

Proof. The proof follows essentially the same argument as the original proof in [22], but uses responsive environments instead of general environments.

Let $\mathcal{Q}, \mathcal{P}_1, \ldots, \mathcal{P}_k, \mathcal{R}_1, \ldots, \mathcal{R}_k$ be systems of IITMs such that the requirements of the theorem are fulfilled. We will first prove the theorem for $k = 1$:

Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q} \,|\, \mathcal{P}_1)$ be a responsive environment. We can assume that start is a tape of $\mathcal{E}$; otherwise there is no master IITM in $\mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{P}$, thus every run terminates directly with empty output, in which case the theorem holds true. If $\mathcal{P}_1$ sends a restricting message m on an external network tape t, then t is also an external network tape of $\mathcal{Q} \,|\, \mathcal{P}_1$ since $\mathcal{Q}$ and $\mathcal{P}_1$ only connect via I/O tapes. Because $\mathcal{E}$ is a responsive environment for $\mathcal{Q} \,|\, \mathcal{P}_1$ we have that the next message m' that is sent to $\mathcal{Q} \,|\, \mathcal{P}_1$ will be on tape t^{-1} with $(m, m') \in R$ (except for a negligible set of runs). Using Lemma C.16 and Lemma C.18, we obtain $\mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{P}_1 \equiv [\mathcal{E} \,|\, \mathcal{Q}]_{\mathcal{P}_1} \,|\, \mathcal{P}_1$ with $[\mathcal{E} \,|\, \mathcal{Q}]_{\mathcal{P}_1} \in \mathsf{Env}_R(\mathcal{P}_1)$. Then $\mathcal{E} \,|\, \mathcal{P}_1 \equiv \mathcal{E} \,|\, \mathcal{R}_1$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P}_1)$ implies $[\mathcal{E} \,|\, \mathcal{Q}]_{\mathcal{P}_1} \,|\, \mathcal{P}_1 \equiv [\mathcal{E} \,|\, \mathcal{Q}]_{\mathcal{P}_1} \,|\, \mathcal{R}_1$. Using Lemma C.19, we obtain $[\mathcal{E} \,|\, \mathcal{Q}]_{\mathcal{P}_1} \,|\, \mathcal{R}_1 \equiv \mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{R}_1$. This implies that $\mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{P}_1 \equiv \mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{R}_1$.

We still have to prove that $\mathcal{Q} \,|\, \mathcal{R}_1$ is R-environmentally bounded: By Lemma C.11 we know that $[\mathcal{E} \,|\, \mathcal{Q}]_{\mathcal{P}_1} \in \mathsf{Env}_R(\mathcal{R}_1)$, and thus, since $\mathcal{R}_1$ is R-environmentally bounded, we know that $[\mathcal{E} \,|\, \mathcal{Q}]_{\mathcal{P}_1} \,|\, \mathcal{R}_1$ is almost bounded. By Lemma C.19 we conclude that $\mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{R}_1$ is almost bounded. Since this holds true for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q} \,|\, \mathcal{P}_1)$ and thus, by Lemma C.11, for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q} \,|\, \mathcal{R}_1)$, we have that $\mathcal{Q} \,|\, \mathcal{R}_1$ is R-environmentally bounded.

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

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

which can be connected to $\mathcal{P}_r$ or $\mathcal{R}_r$. Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k)$ be a responsive environment. By applying the case "$k = 1$" k times, we obtain that:

$$\begin{aligned}
&\mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \cdots \,|\, \mathcal{P}_k \\
&= \mathcal{E} \,|\, \mathcal{H}_1 \,|\, \mathcal{P}_1 && \text{(syntactic reordering of systems)} \\
&\equiv \mathcal{E} \,|\, \mathcal{H}_1 \,|\, \mathcal{R}_1 && \text{(case } k = 1 \text{ with } \mathcal{H}_1 \text{ playing the role of } \mathcal{Q}) \\
&= \mathcal{E} \,|\, \mathcal{H}_2 \,|\, \mathcal{P}_2 && \text{(syntactic reordering of systems)} \\
&\equiv \mathcal{E} \,|\, \mathcal{H}_2 \,|\, \mathcal{R}_2 && \text{(Lemma C.11 and case } k = 1 \text{ with } \mathcal{H}_2 \text{ playing the role of } \mathcal{Q}) \\
&\;\;\vdots \\
&= \mathcal{E} \,|\, \mathcal{H}_k \,|\, \mathcal{P}_k && \text{(syntactic reordering of systems)} \\
&\equiv \mathcal{E} \,|\, \mathcal{H}_k \,|\, \mathcal{R}_k && \text{(Lemma C.11 and case } k = 1 \text{ with } \mathcal{H}_k \text{ playing the role of } \mathcal{Q}) \\
&= \mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{R}_1 \,|\, \ldots \,|\, \mathcal{R}_k
\end{aligned}$$

In the above argument we need that $\mathcal{H}_r \,|\, \mathcal{P}_r$ is R-environmentally bounded. This is the case: $\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \cdots \,|\, \mathcal{P}_k = \mathcal{H}_1 \,|\, \mathcal{P}_1$ is R-environmentally bounded by assumption. But then $\mathcal{H}_1 \,|\, \mathcal{R}_1$ is also R-environmentally because the theorem holds true for "$k = 1$". Iterating this argument proves the claim; in particular this shows that $\mathcal{Q} \,|\, \mathcal{R}_1 \,|\, \ldots \,|\, \mathcal{R}_k$ is R-environmentally bounded, which is the second part of the theorem. □

Using the above theorem, we can now prove the composition theorem for a constant number of protocol systems.

Theorem C.26 (Composition Theorem for a Constant Number of Protocol Systems). *Let R be a restriction. Let $k \geq 1$, $\mathcal{Q}$ be a system of IITMs without* start *and* decision *tape, and $\mathcal{P}_1, \ldots, \mathcal{P}_k, \mathcal{F}_1, \ldots, \mathcal{F}_k$ be protocol systems such that all systems have pairwise disjoint sets of network tapes and the following conditions are satisfied:*

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

Then, $\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k \leq_R \mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{F}_k$

Proof. Let $\mathcal{Q}, \mathcal{P}_1, \ldots, \mathcal{P}_k, \mathcal{F}_1, \ldots, \mathcal{F}_k$ be as assumed in the theorem. For all $j \leq k$ let $\mathcal{S}_j \in \mathrm{Sim}_R^{\mathcal{P}_j}(\mathcal{F}_j)$ be a responsive simulator such that $\mathcal{E} \,|\, \mathcal{P}_j \equiv \mathcal{E} \,|\, \mathcal{S}_j \,|\, \mathcal{F}_j$ for all responsive environments $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P}_j)$. Note that, since $\mathcal{Q}, \mathcal{P}_1, \cdots, \mathcal{P}_k, \mathcal{F}_1, \cdots, \mathcal{F}_k$ have disjoint sets of network tapes, the systems $\mathcal{Q}, \mathcal{S}_1, \mathcal{F}_1, \cdots, \mathcal{S}_k, \mathcal{F}_k$ are connectable.

By applying Theorem C.25 (with $\mathcal{R}_j := \mathcal{S}_j \,|\, \mathcal{F}_j$ for $j \leq k$) we directly obtain that $\mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k \equiv \mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{S}_1 \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{S}_k \,|\, \mathcal{F}_k$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k)$. We also obtain that $\mathcal{Q} \,|\, \mathcal{S}_1 \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{S}_k \,|\, \mathcal{F}_k$ is R-environmentally bounded.

We still have to show that $\mathcal{S}_1 \,|\, \ldots \,|\, \mathcal{S}_k \in \mathrm{Sim}_R^{\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k}(\mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{F}_k)$. It is easy to see that all conditions regarding interfaces of responsive simulators are fulfilled. We have also just proven the runtime condition for $\mathcal{S}_1 \,|\, \ldots \,|\, \mathcal{S}_k$. So we only have to show that restricting messages from $\mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{F}_k$ are answered correctly in an overwhelming set of runs. Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q} \,|\, \mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_k)$ be a responsive environment. By Lemma C.11 we know that $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q} \,|\, \mathcal{S}_1 \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{S}_k \,|\, \mathcal{F}_k)$. We may assume that start is a tape of $\mathcal{E}$; otherwise there is no master IITM in $\mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{S}_1 \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{S}_k \,|\, \mathcal{F}_k$ and every run directly terminates with empty output. Since there are no messages sent in such a run, there also is no restricting message that is answered incorrectly and the claim follows.

Let the event E be the set of all runs of $\mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{S}_1 \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{S}_k \,|\, \mathcal{F}_k$ where a restricting message from $\mathcal{Q} \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{F}_k$ is answered incorrectly, let E' be the set of all runs of $\mathcal{E} \,|\, \mathcal{Q} \,|\, \mathcal{S}_1 \,|\, \mathcal{F}_1 \,|\, \ldots \,|\, \mathcal{S}_k \,|\, \mathcal{F}_k$ where

a restricting message from $\mathcal{Q}$ is answered incorrectly, and let E_j for $j \leq k$ be the set of all runs of $\mathcal{E} \mid \mathcal{Q} \mid \mathcal{S}_1 \mid \mathcal{F}_1 \mid \ldots \mid \mathcal{S}_k \mid \mathcal{F}_k$ where a restricting message from $\mathcal{F}_j$ is answered incorrectly. Note that $\mathcal{Q}, \mathcal{F}_1, \ldots, \mathcal{F}_k$ have disjoint network tapes, so if one of those systems sends a restricting message on a network tape, then this message is sent on an external network tape of $\mathcal{Q} \mid \mathcal{F}_1 \mid \ldots \mid \mathcal{F}_k$. In particular, we have that $E = E' \cup \bigcup_{j \leq k} E_j$, and thus, $\Pr[E] \leq \Pr[E'] + \sum_{j \leq k} \Pr[E_j]$.

Since every incorrectly answered restricting message from $\mathcal{Q}$ is sent on an external network tape of $\mathcal{Q} \mid \mathcal{S}_1 \mid \mathcal{F}_1 \mid \ldots \mid \mathcal{S}_k \mid \mathcal{F}_k$ (network tapes of $\mathcal{Q}$, $\mathcal{P}_j$ and $\mathcal{F}_j$ are disjoint for all $j \leq k$) and since $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q} \mid \mathcal{S}_1 \mid \mathcal{F}_1 \mid \ldots \mid \mathcal{S}_k \mid \mathcal{F}_k)$ we directly obtain that E' is negligible.

Now, let us consider the event E_j for some $j \leq k$. For brevity of presentation, we define the system $\mathcal{H} := \mathcal{E} \mid \mathcal{Q} \mid \mathcal{S}_1 \mid \mathcal{F}_1 \mid \ldots \mid \mathcal{S}_{j-1} \mid \mathcal{F}_{j-1} \mid \mathcal{S}_{j+1} \mid \mathcal{F}_{j+1} \mid \ldots \mid \mathcal{S}_k \mid \mathcal{F}_k$, i.e. the system $\mathcal{H}$ does not contain $\mathcal{S}_j$ and $\mathcal{F}_j$. We will construct a responsive environment for $\mathcal{S}_j \mid \mathcal{F}_j$: First we observe that if the system $\mathcal{S}_j \mid \mathcal{F}_j$ sends a restricting message on an external network tape (in a run of $\mathcal{E} \mid \mathcal{Q} \mid \mathcal{S}_1 \mid \mathcal{F}_1 \mid \ldots \mid \mathcal{S}_k \mid \mathcal{F}_k$), the restricting message is sent on an external network tape of $\mathcal{Q} \mid \mathcal{S}_1 \mid \mathcal{F}_1 \mid \ldots \mid \mathcal{S}_k \mid \mathcal{F}_k$ since the systems $\mathcal{Q}, \mathcal{S}_1 \mid \mathcal{F}_1, \ldots, \mathcal{S}_k \mid \mathcal{F}_k$ have disjoint network tapes. Because of $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q} \mid \mathcal{S}_1 \mid \mathcal{F}_1 \mid \ldots \mid \mathcal{S}_k \mid \mathcal{F}_k)$ those messages are answered correctly in an overwhelming set of runs; we will refer to this observation by (*) in what follows. Since the system $\mathcal{E} \mid \mathcal{Q} \mid \mathcal{S}_1 \mid \mathcal{F}_1 \mid \ldots \mid \mathcal{S}_k \mid \mathcal{F}_k$ is almost bounded and $\mathcal{E}$ contains a master IITM, we can use Lemma C.16, and because of (*), we can also use Lemma C.18 to obtain $\mathcal{E} \mid \mathcal{Q} \mid \mathcal{S}_1 \mid \mathcal{F}_1 \mid \ldots \mid \mathcal{S}_k \mid \mathcal{F}_k = \mathcal{H} \mid \mathcal{S}_j \mid \mathcal{F}_j \equiv [\mathcal{H}]_{\mathcal{S}_j \mid \mathcal{F}_j} \mid \mathcal{S}_j \mid \mathcal{F}_j$ with $[\mathcal{H}]_{\mathcal{S}_j \mid \mathcal{F}_j} \in \mathsf{Env}_R(\mathcal{S}_i \mid \mathcal{F}_i)$. Furthermore, by Lemma C.11, we also have that $[\mathcal{H}]_{\mathcal{S}_j \mid \mathcal{F}_j} \in \mathsf{Env}_R(\mathcal{P}_j)$.

Because the system $\mathcal{S}_j$ is a responsive simulator, we know that restricting messages from $\mathcal{F}_j$ must be answered correctly in an overwhelming set of runs of $[\mathcal{H}]_{\mathcal{S}_j \mid \mathcal{F}_j} \mid \mathcal{S}_j \mid \mathcal{F}_j$. But then restricting messages from $\mathcal{F}$ in $\mathcal{H} \mid \mathcal{S}_j \mid \mathcal{F}_j$ must also be answered correctly in an overwhelming set of runs since $\mathcal{H} \mid \mathcal{S}_j \mid \mathcal{F}_j$ and $[\mathcal{H}]_{\mathcal{S}_j \mid \mathcal{F}_j} \mid \mathcal{S}_j \mid \mathcal{F}_j$ are identical in an overwhelming set of runs. This implies that the event E_j is negligible.

Overall, we have that $\Pr[E] \leq \Pr[E'] + \sum_{j \leq k} \Pr[E_j]$ where the right side is a sum of $k+1$ negligible functions, and hence, $\Pr[E]$ is negligible. □

C.7 Unbounded Self-Composition of SID Dependent Protocols

We now prove Theorem 4.9 in the IITM model. Recall that this composition theorem says that it suffices to prove that a single session of a real protocol $\mathcal{P}$ realizes a single session of an ideal protocol $\mathcal{F}$ in order to conclude that multiple sessions of the real protocol realize multiple sessions of the ideal protocol.

For this purpose, we first recall and adapt the definitions of *session identifier (SID) functions* and *σ-session versions* from [22]. An SID function assigns an SID (or $\bot$) to every message sent or received on a tape of an IITM. An IITM M is called a σ-session version if it does not accept messages (in mode **CheckAddress**) for which σ outputs $\bot$. Also, if M accepted a message on some tape at some point for which σ returned the SID $sid \neq \bot$, then later on M may only accept messages with the same SID sid, i.e., for which σ returns sid. Also, M may only output messages with this sid. A system is called a σ-session version if all IITMs in that system are σ-session versions.

Then, we recall and adapt the definitions of *σ-single session environments and simulators* from [22]. Such environments may output messages which all have the same SID (according to σ), and hence, these environments can create only a single protocol session in every run. The (restricted) simulators have to work only for those environments.

Finally, we state a version of the composition theorem for unbounded self-composition from [22] (see Theorem 10 in that work). We adapt this theorem for responsive environments.

As mentioned, we start with the definition of SID functions and σ-session versions.

Definition C.27 (Session Identifier (SID) Function). *A function $\sigma : \{0,1\}^* \times \mathcal{T} \to \{0,1\}^* \cup \{\bot\}$ (where $\mathcal{T}$ is a set of tapes names) is called a* session identifier (SID) function *if it is computable in polynomial time (in the length of its input).*

Definition C.28 (σ-Session Versions). *Let σ be an SID function and let M be an IITM such that σ is defined for all (names of) tapes of M. Then, M is a* σ-session machine *(also called a* σ-session version*) if for*

every system $\mathcal{Q}$ such that $\mathcal{Q}$ and M are connectable the following conditions are satisfied for every η, a, and every run ρ of $(\mathcal{Q}|M)(1^\eta, a)$:

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 non-empty 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 $\mathcal{R}$ is called a σ-session system/version *if every IITM occuring in $\mathcal{R}$ is a σ-session version.*

We will also use a stronger variant of σ-session versions. An IITM M is called *σ-complete*, if it fulfills the conditions stated in Definition C.28, 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**.

Definition C.29 (σ-Single Session Responsive Environment). *Let R be a restriction. Let σ be an SID function, let $\mathcal{P}$ be a system of IITMs and let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ be a responsive environment for $\mathcal{P}$. The system $\mathcal{E}$ is called a* σ-single session responsive environment *if for every system $\mathcal{Q}$ such that $\mathcal{E}$ and $\mathcal{Q}$ are connectable the following holds true for every η, a and in every run ρ of $(\mathcal{E} \mid \mathcal{Q})(1^\eta, a)$:*

Let $m_0 \neq \epsilon$ (where ϵ is the empty bit string) be the first message output by $\mathcal{E}$ on some external tape $c_0 \neq \mathsf{decision}$ in ρ. Then $\sigma(m_0, c_0) \neq \bot$ and every message $m \neq \epsilon$ output by $\mathcal{E}$ on an external tape $c \neq \mathsf{decision}$ in ρ satisfies $\sigma(m, c) = \sigma(m_0, c_0)$.

We denote the set of all σ-single session responsive environments for a system $\mathcal{P}$ by $\mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$.

Definition C.30 (σ-Single Session Responsive Simulator). *Let R be a restriction. Let σ be an SID function and let $\mathcal{P}, \mathcal{F}$ be protocol systems. Let $\mathcal{S}$ be an adversarial system 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 $\mathcal{F}$, $\mathcal{S} \mid \mathcal{F}$ and $\mathcal{P}$ have the same external interface, and $\mathcal{E} \mid \mathcal{S} \mid \mathcal{F}$ is almost bounded for all $\mathcal{E}$ in $\mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$.*

Let $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$ be a σ-single session responsive environment. Define the event E to be the set of all runs of $\mathcal{E} \mid \mathcal{S} \mid \mathcal{F}$ where the following holds true: If $\mathcal{F}$ sends a restricting message $m \in R[0]$ on an external tape $t \in \mathsf{NET}$ and if there exists a message m' such that m' is the first message received on an external tape t' of $\mathcal{F}$ after sending m, then $t' = t^{-1}$ and $(m, m') \in R$.

The simulator $\mathcal{S}$ is called a σ-single session responsive simulator *(with respect to σ and R) if and only if $\Pr[E]$ is overwhelming for all σ-single session responsive environments $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$. We denote the set of all σ-single session responsive simulators for protocol systems $\mathcal{P}$ and $\mathcal{F}$ by $\mathrm{Sim}^{\mathcal{P}}_{R,\sigma\text{-single}}(\mathcal{F})$.*

Note that every responsive simulator also is a σ-single session responsive simulator, since the definition of the latter is a relaxed version of the former (i.e. the runtime condition and the event E are only defined for a subset of all responsive environments).

Definition C.31 (σ-Single Session Realization). *Let R be a restriction. Let σ be an SID function and let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems, the* real *and* ideal *protocol, respectively, such that $\mathcal{P}$ and $\mathcal{F}$ are σ-session versions. Then, $\mathcal{P}$* single-session realizes *$\mathcal{F}$ w.r.t. σ and R ($\mathcal{P} \leq_{R,\sigma\text{-single}} \mathcal{F}$) if and only if there exists $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{R,\sigma\text{-single}}(\mathcal{F})$ such that $\mathcal{E} \mid \mathcal{P} \equiv \mathcal{E} \mid \mathcal{S} \mid \mathcal{F}$ for every σ-single session responsive environment $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$.*

Now, the following theorem says that if a single session of $\mathcal{P}$ realizes a single session of $\mathcal{F}$, i.e., if $\mathcal{P}$ realizes $\mathcal{F}$ for all σ-single session (responsive) environments, then multiple session of $\mathcal{P}$ realize multiple sessions of $\mathcal{F}$, i.e., $\mathcal{P}$ realizes $\mathcal{F}$ for all (responsive) environments.

Theorem C.32 (Composition Theorem for Unbounded Self-Composition of SID Dependent Protocols). *Let R be a restriction. 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 and $\mathcal{P} \leq_{R,\sigma\text{-single}} \mathcal{F}$. Then, $\mathcal{P} \leq_R \mathcal{F}$.*

Before we can prove this theorem we have to show some additional lemmas and another theorem.

In what follows, we will often use the (informal) term *copy of a system* to denote the set of all instances of a σ-session version (in some run) that accept messages with the same SID. Note that this term is justified, since σ-session versions will never accept two messages with different SIDs and thus we can group instances according to the SID of the first message they accepted.

Lemma C.33 (Variant of Lemma C.11 for σ-single session environments). *Let R be a restriction. Let $\mathcal{P}$ and $\mathcal{P}'$ be two systems of IITMs such that neither of them has a* start *or* decision *tape, both systems have the same external interface, and $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{P}'$ for all $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$. Then, $\mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P}) = \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P}')$.*

Proof. The original proof can easily be adjusted: First, one assumes $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$. Then it is obvious that $\mathcal{E}' \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$ by construction. The remainder of the proof stays exactly the same. □

Lemma C.34 (Variant of Lemma C.19 for σ-single session environments). *Let R be a restriction. Let $\mathcal{R}$ and $\mathcal{Q}$ be connectable systems such that $\mathcal{R}|\mathcal{Q}$ is almost bounded,* start *is a tape of $\mathcal{R}$ (i.e. $\mathcal{R}$ contains a master IITM), and* decision *is not a tape of $\mathcal{Q}$. Furthermore, let $\mathcal{Q}'$ be a system which is compatible with $\mathcal{Q}$ and satisfies the following condition: $\mathcal{E}|\mathcal{Q} \equiv \mathcal{E}|\mathcal{Q}'$ for every $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{Q})$ such that $\mathcal{E}|\mathcal{Q}$ is almost bounded. If $[\mathcal{R}]_{\mathcal{Q}} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{Q})$, then*

$$[\mathcal{R}]_{\mathcal{Q}}|\mathcal{Q}' \equiv \mathcal{R}|\mathcal{Q}' \ .$$

Moreover, if $[\mathcal{R}]_{\mathcal{Q}} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{Q})$ and $[\mathcal{R}]_{\mathcal{Q}}|\mathcal{Q}'$ is almost bounded, then $\mathcal{R}|\mathcal{Q}'$ is almost bounded too.

Proof. The proof is analogous to the proof of Lemma C.19. □

The following theorem says that unbounded self-composition holds true for the computational indistinguishability relation. From this theroem, we will easily obtain Theorem C.32, i.e., unbounded self-composition for strong simulatability.

Theorem C.35. *Let R be a restriction. Let σ be an SID function and let $\mathcal{P}$ be a protocol system such that $\mathcal{P}$ is a σ-session version. Let $\mathcal{Q}$ be a system of IITMs compatible with $\mathcal{P}$ such that $\mathcal{Q}$ is a σ-session version, every machine in $\mathcal{Q}$ that is not in the scope of a bang accepts all messages in mode* **CheckAddress** *and $\mathcal{E}|\mathcal{Q}$ is almost bounded for all $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{Q})$.*[19]

If $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{Q}$ for all $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$, then $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{Q}$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ and $\mathcal{Q}$ is R-environmentally bounded.

Proof. Let $\mathcal{P}, \mathcal{Q}$ be systems of IITMs as required by the theorem such that $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{Q}$ for all $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$. Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$. In the following, by a hybrid argument, where we replace the copies of $\mathcal{P}$ by copies of $\mathcal{Q}$ with the same SID, we show that $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{Q}$. Then, we use this result to show that $\mathcal{Q}$ is R-environmentally bounded.

By Lemma 6 from [22], we may assume that $\mathcal{E}$ is a single IITM which, in mode **CheckAddress**, accepts all messages. We may also assume that start is an external tape of $\mathcal{E}$ (otherwise there would be no master IITM in $\mathcal{E}|\mathcal{P}$ and $\mathcal{E}|\mathcal{Q}$, i.e. every run immediately ends with empty output. Thus, $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{Q}$ trivially holds true in this case). In addition, we may assume that the only external tapes of $\mathcal{E} \,|\, \mathcal{P}$ (and hence, $\mathcal{E} \,|\, \mathcal{Q}$) are start and decision. Moreover, we may assume, without loss of generality, that $\mathcal{E}$ is such that every message m that $\mathcal{E}$ outputs on tape c (except if m is output on tape decision) has some SID, i.e. $\sigma(m, c) \neq \bot$: Since $\mathcal{E}$ will interact only with σ-session versions, messages without an SID would be rejected by these σ-session versions anyway. Since $\mathcal{E}$ is universally bounded, it follows that there exists a polynomial p_ϵ such that the number of different sessions (i.e. messages with distinct SIDs output by $\mathcal{E}$) is bounded from above by $p_\epsilon(\eta, |a|)$ (where η is the security parameter and a is the external input given to $\mathcal{E}$).

In what follows, let $\mathcal{Q}'$ be the variant of $\mathcal{Q}$ obtained from $\mathcal{Q}$ by renaming every tape c occurring in $\mathcal{Q}$ to c'. Analogously, let $\mathcal{P}''$ be obtained from $\mathcal{P}$ by renaming every tape c occurring in $\mathcal{P}$ to c''. By this, we

[19] Note that $\mathcal{Q}$ is essentially a protocol system with a relaxed runtime bound.

have that $\mathcal{P}$, $\mathcal{Q}'$, and $\mathcal{P}''$ have pairwise disjoint sets of external tapes, and hence, these systems are pairwise connectable.

We now define an IITM $\mathcal{E}_r$ (for every $r \in \mathbb{N}$) which essentially simulates $\mathcal{E}$ and which will run in the system $\mathcal{E}_r|\mathcal{P}''|\mathcal{Q}'|\mathcal{P}$ or $\mathcal{E}_r|\mathcal{P}''|\mathcal{Q}'|\mathcal{Q}$, respectively. The IITM $\mathcal{E}_r$ randomly shuffles the copies of the systems $\mathcal{P}$ and $\mathcal{Q}$, respectively, invoked by $\mathcal{E}$ by choosing a random permutation on $\{1, \ldots, p_\epsilon(\eta, |a|)\}$. Intuitively, this permutation is needed because $\mathcal{E}$ must not "know" whether the copy invoked in the system $\mathcal{P}''|\mathcal{Q}'|\mathcal{P}$ ($\mathcal{P}''|\mathcal{Q}'|\mathcal{Q}$) is a copy of $\mathcal{P}''$, $\mathcal{Q}'$, or $\mathcal{P}$ ($\mathcal{P}''$, $\mathcal{Q}'$, or $\mathcal{Q}$). This is needed in the proof of Lemma C.40. The first $r-1$ copies (after shuffling) of the protocol invoked by $\mathcal{E}$ will be copies of $\mathcal{Q}'$, the r-th copy will be the external system $\mathcal{P}$ or $\mathcal{Q}$, respectively, and the remaining copies will be copies of $\mathcal{P}''$.

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 also always accept every message 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 $\mathcal{Q}'$ and $\mathcal{P}''$, respectively.

Next, we need to specify how $\mathcal{E}_r$ redirects protocol invocations of $\mathcal{E}$ in the way sketched 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_ϵ, it will always hold that $l \leq p_\epsilon(\eta, |a|)$. Furthermore, in the first activation with security parameter $\eta \in \mathbb{N}$ and external input $a \in \{0,1\}^*$, $\mathcal{E}_r$ chooses a permutation π of $\{1, \ldots, p_\epsilon(\eta, |a|)\}$ uniformly at random. From now on, $\mathcal{E}_r$ simulates $\mathcal{E}$ 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 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 tape c of $\mathcal{P}$ (and hence, $\mathcal{Q}$) with $s := \sigma(m, c)$, then $\mathcal{E}_r$ 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 $\mathcal{E}_r$ writes m on tape c'.
   b) If $\pi(j) = r$, then $\mathcal{E}_r$ writes m on c.
   c) If $\pi(j) > r$, 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 $\mathcal{P}''$ corresponding to an external tape c of $\mathcal{P}$), 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 $\mathcal{Q}'$ corresponding to an external tape c of $\mathcal{Q}$), then $\mathcal{E}_r$ behaves as $\mathcal{E}$ in case input was received on tape c.
4. If $\mathcal{E}_r$ receives input on tape c (where c is an external tape of $\mathcal{P}$ or $\mathcal{Q}$, respectively), then $\mathcal{E}_r$ behaves as $\mathcal{E}$ in case input was received on tape c.

It is easy to see that $\mathcal{E}_r$ is universally bounded for every $r \in \mathbb{N}$ since $\mathcal{E}$ is universally bounded. We define the following hybrid systems, for every $r \in \mathbb{N}$:

$$\mathcal{H}_r := \mathcal{E}_r|\mathcal{P}''|\mathcal{Q}' \ ,$$

which can be connected to $\mathcal{P}$ (and hence $\mathcal{Q}$).

By $\mathcal{E}_{p_\epsilon}$ and $\mathcal{H}_{p_\epsilon}$ we denote the systems which first set $r = p_\epsilon(\eta, |a|)$ and then behave exactly as $\mathcal{E}_r$ and $\mathcal{H}_r$, respectively. By construction, for every $r \in \mathbb{N}$, the systems $\mathcal{E}|\mathcal{P}$ and $\mathcal{H}_1|\mathcal{P}$, the systems $\mathcal{H}_r|\mathcal{Q}$ and $\mathcal{H}_{r+1}|\mathcal{P}$, and the systems $\mathcal{E}|\mathcal{Q}$ and $\mathcal{H}_{p_\epsilon}|\mathcal{Q}$, respectively, behave exactly the same (see below). In particular, for all $r \in \mathbb{N}$, we have that:

$$\mathcal{E}|\mathcal{P} \equiv \mathcal{H}_1|\mathcal{P} \ , \tag{1}$$

$$\mathcal{H}_r|\mathcal{Q} \equiv \mathcal{H}_{r+1}|\mathcal{P} \ , \text{ and} \tag{2}$$

$$\mathcal{E}|\mathcal{Q} \equiv \mathcal{H}_{p_\epsilon}|\mathcal{Q} \ . \tag{3}$$

For (1) we need that $\mathcal{P}$ is a protocol system and a σ-session version:

The second property is necessary since it guarantees that two instances of machines in $\mathcal{P}$ with different SIDs will never send messages to each other (since they only send messages with their own SID or produce empty output, which activates the environment), i.e. it is no problem that the systems $\mathcal{P}''$ and $\mathcal{P}$ in $\mathcal{H}_1|\mathcal{P}$ do not have a connection to each other.

Furthermore, for machines M of $\mathcal{P}$ that are in the scope of a bang, it is easy to see that every instance of such a machine in a run of $\mathcal{E}|\mathcal{P}$ corresponds to one instance of this machine (either in $\mathcal{P}''$ or $\mathcal{P}$, depending on the SID of the instance) in a run of $\mathcal{H}_1|\mathcal{P}$.

If there is a machine M in $\mathcal{P}$ that is not in the scope of a bang the following problem can occur: In a run of $\mathcal{E}|\mathcal{P}$ there can only be at most one instance of such a machine, while there may be two instances (one in $\mathcal{P}''$ and one in $\mathcal{P}$) in a run of $\mathcal{H}_1|\mathcal{P}$, i.e. one of those instances would not have a corresponding instance of M in $\mathcal{E}|\mathcal{P}$. This is where we need that $\mathcal{P}$ is a protocol system, i.e. such a machine M must accept all messages m on all tapes c in mode **CheckAddress**. Since M is a σ-session version, it accepts only messages with the same SID and thus we obtain $\sigma(m,c) = s$ for some fixed value $s \neq \bot$, $m \in \{0,1\}^*$, and all input tapes c of M. (If $\sigma(m,c)$ were $\bot$ for some $m \in \{0,1\}^*$ and some input tape c of M, then M would accept m, in contradiction to the assumption that M is a σ-session version. Similarly, if there would exist (m,c) and (m',c') with $\sigma(m,c) \neq \sigma(m',c')$, then M would accept both (m,c) and (m',c'), again in contradiction to the assumption that M is a σ-session version.) Since $\mathcal{E}_1$ never sends messages with the same SID to both $\mathcal{P}$ and $\mathcal{P}''$ there will be at most one copy of either $\mathcal{P}$ or $\mathcal{P}''$ with SID s. But then there is also only one instance of M in a run of $\mathcal{H}_1|\mathcal{P}$, which corresponds to the single instance of M in a run of $\mathcal{E}|\mathcal{P}$.

A similar argument is used for (2) and (3).

Since $\mathcal{E}|\mathcal{P}$ is almost bounded by assumption (i.e. $\mathcal{P}$ is a protocol system and thus R-environmentally bounded), it is easy to see that $\mathcal{H}_1|\mathcal{P}$ is almost bounded too. Moreover, it is easy to see that $\mathcal{E}|\mathcal{Q}$ is almost bounded if and only if $\mathcal{H}_{p_\epsilon}|\mathcal{Q}$ is almost bounded.

We proceed by showing the following claim.

Claim I. $\mathcal{H}_{p_\epsilon}|\mathcal{Q}$ is almost bounded.

This is a crucial step of the proof. For this, we first have to define some events that describe the runtime of systems and the probability of answering a restricting message incorrectly.

For different systems, we define the event (i.e. a set of runs or equivalently a set of random coins) that the j-th copy of the system takes more than $q(\eta,|a|)$ steps. More specifically, for every polynomial q, natural numbers $r,i,j,\eta \in \mathbb{N}$, 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 instances of machines in $\mathcal{Q}'$ with SID $L[j]$ took more than $q(\eta,|a|)$ steps in ρ, where only the steps in mode **Compute** are counted; 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 $\mathcal{P}$ (with SID $L[j]$) took more than $q(\eta,|a|)$ steps in ρ, with the steps counted as above.
   c) $\pi(j) > r$ and the instances of machines in $\mathcal{P}''$ with SID $L[j]$ took more than $q(\eta,|a|)$ steps in ρ, with the steps counted as above.
2. $B^{q}_{\mathcal{H}_r|\mathcal{P}} := \bigcup_{i\in\mathbb{N}} B^{q,i}_{\mathcal{H}_r|\mathcal{P}}$.
3. $B^{q,\neq j}_{\mathcal{H}_r|\mathcal{P}} := \bigcup_{i\in\mathbb{N}\setminus\{j\}} B^{q,i}_{\mathcal{H}_r|\mathcal{P}}$.
4. A run ρ belongs to $B^{q,\pi^{-1}(i)}_{\mathcal{H}_r|\mathcal{P}}$ iff for the permutation, say π', chosen in ρ, it holds that $\rho \in B^{q,j}_{\mathcal{H}_r|\mathcal{P}}$ with $j = \pi'^{-1}(i)$. The event $B^{q,\neq\pi^{-1}(i)}_{\mathcal{H}_r|\mathcal{P}}$ is defined analogously.
5. Analogously to the above events, we define the following events where the external system $\mathcal{P}$ is replaced by $\mathcal{Q}$: $B^{q,j}_{\mathcal{H}_r|\mathcal{Q}}$, $B^{q}_{\mathcal{H}_r|\mathcal{Q}}$, $B^{q,\neq j}_{\mathcal{H}_r|\mathcal{Q}}$, $B^{q,\pi^{-1}(i)}_{\mathcal{H}_r|\mathcal{Q}}$, $B^{q,\neq\pi^{-1}(i)}_{\mathcal{H}_r|\mathcal{Q}}$.

We note that for all r, the system $\mathcal{H}_r|\mathcal{P}$ (resp., $\mathcal{H}_r|\mathcal{Q}$) is almost bounded if and only if there exists a negligible function f and a polynomial q such that $\Pr\left[B^{q}_{\mathcal{H}_r|\mathcal{P}}(1^\eta,a)\right] \leq f(\eta,a)$ (resp., $\Pr\left[B^{q}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a)\right] \leq f(\eta,a)$) for

all $\eta \in \mathbb{N}$ 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 $p_\epsilon(\eta, |a|)$). 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 $\mathcal{H}_r|\mathcal{P}$ (resp., $\mathcal{H}_r|\mathcal{Q}$) is polynomially bounded.

Furthermore, for a system $\mathcal{S}$ with tape start and a polynomial q we define the single IITM $[\mathcal{S}]_q$ to be the IITM that simulates $\mathcal{S}$ but simulates at most $q(\eta, |a|)$ many steps of $\mathcal{S}$ in mode **Compute**. That is, $[\mathcal{S}]_q$ works just like the definition of $[\mathcal{S}]_\mathcal{R}$ (for some system $\mathcal{R}$ such that $\mathcal{S}|\mathcal{R}$ is almost bounded) from Lemma C.16 but uses a fixed polynomial q instead of the runtime bound of $\mathcal{S}|\mathcal{R}$.

Finally, we also need to define some events that include runs where a restricting message from some part of a system is answered incorrectly. More specifically:

1. For all systems $\mathcal{S}, \mathcal{R}$ the event $C_\mathcal{S}^\mathcal{R} = C_\mathcal{S}^\mathcal{R}(1^\eta, a)$ is defined to be the set of all runs of $\mathcal{S}|\mathcal{R}$ where a restricting message of $\mathcal{R}$ is answered incorrectly.
2. For all $r \in \mathbb{N}$ and polynomials q the event $\hat{C}_{r,q}^\mathcal{P} = \hat{C}_{r,q}^\mathcal{P}(1^\eta, a)$ is defined to be the set of all runs of $[\mathcal{H}_r]_q|\mathcal{P} = [\mathcal{E}_r|\mathcal{P}''|\mathcal{Q}']_q|\mathcal{P}$ where a restricting message of $\mathcal{P}$, $\mathcal{P}''$, or $\mathcal{Q}'$ is answered incorrectly (by $\mathcal{E}_r$).
3. Analogous to $\hat{C}_{r,q}^\mathcal{P}$ we define $\hat{C}_{r,q}^\mathcal{Q}$ by replacing all occurences of $\mathcal{P}$ with $\mathcal{Q}$.

After having fixed some definitions, we can now start proving Claim I. First, we present a lemma which shows that, for all $r \in \mathbb{N}$, there exists a polynomial q_r such that the event $B_{\mathcal{H}_r|\mathcal{Q}}^{q_r}$ occurs with negligible probability. As argued above, this is equivalent to $\mathcal{H}_r|\mathcal{Q}$ being almost bounded. This however does not suffice to prove that $\mathcal{H}_{p_\epsilon}|\mathcal{Q}$ is almost bounded, i.e. we need a uniform bound that is independent of r (as will be established in Lemma C.40). This is in fact one of the main difficulties in proving Claim I and the only reason the permutation π is required. Additionally, the next lemma also shows some properties of the system $\mathcal{E}_r$ which are required by the following lemmas.

Lemma C.36. *For all $r \in \mathbb{N}$ the following holds true:*

1. *There exists a polynomial q_r and a negligible function f_r such that for all $\eta \in \mathbb{N}$ and $a \in \{0,1\}^*$:*
$$\Pr\left[B_{\mathcal{H}_r|\mathcal{Q}}^{q_r}(1^\eta, a)\right] \leq f_r(\eta, a) \ .$$
2. *$\mathcal{E}_r \in \mathsf{Env}_R(\mathcal{P}''|\mathcal{Q}'|\mathcal{P})$ and $\mathcal{E}_r \in \mathsf{Env}_R(\mathcal{P}''|\mathcal{Q}'|\mathcal{Q})$.*

Proof. The case $r = 0$ is simple: By construction the system $\mathcal{E}_0$ in a run of $\mathcal{E}_0|\mathcal{P}''|\mathcal{Q}'|\mathcal{P}$ sends all messages to $\mathcal{P}''$, i.e., $\mathcal{Q}'$ and $\mathcal{P}$ will never be activated by $\mathcal{E}_0$ and, since they do not contain a master IITM, they are also not actived via start. Thus, we can drop $\mathcal{Q}'$ and $\mathcal{P}$ and only look at $\mathcal{E}_0|\mathcal{P}''$, which behaves just as $\mathcal{E}|\mathcal{P}$. Since $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ and $\mathcal{E}|\mathcal{P}$ is almost bounded (because $\mathcal{P}$ is R-environmentally bounded), we have that $\mathcal{E}_0 \in \mathsf{Env}_R(\mathcal{P}''|\mathcal{Q}'|\mathcal{P})$ and $\mathcal{E}_0|\mathcal{P}''|\mathcal{Q}'|\mathcal{P}$ is almost bounded. We can also easily define a bijection between runs of $\mathcal{E}_0|\mathcal{P}''|\mathcal{Q}'|\mathcal{P}$ and $\mathcal{E}_0|\mathcal{P}''|\mathcal{Q}'|\mathcal{Q}$, since $\mathcal{P}$ and $\mathcal{Q}$, respectively, are never activated in any run. By this, we obtain that $\mathcal{E}_0 \in \mathsf{Env}_R(\mathcal{P}''|\mathcal{Q}'|\mathcal{Q})$ and $\mathcal{E}_0|\mathcal{P}''|\mathcal{Q}'|\mathcal{Q}$ is almost bounded (the latter implies that q_r and f_r for $r = 0$ exist, as argued above).

Now, let $r > 0$ and suppose that the lemma holds for $r-1$, i.e. $\mathcal{H}_{r-1}|\mathcal{Q}$ is almost bounded, $\mathcal{E}_{r-1} \in \mathsf{Env}_R(\mathcal{P}''|\mathcal{Q}'|\mathcal{P})$, and $\mathcal{E}_{r-1} \in \mathsf{Env}_R(\mathcal{P}''|\mathcal{Q}'|\mathcal{Q})$. Because $\mathcal{H}_{r-1}|\mathcal{Q}$ and $\mathcal{H}_r|\mathcal{P}$ behave exactly the same (see Equation (2)), $\mathcal{E}_{r-1} \in \mathsf{Env}_R(\mathcal{P}''|\mathcal{Q}'|\mathcal{Q})$ implies $\mathcal{E}_r \in \mathsf{Env}_R(\mathcal{P}''|\mathcal{Q}'|\mathcal{P})$ and $\mathcal{H}_{r-1}|\mathcal{Q}$ being almost bounded implies that $\mathcal{H}_r|\mathcal{P}$ is almost bounded. We also know that restricting messages from $\mathcal{P}$ are answered correctly in an overwhelming set of runs of $\mathcal{H}_r|\mathcal{P}$, since all external tapes of $\mathcal{P}$ are external tapes of $\mathcal{P}''|\mathcal{Q}'|\mathcal{P}$ and $\mathcal{E}_r$ is a responsive environment for $\mathcal{P}''|\mathcal{Q}'|\mathcal{P}$. Now Lemma C.16 and Lemma C.18 imply $\mathcal{H}_r|\mathcal{P} \equiv [\mathcal{H}_r]_\mathcal{P}|\mathcal{P}$ with $[\mathcal{H}_r]_\mathcal{P} \in \mathsf{Env}_R(\mathcal{P})$. Note that $[\mathcal{H}_r]_\mathcal{P}$ is a σ-single session restricted environment, i.e. $[\mathcal{H}_r]_\mathcal{P} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$, since the only external tapes (except for start and $\mathsf{decision}$) of $[\mathcal{H}_r]_\mathcal{P}$ are those that connect $\mathcal{E}_r$ and $\mathcal{P}/\mathcal{Q}$ and by construction, $\mathcal{E}_r$ sends only messages with the same SID on these tapes. By Lemma C.33 we obtain that $[\mathcal{H}_r]_\mathcal{P} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{Q})$, and in particular, that $[\mathcal{H}_r]_\mathcal{P}|\mathcal{Q}$ is almost bounded. Finally, by Lemma C.34, we also obtain $[\mathcal{H}_r]_\mathcal{P}|\mathcal{Q} \equiv \mathcal{H}_r|\mathcal{Q}$ and that $\mathcal{H}_r|\mathcal{Q}$ is almost bounded, which implies the existance of q_r and f_r.

We still have to show that $\mathcal{E}_r \in \mathsf{Env}_R(\mathcal{P}''|\mathcal{Q}'|\mathcal{Q})$. Suppose this was not the case, i.e. a restricting message from $\mathcal{P}''|\mathcal{Q}'|\mathcal{Q}$ is answered incorrectly in a non-negligible set of runs. Then there must be a non-negligible set of runs of $[\mathcal{H}_r]_{\mathcal{P}}|\mathcal{Q}$ where a restricting message from $\mathcal{Q}$ or $\mathcal{P}''|\mathcal{Q}'$ (which are simulated by $[\mathcal{H}_r]_{\mathcal{P}}$) is answered incorrectly. This is because $[\mathcal{H}_r]_{\mathcal{P}}|\mathcal{Q}$ behaves just like $\mathcal{H}_r|\mathcal{Q}$ except for a negligible set of runs where a runtime bound is reached. Furthermore, because $\mathcal{E}_r \in \mathsf{Env}_R(\mathcal{P}''|\mathcal{Q}'|\mathcal{P})$ and every restricting message of $\mathcal{P}$ or $\mathcal{P}''|\mathcal{Q}'$ is a restricting message of $\mathcal{P}''|\mathcal{Q}'|\mathcal{P}$, we obtain with a similar argument that there must be a negligible set of runs of $[\mathcal{H}_r]_{\mathcal{P}}|\mathcal{P}$ where a restricting message from $\mathcal{P}$ or $\mathcal{P}''|\mathcal{Q}'$ is answered incorrectly. Now one can construct a new environment $\mathcal{E}' \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$ that distinguishes $\mathcal{P}$ and $\mathcal{Q}$. The system $\mathcal{E}'$ simulates $[\mathcal{H}_r]_{\mathcal{P}}$ (the system $[\mathcal{H}_r]_{\mathcal{P}}$ is universally bounded and thus can be simulated by an environment) but connects to all tapes of $\mathcal{P}$ and $\mathcal{Q}$, respectively, and produces different output on tape decision: It outputs 1 if and only if a restricting message from $\mathcal{P}''|\mathcal{Q}'$ or $\mathcal{P}$ and $\mathcal{Q}$, respectively, was answered incorrectly during the run. Note that $\mathcal{E}'$ can easily check this since it sees all messages on all tapes except for internal tapes of $\mathcal{P}$ and $\mathcal{Q}$, respectively, and R is decidable in polynomial time in the second component (also see the proof of Lemma C.11, which includes a more detailed discussion on why this can be done by a universally bounded system). It is also obvious that $\mathcal{E}' \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$ since $[\mathcal{H}_r]_{\mathcal{P}} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$. Since this contradicts the assumption that $\mathcal{P}$ and $\mathcal{Q}$ are indistinguishable for all σ-single session responsive environments, it must hold true that $\mathcal{E}_r \in \mathsf{Env}_R(\mathcal{P}''|\mathcal{Q}'|\mathcal{Q})$

This concludes the proof of Lemma C.36. □

In the next steps of the proof of Claim I we will define several systems that get r as input and then simulate $[\mathcal{H}_r]_q$ for some polynomial q. We need that such a system is a responsive environment, i.e., correctly answers restricting messages from $\mathcal{P}$ in an overwhelming set of runs. For this it is not sufficient to know that $[\mathcal{H}_r]_q \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$, i.e. that for every r there exists a negligible function that bounds $\Pr\left[C^{\mathcal{P}}_{[\mathcal{H}_r]_q}\right]$. Instead we need a universal function that bounds $\Pr\left[C^{\mathcal{P}}_{[\mathcal{H}_r]_q}\right]$ for all values of r (in some range). The next two lemmas show that such a function exists.

Lemma C.37. *Let q be a polynomial. Then there exists a negligible function $g_{1,q}$ such that for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta, |a|)$:*

$$\left|\Pr\left[\hat{C}^{\mathcal{P}}_{r,q}(1^\eta, a)\right] - \Pr\left[\hat{C}^{\mathcal{Q}}_{r,q}(1^\eta, a)\right]\right| \leq g_{1,q}(\eta, a) \ .$$

Proof. We prove this lemma for the case of non-uniform environments, that is, environments that obtain external input. Using the same technique as in [22] (see Appendix B), namely sampling of runs, one easily obtains a proof for uniform environments, i.e., those that do not obtain external input, as well.

We define an IITM $D \in \mathsf{Env}_R(\mathcal{P})$ which expects to receive (r, a) as external input and then simulates $[\mathcal{H}_r]_q$ with external input a and outputs 1 if and only if $[\mathcal{H}_r]_q$ is about to send an incorrect answer to a restricting message of $\mathcal{P}''$, $\mathcal{Q}'$, or $\mathcal{P}$ and $\mathcal{Q}$, respectively.

More formally, D is defined as follows. The IITM D has the same (external) tapes as $\mathcal{H}_r$. 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 $r \in \{1, \ldots, p_\epsilon(\eta, |a|)\}$ and $a \in \{0,1\}^*$. If the external input is not of this form, D outputs 0 on decision. Otherwise, D simulates the system $[\mathcal{H}_r]_q = [\mathcal{E}_r|\mathcal{P}''|\mathcal{Q}']_q$ with external input a, i.e. it first activates $\mathcal{E}_r$ with input a on start and then simulates all the machines in $[\mathcal{H}_r]_q$. If D receives input on a tape, then D forwards this input to the simulated $[\mathcal{H}_r]_q$. If $[\mathcal{H}_r]_q$ produces output, then D first checks whether the output is an incorrect response to a restricting message of $\mathcal{P}''$, $\mathcal{Q}'$, and $\mathcal{P}$ and $\mathcal{Q}$, respectively. Note that this is easy to check since D sees all restricting messages from those systems (recall that we assume that $\mathcal{E}$ and hence $\mathcal{E}_r$, $\mathcal{H}_r$, and D connect to all tapes of $\mathcal{P}$ and $\mathcal{Q}$, respectively) and R is decidable in polynomial time in the second component (again, see the proof of Lemma C.11, which includes a more detailed discussion on why this can be done by a universally bounded system). If the answer is an incorrect response, D outputs 1 on decision instead of sending the message. If the run terminates (i.e. $[\mathcal{H}_r]_q$ outputs something on decision or $[\mathcal{H}_r]_q$ produces empty output, which terminates the run because $[\mathcal{H}_r]_q$ is a master IITM) then D halts with output 0 on decision.

It is easy to see that D is universally bounded (i.e. $D \in \mathsf{Env}(\mathcal{P})$) and that it is a responsive environment for $\mathcal{P}$ and $\mathcal{Q}$ since it will never incorrectly answer a restricting message from $\mathcal{P}$ and $\mathcal{Q}$, respectively. It is also

obvious that D is a σ-single session environment, since it simulates $[\mathcal{H}_r]_q$ (for some r) and $\mathcal{H}_r$ only sends messages with a single SID to $\mathcal{P}$ and $\mathcal{Q}$, respectively. We have for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta)$:

$$\begin{aligned} \Pr\left[(D|\mathcal{P})(1^\eta,(r,a))=1\right] &= \Pr\left[\hat{C}^{\mathcal{P}}_{r,q}(1^\eta,a)\right] \text{ and} \\ \Pr\left[(D|\mathcal{Q})(1^\eta,(r,a))=1\right] &= \Pr\left[\hat{C}^{\mathcal{Q}}_{r,q}(1^\eta,a)\right] . \end{aligned} \tag{4}$$

Since $\mathcal{E}'|\mathcal{P} \equiv \mathcal{E}'|\mathcal{Q}$ for all $\mathcal{E}' \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$ and $D \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$, we have that $D|\mathcal{P} \equiv D|\mathcal{Q}$. In particular, there exists a negligible function $g'_{1,q}$ such that for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta)$:

$$\begin{aligned} g'_{1,q}(\eta,(r,a)) &\geq \left|\Pr\left[(D|\mathcal{P})(1^\eta,(r,a))=1\right] - \Pr\left[(D|\mathcal{Q})(1^\eta,(r,a))=1\right]\right| \\ &\overset{(4)}{=} \left|\Pr\left[\hat{C}^{\mathcal{P}}_{r,q}(1^\eta,a)\right] - \Pr\left[\hat{C}^{\mathcal{Q}}_{r,q}(1^\eta,a)\right]\right| . \end{aligned}$$

Let $g_{1,q}(\eta,a) := \max_{r \leq p_\epsilon(\eta,|a|)} g'_{1,q}(\eta,(r,a))$. It is easy to see that $g_{1,q}$ is a negligible function. Now, clearly we have that

$$\left|\Pr\left[\hat{C}^{\mathcal{P}}_{r,q}(1^\eta,a)\right] - \Pr\left[\hat{C}^{\mathcal{Q}}_{r,q}(1^\eta,a)\right]\right| \leq g_{1,q}(\eta,a)$$

for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $r \leq p_\epsilon(\eta,|a|)$. Note that $g_{1,q}$ does not depend on r (only on q). This concludes the proof of Lemma C.37. □

We can now use Lemma C.37 to prove that, for every polynomial q, the probability of $C^{\mathcal{P}}_{[\mathcal{H}_r]_q}$ is bounded by a negligible function $g_{2,q}$ which is independent of r (for $1 \leq r \leq p_\epsilon(\eta,|a|)$). Note that this also implies $[\mathcal{H}_r]_q \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$ since $[\mathcal{H}_r]_q$ is universally bounded and the only external tapes (except for start and decision) of $[\mathcal{H}_r]_q$ are those that connect $\mathcal{E}_r$ and $\mathcal{P}/\mathcal{Q}$ and by construction, $\mathcal{E}_r$ sends only messages with the same SID on these tapes.

Lemma C.38. *Let q be a polynomial. There exists a negligible function $g_{2,q}$ such that for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta,|a|)$:*

$$\Pr\left[C^{\mathcal{P}}_{[\mathcal{H}_r]_q}(1^\eta,a)\right] \leq g_{2,q}(\eta,a) .$$

Proof. We will show the following stronger statement instead: Let q be a polynomial. There exists a negligible function $g_{2,q}$ such that for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta,|a|)$ it holds true that $\Pr\left[\hat{C}^{\mathcal{P}}_{r,q}(1^\eta,a)\right] \leq g_{2,q}(\eta,a)$. Since $C^{\mathcal{P}}_{[\mathcal{H}_r]_q} \subseteq \hat{C}^{\mathcal{P}}_{r,q}$, the lemma trivially follows by using the same $g_{2,q}$.

First, let r = 1. Observe that $\Pr\left[\hat{C}^{\mathcal{P}}_{1,q}(1^\eta,a)\right]$ is bounded by a negligible function $g(\eta,a)$ since $\mathcal{E}_1 \in \mathsf{Env}_R(\mathcal{P}''|\mathcal{Q}'|\mathcal{P})$ by Lemma C.36.

Now let $r > 1$. We have that

$$\begin{aligned} \Pr\left[\hat{C}^{\mathcal{P}}_{r,q}(1^\eta,a)\right] &\overset{(2)}{=} \Pr\left[\hat{C}^{\mathcal{Q}}_{r-1,q}(1^\eta,a)\right] \\ &\overset{\text{Lemma C.37}}{\leq} \Pr\left[\hat{C}^{\mathcal{P}}_{r-1,q}(1^\eta,a)\right] + g_{1,q}(\eta,a). \end{aligned}$$

By induction on r, we obtain that

$$\begin{aligned} \Pr\left[\hat{C}^{\mathcal{P}}_{r,q}(1^\eta,a)\right] &\leq (r-1)\cdot g_{1,q}(\eta,a) + \Pr\left[\hat{C}^{\mathcal{P}}_{1,q}(1^\eta,a)\right] \\ &\leq (r-1)\cdot g_{1,q}(\eta,a) + g(\eta,a). \end{aligned}$$

We define

$$g_{2,q}(\eta,a) := p_\epsilon(\eta,a)\cdot g_{1,q}(\eta,a) + g(\eta,a)$$

and obtain

$$\Pr\left[\hat{C}^{\mathcal{P}}_{r,q}(1^\eta, a)\right] \leq g_{2,q}(\eta, a)$$

for all $1 \leq r \leq p_\epsilon(\eta, |a|)$. Note that $g_{2,q}$ is negligible and does not depend on r (only on q). This concludes the proof of Lemma C.38. □

We can now continue to prove that $\mathcal{H}_{p_\epsilon}|\mathcal{Q}$ is almost bounded. First, we will show in the next lemma that for all $r \leq p_\epsilon(\eta, |a|)$ the difference of the probabilities that the events $B^{q_1, \neq \pi^{-1}(r)}_{\mathcal{H}_r|\mathcal{P}}$ and $B^{q_1, \neq \pi^{-1}(r)}_{\mathcal{H}_r|\mathcal{Q}}$ occur is negligible (where q_1 is the polynomial from Lemma C.36). 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 C.39. *There exists a negligible function f' such that for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta, |a|)$:*

$$\left|\Pr\left[B^{q_1, \neq \pi^{-1}(r)}_{\mathcal{H}_r|\mathcal{P}}(1^\eta, a)\right] - \Pr\left[B^{q_1, \neq \pi^{-1}(r)}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta, a)\right]\right| \leq f'(\eta, a) \ .$$

Proof. Just as in Lemma C.37, we prove this lemma for the case of non-uniform environments. Again, using the same technique as in [22] (see Appendix B) one easily obtains a proof for uniform environments.

We define an IITM $D \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$ 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 exceeded by any of the internally simulated sessions. If the runtime bound is exceeded, D outputs 1 on decision. If the run stops and the runtime bound has not been exceeded, 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 $r \in \{1, \ldots, p_\epsilon(\eta, |a|)\}$ and $a \in \{0,1\}^*$. If the external input is not of this form, D outputs 0 on decision. Otherwise, D simulates the system $\mathcal{H}_r = \mathcal{E}_r|\mathcal{P}''|\mathcal{Q}'$ with external input a, i.e. it first activates $\mathcal{E}_r$ 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., $\mathcal{E}_r$ outputs something on decision or $\mathcal{E}_r$ produces empty output, which terminates the run because $\mathcal{E}_r$ 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., $D \in \mathsf{Env}(\mathcal{P})$) and that it is σ-single session. We now show that $D \in \mathsf{Env}_R(\mathcal{P})$: If the input of the system $D|\mathcal{P}$ is not of the expected form, D will never send a message to $\mathcal{P}$ and thus never violate the condition of responsive environments. So we only have to find a bound for those cases where the input is of the correct form, i.e. the input a' is of the form (r, a) with $r \in \{1, \ldots, p_\epsilon(\eta, |a|)\}$.

Since D is universally bounded, there exists a polynomial q such that the runtime of the simulated $\mathcal{H}_r$ in mode **Compute** is bounded by q in every run. Now, observe that $(D|\mathcal{P})(1^\eta, (r, a))$ and $([\mathcal{H}_r]_q|\mathcal{P})(1^\eta, a)$ behave in the same way, except that $(D|\mathcal{P})(1^\eta, (r, a))$ might terminate earlier than $([\mathcal{H}_r]_q|\mathcal{P})(1^\eta, a)$, namely if in a session simulated by D the runtime bound q_1 is reached. Hence, we have that $\Pr\left[C^{\mathcal{P}}_D(1^\eta, (r, a))\right] \leq \Pr\left[C^{\mathcal{P}}_{[\mathcal{H}_r]_q}(1^\eta, a)\right]$. This is because $\Pr\left[C^{\mathcal{P}}_D(1^\eta, a)\right] = \Pr\left[C^{\mathcal{P}}_{[\mathcal{H}_r]_q}(1^\eta, a) \cap \text{Runtime}_{q_1}\right]$ where Runtime_{q_1} is the event that a restricting message of $\mathcal{P}$ is answered incorrectly before any session of $\mathcal{P}''$ and $\mathcal{Q}'$ in $[\mathcal{H}_r]_q$ exceeds the runtime bound q_1. Now, by Lemma C.38 this implies $\Pr\left[C^{\mathcal{P}}_D(1^\eta, (r, a))\right] \leq g_{2,q}(\eta, a)$ (note that this bound is the same for every valid r, i.e. it is independent of r).

We define $g(\eta, a') := \max_{a \in \{0,1\}^{\leq |a'|}} g_{2,q}(\eta, a)$. It is easy to see that g is negligible. Since $g_{2,q}(\eta, a)$ bounds $\Pr\left[C^{\mathcal{P}}_D(1^\eta, (r, a))\right]$ for all $1 \leq r \leq p_\epsilon(\eta, |a|)$ and, by the definition of D, $\Pr\left[C^{\mathcal{P}}_D(1^\eta, a')\right] = 0$ for all a' such

that a' cannot be parsed as (r,a) for some r with $1 \leq r \leq p_\epsilon(\eta,|a|)$, we have that $\Pr\left[C_D^{\mathcal{P}}(1^\eta,a')\right] \leq g(\eta,a')$ for all $\eta \in \mathbb{N}$, $a' \in \{0,1\}^*$. This implies that $D \in \mathsf{Env}_R(\mathcal{P})$, and thus, $D \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$.

By this definition of D, we have for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta,|a|)$:

$$\begin{aligned} \Pr\left[(D|\mathcal{P})(1^\eta,(r,a))=1\right] &= \Pr\left[B_{\mathcal{H}_r|\mathcal{P}}^{q_1,\neq\pi^{-1}(r)}(1^\eta,a)\right] \text{ and} \\ \Pr\left[(D|\mathcal{Q})(1^\eta,(r,a))=1\right] &= \Pr\left[B_{\mathcal{H}_r|\mathcal{Q}}^{q_1,\neq\pi^{-1}(r)}(1^\eta,a)\right] . \end{aligned} \tag{5}$$

Since $\mathcal{E}'|\mathcal{P} \equiv \mathcal{E}'|\mathcal{Q}$ for all $\mathcal{E}' \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$ and $D \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$, we have that $D|\mathcal{P} \equiv D|\mathcal{Q}$. In particular, there exists a negligible function f such that for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta,|a|)$:

$$\begin{aligned} f(\eta,(r,a)) &\geq \left|\Pr\left[(D|\mathcal{P})(1^\eta,(r,a))=1\right] - \Pr\left[(D|\mathcal{Q})(1^\eta,(r,a))=1\right]\right| \\ &\overset{(5)}{=} \left|\Pr\left[B_{\mathcal{H}_r|\mathcal{P}}^{q_1,\neq\pi^{-1}(r)}(1^\eta,a)\right] - \Pr\left[B_{\mathcal{H}_r|\mathcal{Q}}^{q_1,\neq\pi^{-1}(r)}(1^\eta,a)\right]\right| . \end{aligned}$$

Let $f'(\eta,a) := \max_{r \leq p_\epsilon(\eta,|a|)} f(\eta,(r,a))$. It is easy to see that f' is a negligible function. Now, obviously we have that

$$\left|\Pr\left[B_{\mathcal{H}_r|\mathcal{P}}^{q_1,\neq\pi^{-1}(r)}(1^\eta,a)\right] - \Pr\left[B_{\mathcal{H}_r|\mathcal{Q}}^{q_1,\neq\pi^{-1}(r)}(1^\eta,a)\right]\right| \leq f'(\eta,a)$$

for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta,|a|)$. This concludes the proof of Lemma C.39. □

In the next step, we can finally prove that there is a polynomial q' such that there is a universal bound, i.e., one that is independent of r, for the probability of $B_{\mathcal{H}_r|\mathcal{Q}}^{q'}$. This directly implies that $\mathcal{H}_{p_\epsilon}|\mathcal{Q}$ is almost bounded, and hence, Claim I.

Lemma C.40. *There exists a polynomial q' and a negligible function f'' such that for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $0 \leq r \leq p_\epsilon(\eta,|a|)$:*

$$\Pr\left[B_{\mathcal{H}_r|\mathcal{Q}}^{q'}(1^\eta,a)\right] \leq f''(\eta,a) .$$

Proof. We will first prove a weaker version of the lemma which holds true only for $1 \leq r \leq p_\epsilon(\eta,|a|)$. For this we use the polynomial q_1 from Lemma C.36 and construct a negligible function g. Afterwards we use this result to show the lemma for $0 \leq r \leq p_\epsilon(\eta,|a|)$.

Let $\eta \in \mathbb{N}$, and $a \in \{0,1\}^*$. For all $1 \leq r \leq p_\epsilon(\eta,|a|)$ we define

$$t_r := t_r(1^\eta,a) := \Pr\left[B_{\mathcal{H}_r|\mathcal{Q}}^{q_1}(1^\eta,a)\right] .$$

We need to show that there exists a negligible function that bounds t_r from above for every $1 \leq r \leq p_\epsilon(\eta,|a|)$.

Let $1 \leq r \leq p_\epsilon(\eta,|a|)$. If $r=1$, then $t_r = t_1 \leq f_1(\eta,a)$ by Lemma C.36. 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:

$$\Pr\left[B_{\mathcal{H}_r|\mathcal{Q}}^{q_1,\pi^{-1}(r)}(1^\eta,a) \setminus B_{\mathcal{H}_r|\mathcal{Q}}^{q_1,\neq\pi^{-1}(r)}(1^\eta,a)\right] = \Pr\left[B_{\mathcal{H}_r|\mathcal{Q}}^{q_1,\pi^{-1}(j)}(1^\eta,a) \setminus B_{\mathcal{H}_r|\mathcal{Q}}^{q_1,\neq\pi^{-1}(j)}(1^\eta,a)\right] . \tag{6}$$

Intuitively, the event $B_1 := B_{\mathcal{H}_r|\mathcal{Q}}^{q_1,\pi^{-1}(r)}(1^\eta,a) \setminus B_{\mathcal{H}_r|\mathcal{Q}}^{q_1,\neq\pi^{-1}(r)}(1^\eta,a)$ 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 $B_2 := B_{\mathcal{H}_r|\mathcal{Q}}^{q_1,\pi^{-1}(j)}(1^\eta,a) \setminus B_{\mathcal{H}_r|\mathcal{Q}}^{q_1,\neq\pi^{-1}(j)}(1^\eta,a)$ says that the number of steps taken in the j-th 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 (6) 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, by (6) we have that:

$$\begin{aligned}
\Pr\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a)\right] &\geq \Pr\left[\bigcup_{j=1}^{r-1} B^{q_1,\pi^{-1}(j)}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a)\right] \\
&\geq \Pr\left[\bigcup_{j=1}^{r-1} B^{q_1,\pi^{-1}(j)}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a) \setminus B^{q_1,\neq\pi^{-1}(j)}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a)\right] \\
&= \sum_{j=1}^{r-1} \Pr\left[B^{q_1,\pi^{-1}(j)}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a) \setminus B^{q_1,\neq\pi^{-1}(j)}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a)\right] \\
&\overset{(6)}{=} (r-1)\cdot \Pr\left[B^{q_1,\pi^{-1}(r)}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a) \setminus B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a)\right] \quad .
\end{aligned} \tag{7}$$

We conclude that:

$$\begin{aligned}
t_r &= \Pr\left[B^{q_1}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a)\right] \\
&= \Pr\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a)\right] + \Pr\left[B^{q_1,\pi^{-1}(r)}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a) \setminus B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a)\right] \\
&\leq \frac{r}{r-1}\cdot \Pr\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta,a)\right] && \text{due to (7)} \\
&\leq \frac{r}{r-1}\cdot \left(\Pr\left[B^{q_1,\neq\pi^{-1}(r)}_{\mathcal{H}_r|\mathcal{P}}(1^\eta,a)\right] + f'(\eta,a)\right) && \text{due to Lemma C.39} \\
&\leq \frac{r}{r-1}\cdot \left(\Pr\left[B^{q_1}_{\mathcal{H}_r|\mathcal{P}}(1^\eta,a)\right] + f'(\eta,a)\right) \quad .
\end{aligned}$$

Since the systems $\mathcal{H}_r|\mathcal{P}$ and $\mathcal{H}_{r-1}|\mathcal{Q}$ behave exactly the same, we have that

$$\Pr\left[B^{q_1}_{\mathcal{H}_r|\mathcal{P}}(1^\eta,a)\right] = \Pr\left[B^{q_1}_{\mathcal{H}_{r-1}|\mathcal{Q}}(1^\eta,a)\right] = t_{r-1}$$

and we obtain the following simple recurrence relation:

$$t_r \leq \frac{r}{r-1}\cdot (t_{r-1} + f'(\eta,a)) \quad .$$

By induction on r it can be shown that:

$$t_r \leq \left(\prod_{j=1}^{r-1}\frac{j+1}{j}\right)\cdot t_1 + \left(\sum_{j=1}^{r-1}\prod_{i=j}^{r-1}\frac{i+1}{i}\right)\cdot f'(\eta,a) \quad . \tag{8}$$

From this we obtain for $1 \leq r \leq p_\epsilon(\eta,|a|)$:

$$\begin{aligned}
t_r &\leq \left(\prod_{j=1}^{r-1}\frac{j+1}{j}\right)\cdot t_1 + \left(\sum_{j=1}^{r-1}\prod_{i=j}^{r-1}\frac{i+1}{i}\right)\cdot f'(\eta,a) && \text{due to (8)} \\
&= r\cdot t_1 + \left(\sum_{j=1}^{r-1}\frac{r}{j}\right)\cdot f'(\eta,a) \\
&\leq r\cdot t_1 + (r-1)\cdot r\cdot f'(\eta,a) \\
&\leq p_\epsilon(\eta,|a|)\cdot f_1(\eta,a) + p^2_\epsilon(\eta,|a|)\cdot f'(\eta,a) && \text{due to Lemma C.36.}
\end{aligned}$$

Hence, with

$$g(\eta, a) := p_\epsilon(\eta, |a|) \cdot f_1(\eta, a) + p_\epsilon^2(\eta, |a|) \cdot f'(\eta, a)$$

we obtain that

$$\Pr\left[B^{q_1}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta, a)\right] = t_r \leq g(\eta, a) \ ,$$

for all $1 \leq r \leq p_\epsilon(\eta, |a|)$. Note that g is negligible and does not depend on r.

Now we still have to look at the case $r = 0$. By Lemma C.36 we already know that $\Pr\left[B^{q_0}_{\mathcal{H}_0|\mathcal{Q}}(1^\eta, a)\right] \leq f_0(\eta, a)$. We define the polynomial $q'(\eta, |a|) := q_0(\eta, |a|) + q_1(\eta, |a|)$. Observe that both the results for $r = 0$ and $1 \leq r \leq p_\epsilon(\eta)$ also hold for the polynomial q' since every run of $\mathcal{H}_r|\mathcal{Q}(1^\eta, a)$ that exceeds the runtime bound $q'(\eta, |a|)$ also exceeds both the runtime bounds $q_0(\eta, |a|)$ and $q_1(\eta, |a|)$ (note that q_0 and q_1 can not be negative). Furthermore we define the negligible function $f''(\eta, a) = f_0(\eta, a) + g(\eta, a)$. Now the following holds true for all $0 \leq r \leq p_\epsilon(\eta, |a|)$ by construction:

$$\Pr\left[B^{q'}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta, a)\right] \leq f''(\eta, a) \ .$$

Note that f'' does not depend on the value of r. This concludes the proof of Lemma C.40. □

Lemma C.40 immediately implies that $\Pr\left[B^{q'}_{\mathcal{H}_{p_\epsilon}|\mathcal{Q}}(1^\eta, a)\right] \leq f''(\eta, a)$ for all $\eta \in \mathbb{N}$ and $a \in \{0,1\}^*$. In particular, it follows that $\mathcal{H}_{p_\epsilon}|\mathcal{Q}$ is almost bounded, and hence, Claim I. By (3), this immediately implies that $\mathcal{E}|\mathcal{Q}$ is almost bounded (for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$).

Furthermore, Lemma C.40 also implies that $\Pr\left[B^{q'}_{\mathcal{H}_r|\mathcal{P}}(1^\eta, a)\right] \leq f''(\eta, a)$ for all $1 \leq r \leq p_\epsilon(\eta, |a|)$ since $\mathcal{H}_r|\mathcal{Q}$ and $\mathcal{H}_{r+1}|\mathcal{P}$ behave exactly the same (see Equation (2)). In fact, the only reason to include the case $r = 0$ in Lemma C.40 is to obtain this result.

After having bounded the probability for exceeding a certain runtime, we will now proceed in the proof of Theorem C.35 with showing that $\mathcal{P}$ and $\mathcal{Q}$ are indistinguishable by any $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$, i.e., $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{Q}$. For this we need just one more result: It is necessary to show that, for every polynomial q, all σ-single session responsive environments $[\mathcal{H}_r]_q$ (for $1 \leq r \leq p_\epsilon(\eta, |a|)$) can distinguish $\mathcal{P}$ and $\mathcal{Q}$ with only roughly the same probability. Note that this is stronger than what we have already shown (i.e. $[\mathcal{H}_r]_q \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$) since we have to find a single negligible function that holds for all possible values of r.

Lemma C.41. *Let q be a polynomial. There exists a negligible function $g_{3,q}$ such that the following holds true for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta, |a|)$:*

$$|\Pr\left[[\mathcal{H}_r]_q|\mathcal{P}(1^\eta, a) = 1\right] - \Pr\left[[\mathcal{H}_r]_q|\mathcal{Q}(1^\eta, a) = 1\right]| \leq g_{3,q}(\eta, a).$$

Proof. Just as in Lemmas C.37 and C.39, we prove this lemma for the case of non-uniform environments. Again, using the same technique as in [22] (see Appendix B) one easily obtains a proof for uniform environments.

This proof follows essentially the same idea as the proofs of Lemma C.37 and Lemma C.39. We define an IITM $D \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$ which expects to receive (r, a) as external input and then simulates $[\mathcal{H}_r]_q$ with external input a. The IITM D outputs 1 on tape decision if and only if the simulated $[\mathcal{H}_r]_q$ wants to output 1 on tape 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 $r \in \{1, \ldots, p_\epsilon(\eta, |a|)\}$ and $a \in \{0,1\}^*$. If the external input is not of this form, D outputs 0 on decision. Otherwise, D simulates the system $[\mathcal{H}_r]_q$ with external input a. If $[\mathcal{H}_r]_q$ produces output, then so does D and if D receives input, then D forwards this input to the simulated $[\mathcal{H}_r]_q$. If the run terminates (i.e. $[\mathcal{H}_r]_q$ outputs something on decision or $[\mathcal{H}_r]_q$ produces empty output, which terminates the run because $[\mathcal{H}_r]_q$ is a master IITM), then D terminates its run in the same way (i.e. it either outputs the same message on tape decision or produces empty output).

It is easy to see that D is universally bounded (i.e. $D \in \mathsf{Env}(\mathcal{P})$) and that it is σ-single session. By Lemma C.38 we also know that $\Pr\left[C^{\mathcal{P}}_{[\mathcal{H}_r]_q}(1^\eta, a)\right] \leq g_{2,q}(\eta, a)$ for all $1 \leq r \leq p_\epsilon(\eta, |a|)$. By this we have that $\Pr\left[C^{\mathcal{P}}_{D}(1^\eta, a')\right] \leq \max_{a \in \{0,1\}^{\leq |a'|}} g_{2,q}(\eta, a)$, which is negligible, and thus $D \in \mathsf{Env}_R(\mathcal{P})$. Overall, this implies that $D \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$.

We have for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta, |a|)$:

$$\begin{aligned} &\Pr\left[(D|\mathcal{P})(1^\eta, (r, a)) = 1\right] = \Pr\left[[\mathcal{H}_r]_q|\mathcal{P}(1^\eta, a) = 1\right] \text{ and} \\ &\Pr\left[(D|\mathcal{Q})(1^\eta, (r, a)) = 1\right] = \Pr\left[[\mathcal{H}_r]_q|\mathcal{Q}(1^\eta, a) = 1\right] \ . \end{aligned} \tag{9}$$

Since $\mathcal{E}'|\mathcal{P} \equiv \mathcal{E}'|\mathcal{Q}$ for all $\mathcal{E}' \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$ and $D \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$, we have that $D|\mathcal{P} \equiv D|\mathcal{Q}$. In particular, there exists a negligible function $g'_{3,q}$ such that for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta, |a|)$:

$$\begin{aligned} g'_{3,q}(\eta, (r, a)) &\geq \left|\Pr\left[(D|\mathcal{P})(1^\eta, (r, a)) = 1\right] - \Pr\left[(D|\mathcal{Q})(1^\eta, (r, a)) = 1\right]\right| \\ &\overset{(9)}{=} \left|\Pr\left[[\mathcal{H}_r]_q|\mathcal{P}(1^\eta, a) = 1\right] - \Pr\left[[\mathcal{H}_r]_q|\mathcal{Q}(1^\eta, a) = 1\right]\right| \ . \end{aligned}$$

Let $g_{3,q}(\eta, a) := \max_{r \leq p_\epsilon(\eta,|a|)} g'_{3,q}(\eta, (r, a))$. It is easy to see that $g_{3,q}$ is a negligible function. Now, obviously we have that

$$\left|\Pr\left[[\mathcal{H}_r]_q|\mathcal{P}(1^\eta, a) = 1\right] - \Pr\left[[\mathcal{H}_r]_q|\mathcal{Q}(1^\eta, a) = 1\right]\right| \leq g_{3,q}(\eta, a)$$

for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta, |a|)$. This concludes the proof of Lemma C.41. □

We can now conclude the proof of Theorem C.35. First we observe that there exists a single polynomial q that bounds the runtime of $\mathcal{E}_r$ (in every run) for all $1 \leq r \leq p_\epsilon(\eta, |a|)$. This is because every $\mathcal{E}_r$ just simulates $\mathcal{E}$, which is universally bounded, and because $r \leq p_\epsilon(\eta, |a|)$.

We define the polynomial $q''(\eta, |a|) := q(\eta, |a|) + p_\epsilon(\eta, |a|) \cdot q'(\eta, |a|)$, where q is the polynomial from the last paragraph and q' is the polynomial from Lemma C.40. Now we have that, for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta, |a|)$, the system $\mathcal{H}_r$ in a run of $\mathcal{H}_r|\mathcal{P}(1^\eta, a)$ will exceed the runtime $q''(\eta, |a|)$ in mode **Compute** only if at least one copy of $\mathcal{P}''$ or $\mathcal{Q}'$ exceeds the runtime $q'(\eta, |a|)$. That is, every run of $\mathcal{H}_r|\mathcal{P}(1^\eta, a)$ where $\mathcal{H}_r$ needs more than $q''(\eta, |a|)$ steps is in $B^{q'}_{\mathcal{H}_r|\mathcal{P}}(1^\eta, a)$. Analogously, we have that runs of $\mathcal{H}_r|\mathcal{Q}(1^\eta, a)$ where $\mathcal{H}_r$ exceeds the runtime $q''(\eta, |a|)$ must be in $B^{q'}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta, a)$.

Since $\Pr\left[B^{q'}_{\mathcal{H}_r|\mathcal{P}}(1^\eta, a)\right] \leq f''(\eta, a)$ and $\Pr\left[B^{q'}_{\mathcal{H}_r|\mathcal{Q}}(1^\eta, a)\right] \leq f''(\eta, a)$ for all $1 \leq r \leq p_\epsilon(\eta, |a|)$ (by Lemma C.40 and the remark following this lemma) we can conclude that the systems $\mathcal{H}_r|\mathcal{P}$ and $[\mathcal{H}_r]_{q''}|\mathcal{P}$ and $\mathcal{H}_r|\mathcal{Q}$ and $[\mathcal{H}_r]_{q''}|\mathcal{Q}$ only differ in a negligible set of runs. More formally, we have for all $\eta \in \mathbb{N}$, $a \in \{0,1\}^*$, and $1 \leq r \leq p_\epsilon(\eta, |a|)$:

$$\begin{aligned} &\left|\Pr\left[\mathcal{H}_r|\mathcal{P}(1^\eta, a) = 1\right] - \Pr\left[[\mathcal{H}_r]_{q''}|\mathcal{P}(1^\eta, a) = 1\right]\right| \leq f''(\eta, a) \text{ and} \\ &\left|\Pr\left[\mathcal{H}_r|\mathcal{Q}(1^\eta, a) = 1\right] - \Pr\left[[\mathcal{H}_r]_{q''}|\mathcal{Q}(1^\eta, a) = 1\right]\right| \leq f''(\eta, a) \ . \end{aligned} \tag{10}$$

By this, we obtain the following inequalities, which hold true for all $\eta \in \mathbb{N}$ and $a \in \{0,1\}^*$:

$$\begin{aligned} &\left|\Pr\left[\mathcal{E}|\mathcal{P}(1^\eta, a) = 1\right] - \Pr\left[\mathcal{E}|\mathcal{Q}(1^\eta, a) = 1\right]\right| \\ \overset{(1),(3)}{=} &\left|\Pr\left[\mathcal{H}_1|\mathcal{P}(1^\eta, a) = 1\right] - \Pr\left[\mathcal{H}_{p_\epsilon}|\mathcal{Q}(1^\eta, a) = 1\right]\right| \\ \leq &\sum_{r=1}^{p_\epsilon(\eta,|a|)} \left|\Pr\left[\mathcal{H}_r|\mathcal{P}(1^\eta, a) = 1\right] - \Pr\left[\mathcal{H}_r|\mathcal{Q}(1^\eta, a) = 1\right]\right| \\ \leq &\sum_{r=1}^{p_\epsilon(\eta,|a|)} \left|\Pr\left[\mathcal{H}_r|\mathcal{P}(1^\eta, a) = 1\right] - \Pr\left[[\mathcal{H}_r]_{q''}|\mathcal{P}(1^\eta, a) = 1\right]\right| \\ &+ \sum_{r=1}^{p_\epsilon(\eta,|a|)} \left|\Pr\left[[\mathcal{H}_r]_{q''}|\mathcal{P}(1^\eta, a) = 1\right] - \Pr\left[[\mathcal{H}_r]_{q''}|\mathcal{Q}(1^\eta, a) = 1\right]\right| \\ &+ \sum_{r=1}^{p_\epsilon(\eta,|a|)} \left|\Pr\left[[\mathcal{H}_r]_{q''}|\mathcal{Q}(1^\eta, a) = 1\right] - \Pr\left[\mathcal{H}_r|\mathcal{Q}(1^\eta, a) = 1\right]\right| \end{aligned}$$

$$\overset{(10),\ \text{Lemma C.41}}{\leq} \sum_{r=1}^{p_\epsilon(\eta,|a|)} (f''(\eta,a) + g_{3,q''}(\eta,a) + f''(\eta,a))$$
$$= p_\epsilon(\eta,|a|) \cdot (f''(\eta,a) + g_{3,q''}(\eta,a) + f''(\eta,a))$$

We define $f'''(\eta,a) := p_\epsilon(\eta,|a|) \cdot (f''(\eta,a) + g_{3,q''}(\eta,a) + f''(\eta,a))$. It is easy to see that f''' is negligible and that the following holds true:

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

This implies $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{Q}$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$.

To conclude the proof of Theorem C.35, it remains to show that $\mathcal{Q}$ is R-environmentally bounded. Since $\mathcal{P}$ and $\mathcal{Q}$ are indistinguishable for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$, we can apply Lemma C.11 to obtain $\mathsf{Env}_R(\mathcal{P}) = \mathsf{Env}_R(\mathcal{Q})$. We have already shown that $\mathcal{E}|\mathcal{Q}$ is almost bounded for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ by Lemma C.40 (see the comment following the lemma) and thus $\mathcal{E}|\mathcal{Q}$ is almost bounded for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{Q})$, i.e. $\mathcal{Q}$ is R-environmentally bounded.

This concludes the proof of Theorem C.35.

□

The following lemma corresponds to Lemma 14 in [22], but it is adapted to the case of responsive environments. With this lemma, we can then prove Theorem C.32.

Lemma C.42. *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 and $\mathcal{P} \leq_{R,\sigma\text{-single}} \mathcal{F}$. Then, there exists $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{R,\sigma\text{-single}}(\mathcal{F})$ such that $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{S}|\mathcal{F}$ for all $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$ and $\mathcal{S}$ is a single IITM which is σ-complete.*

Proof. The original proof uses that there exists a simulator $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{\sigma\text{-single}}(\mathcal{F})$ such that $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{S}|\mathcal{F}$ for all $\mathcal{E} \in \mathsf{Env}_{\sigma\text{-single}}(\mathcal{P})$. Then one can construct a new simulator $\mathcal{S}'$ that is defined to be σ-complete and essentially simulates $\mathcal{S}$ and some copies of $\mathcal{F}$ which only interact with $\mathcal{S}$ (except for a negligible set of runs). It is then shown that, by definition, $\mathcal{S}'$ is a σ-single session simulator and $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{S}'|\mathcal{F}$ for all $\mathcal{E} \in \mathsf{Env}_{\sigma\text{-single}}(\mathcal{P})$.

Now, if one uses a simulator $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{R,\sigma\text{-single}}(\mathcal{F})$ with $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{S}|\mathcal{F}$ for all $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$, then it follows with the same construction that $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{S}'|\mathcal{F}$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ and that $\mathcal{S}'$ fulfills the conditions of σ-single session responsive simulators regarding interfaces and runtime. Furthermore, since $\mathcal{E}|\mathcal{S}|\mathcal{F}$ and $\mathcal{E}|\mathcal{S}'|\mathcal{F}$ essentially behave the same for all $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$ (except for a negligible set of runs) there must only be a negligible set of runs of $\mathcal{E}|\mathcal{S}'|\mathcal{F}$ where a restricting message of $\mathcal{F}$ is answered incorrectly (if this were not the case, then there would also be a non-negligible set of runs of $\mathcal{E}|\mathcal{S}|\mathcal{F}$ where a restricting message is answered incorrectly, which contradicts $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{R,\sigma\text{-single}}(\mathcal{F})$). This implies $\mathcal{S}' \in \mathrm{Sim}^{\mathcal{P}}_{R,\sigma\text{-single}}(\mathcal{F})$.

□

We can now prove Theorem C.32:

Proof (Theorem C.32). By Lemma C.42, there exists $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{R,\sigma\text{-single}}(\mathcal{F})$ such that $\mathcal{S}$ is a σ-complete IITM and $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{S}|\mathcal{F}$ for all $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$. Since $\mathcal{S}$ is σ-complete, we can conclude that $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ for all $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$:[20] Because the environment $\mathcal{E}$ 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 $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$, which implies $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|\mathcal{S}|\mathcal{F} \equiv \mathcal{E}|!\,\mathcal{S}|\mathcal{F}$.

Next, we observe that $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ is almost bounded for all $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(!\,\mathcal{S}|\mathcal{F})$: Since $\mathcal{E}|\mathcal{S}|\mathcal{F} \equiv \mathcal{E}|\mathcal{P} \equiv \mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ for all $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$, we can use Lemma C.33 two times to obtain $\mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{S}|\mathcal{F}) = \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P}) = \mathsf{Env}_{R,\sigma\text{-single}}(!\,\mathcal{S}|\mathcal{F})$. Because $\mathcal{S}$ is a σ-single session responsive simulator, the system $\mathcal{E}|\mathcal{S}|\mathcal{F}$ is almost bounded for all $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{S}|\mathcal{F}) = \mathsf{Env}_{R,\sigma\text{-single}}(!\,\mathcal{S}|\mathcal{F})$. As argued above we can easily define a bijection between runs of $\mathcal{E}|\mathcal{S}|\mathcal{F}$ and runs of $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ for all σ-single session environments and thus $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ must also be almost bounded for all $\mathcal{E} \in \mathsf{Env}_{R,\sigma\text{-single}}(!\,\mathcal{S}|\mathcal{F})$.

[20] Note that '!' binds stronger than '|', and hence, the system $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ is the same as $\mathcal{E}|(!\,\mathcal{S})|\mathcal{F}$.

Finally, we note that, by construction, $!\mathcal{S}|\mathcal{F}$ is a σ-session version where every machine that does not occur in the scope of a bang accepts all messages in mode **CheckAddress**. Thus, we can use Theorem C.35 (with $\mathcal{Q} := !\mathcal{S}|\mathcal{F}$) to obtain $\mathcal{E}|\mathcal{P} \equiv \mathcal{E}|!\mathcal{S}|\mathcal{F}$ for all $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ and that $!\mathcal{S}|\mathcal{F}$ is R-environmentally bounded.

Let $\mathcal{E} \in \mathsf{Env}_R(\mathcal{P})$ (and thus, by Lemma C.11, $\mathcal{E} \in \mathsf{Env}_R(!\mathcal{S}|\mathcal{F})$). The remainder of the proof shows that the set of all runs of $\mathcal{E}|!\mathcal{S}|\mathcal{F}$ where a restricting message from $\mathcal{F}$ is answered incorrectly is negligible; this implies $!\mathcal{S} \in \mathrm{Sim}_R^{\mathcal{P}}(\mathcal{F})$ and thus $\mathcal{P} \leq_R \mathcal{F}$. We will first show that there exists a single negligible function which bounds the probability of an incorrectly answered restricting message of a copy of $\mathcal{F}$ for all possible copies. Since there are only polynomial many copies in any run, we can then construct another negligible function which bounds the probability that any copy of $\mathcal{F}$ receives an incorrect answer to a restricting message.

By Lemma 6 from [22], we may assume that $\mathcal{E}$ is a single IITM which, in mode **CheckAddress**, accepts all messages. We may also assume that start is a tape of $\mathcal{E}$ (otherwise there would be no master IITM in $\mathcal{E} \,|\, !\mathcal{S} \,|\, \mathcal{F}$, i.e. every run immediately ends with empty output. Thus, the claim holds true trivially). In addition, we may assume that the only external tapes of $\mathcal{E} \,|\, !\mathcal{S} \,|\, \mathcal{F}$ are start and $\mathsf{decision}$. Moreover, we may assume, again without loss of generality, that $\mathcal{E}$ is such that every message m that $\mathcal{E}$ outputs on tape c (except if m is output on tape $\mathsf{decision}$) has some SID, i.e. $\sigma(m, c) \neq \bot$: Since $\mathcal{E}$ will only interact with σ-session versions, messages without an SID would be rejected by these σ-session versions anyway. Since $\mathcal{E}$ is universally bounded, it follows that there exists a polynomial p_ϵ such that the number of different sessions (i.e. messages with distinct SIDs output by $\mathcal{E}$) is bounded from above by $p_\epsilon(\eta, |a|)$ (where η is the security parameter and a is the external input).

In what follows, let $\mathcal{S}'$ be the variant of $\mathcal{S}$ obtained from $\mathcal{S}$ by renaming every tape c occurring in $\mathcal{S}$ to c'. Analogously, let $\mathcal{F}'$ be obtained from $\mathcal{F}$ by renaming every tape c occurring in $\mathcal{F}$ to c'. By this, we have that $!\mathcal{S}|\mathcal{F}$ and $!\mathcal{S}'|\mathcal{F}'$ have pairwise disjoint sets of external tapes, and hence, these systems are pairwise connectable.

We now define an IITM $\mathcal{E}_r$ (for every $r \in \mathbb{N}$) which basically simulates $\mathcal{E}$ and which will run in the system $\mathcal{E}_r|!\mathcal{S}'|\mathcal{F}'|!\mathcal{S}|\mathcal{F}$. The r-th copy of the protocol invoked by (the simulated) $\mathcal{E}$ will be a copy of $!\mathcal{S}|\mathcal{F}$ and the remaining copies will be copies of $!\mathcal{S}'|\mathcal{F}'$.

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 also always accept every message 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 $!\mathcal{S}|\mathcal{F}$. For each such tape c, we add to $\mathcal{E}_r$ a tape c' to connect to the external tapes of $!\mathcal{S}'|\mathcal{F}'$.

Next, we need to specify how $\mathcal{E}_r$ redirects protocol invocations of $\mathcal{E}$ in the way sketched 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_ϵ, it will always hold that $l \leq p_\epsilon(\eta, |a|)$. After the first activation, $\mathcal{E}_r$ simulates $\mathcal{E}$ 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 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 tape c of $!\mathcal{S}|\mathcal{F}$ with $s := \sigma(m, c)$, then $\mathcal{E}_r$ 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 $j = r$, then $\mathcal{E}_r$ writes m on c.
   b) If $j \neq r$, 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 $!\mathcal{S}'|\mathcal{F}'$ corresponding to an external tape c of $!\mathcal{S}|\mathcal{F}$), 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 $!\mathcal{S}|\mathcal{F}$), then $\mathcal{E}_r$ behaves as $\mathcal{E}$ in case input was received on tape c.

It is easy to see that $\mathcal{E}_r$ is universally bounded for every $r \in \mathbb{N}$ since $\mathcal{E}$ is universally bounded.

We now define a system $\mathcal{E}_\$$ which first chooses some $r \in \{1, \ldots, p_\epsilon(\eta, |a|)\}$ uniformly at random and then behaves exactly as $\mathcal{E}_r$. It is easy to see that $\mathcal{E}_\$$ is universally bounded since $\mathcal{E}_\$$ simulates the universally

bounded system $\mathcal{E}_r$ and r is bounded by a polynomial. Furthermore, we define the system $\mathcal{H}_\$ = \mathcal{E}_\$|!\,\mathcal{S}'|\mathcal{F}'$ for brevity of presentation.

By construction, the systems $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ and $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$ behave exactly the same (see below). That is, there exists a mapping which maps a run ρ of $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ to a set of corresponding runs of $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$ such that the probability of ρ equals the probability of the corresponding set of runs of $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$, where the runs in this set differ only on the choice of r. In particular, we have that:

$$\mathcal{E}|!\,\mathcal{S}|\mathcal{F} \equiv \mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F} \ , \tag{11}$$

For (11) we need that $!\,\mathcal{S}|\mathcal{F}$ is a protocol system and a σ-session version. The second property is necessary since it guarantees that two instances of machines in $!\,\mathcal{S}|\mathcal{F}$ with different SIDs will never send messages to each other (since they only send messages with their own SID), i.e. it is no problem that the systems $!\,\mathcal{S}'|\mathcal{F}'$ and $!\,\mathcal{S}|\mathcal{F}$ in $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$ do not have a connection to each other. Furthermore, for machines M of $!\,\mathcal{S}|\mathcal{F}$ that are in the scope of a bang, it is easy to see that every instance of such a machine in a run of $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ corresponds to one instance of this machine (either in $!\,\mathcal{S}'|\mathcal{F}'$ or $!\,\mathcal{S}|\mathcal{F}$, depending on the SID of the instance) in a run of $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$.

If there is a machine M in $!\,\mathcal{S}|\mathcal{F}$ that is not in the scope of a bang the following problem can occur: In a run of $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ there can only be at most one instance of such a machine, while there may be two instances (one in $!\,\mathcal{S}'|\mathcal{F}'$ and one in $!\,\mathcal{S}|\mathcal{F}$) in a run of $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$, i.e. one of those instances would not have a corresponding instance of M in $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$. This is where we need that $!\,\mathcal{S}|\mathcal{F}$ is a protocol system, i.e. such a machine M must accept all messages m on all tapes c in mode **CheckAddress**. Since M is a σ-session version, i.e. accepts only messages with the same SID, we obtain $\sigma(m,c) = s$ for some fixed value $s \neq \bot$, $m \in \{0,1\}^*$, and all (input-)tapes c of M. Since $\mathcal{E}_r$ and thus $\mathcal{E}_\$$ never sends messages with the same SID to both $!\,\mathcal{S}'|\mathcal{F}'$ and $!\,\mathcal{S}|\mathcal{F}$ there will be at most one copy of either $!\,\mathcal{S}'|\mathcal{F}'$ or $!\,\mathcal{S}|\mathcal{F}$ with SID s. But then there is also at most one instance of M in a run of $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$, which corresponds to the single instance of M in a run of $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$.

Remember that we have to show that restricting messages of $\mathcal{F}$ are answered correctly in an overwhelming set of runs of $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$. For this, we need the following events: Recall from the proof of Lemma C.35 that, for all systems $\mathcal{Q}$ and $\mathcal{R}$, we define the event $C^{\mathcal{R}}_{\mathcal{Q}} = C^{\mathcal{R}}_{\mathcal{Q}}(1^\eta, a)$ to be the set of all runs of $\mathcal{Q}|\mathcal{R}$ where a restricting message of $\mathcal{R}$ is answered incorrectly. With $\overline{C^{\mathcal{R}}_{\mathcal{Q}}}$ we denote the complementary event.

In the following, we will first prove that $\Pr\left[C^{\mathcal{F}}_{\mathcal{H}_\$|!\,\mathcal{S}}\right]$ is negligible, which can then be used to conclude that $\Pr\left[C^{\mathcal{F}}_{\mathcal{E}|!\,\mathcal{S}}\right]$ is negligible. For this result, we need that $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$ is almost bounded: Observe that $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ is almost bounded, since $\mathcal{E} \in \mathsf{Env}_R(!\,\mathcal{S}|\mathcal{F})$ and $!\,\mathcal{S}|\mathcal{F}$ is R-environmentally bounded. Because $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$ and $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ behave exactly the same (see (11) and the explanation above this equation), it is easy to see $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$ is almost bounded as well.

It also holds true that $\mathcal{E}_\$ \in \mathsf{Env}_R(!\,\mathcal{S}'|\mathcal{F}'|!\,\mathcal{S}|\mathcal{F})$: Since the systems $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$ and $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ behave exactly the same we have that in a run ρ of $\mathcal{E}|!\,\mathcal{S}|\mathcal{F}$ a restricting message of $!\,\mathcal{S}|\mathcal{F}$ is not answered correctly iff in every run in the set of runs of $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$ corresponding to ρ this restricting message, which comes from $!\,\mathcal{S}|\mathcal{F}$ or $!\,\mathcal{S}'|\mathcal{F}'$, is not answered correctly as well. Since $\Pr\left[\overline{C^{!\,\mathcal{S}|\mathcal{F}}_{\mathcal{E}}}\right]$ is overwhelming, this directly implies that $\Pr\left[\overline{C^{!\,\mathcal{S}'|\mathcal{F}'|!\,\mathcal{S}|\mathcal{F}}_{\mathcal{E}_\$}}\right]$ is also overwhelming.

Now observe that every restricting message (on an external network tape) of $!\,\mathcal{S}|\mathcal{F}$ in the system $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$ is also a restricting message of $!\,\mathcal{S}'|\mathcal{F}'|!\,\mathcal{S}|\mathcal{F}$ (note that $!\,\mathcal{S}|\mathcal{F}$ and $!\,\mathcal{S}'|\mathcal{F}'$ have disjoint tapes). Thus we have $\Pr\left[C^{!\,\mathcal{S}|\mathcal{F}}_{\mathcal{H}_\$}\right] = \Pr\left[C^{!\,\mathcal{S}|\mathcal{F}}_{\mathcal{E}_\$|!\,\mathcal{S}'|\mathcal{F}'}\right] \leq \Pr\left[C^{!\,\mathcal{S}'|\mathcal{F}'|!\,\mathcal{S}|\mathcal{F}}_{\mathcal{E}_\$}\right]$, i.e., $\Pr\left[C^{!\,\mathcal{S}|\mathcal{F}}_{\mathcal{H}_\$}\right]$ must also be negligible.

By the previous observations, we can use Lemma C.16 and Lemma C.18 to obtain that $[\mathcal{H}_\$]_{!\,\mathcal{S}|\mathcal{F}} \in \mathsf{Env}_R(!\,\mathcal{S}|\mathcal{F})$. Furthermore, $[\mathcal{H}_\$]_{!\,\mathcal{S}|\mathcal{F}}$ is also a σ-single session environment, since the only external tapes (except for start and decision) of $[\mathcal{H}_\$]_{!\,\mathcal{S}|\mathcal{F}}$ are those between (the simulated) $\mathcal{E}_r$ and $!\,\mathcal{S}|\mathcal{F}$, and $\mathcal{E}_r$ only sends messages with the same SID on these tapes. Overall, we have that $[\mathcal{H}_\$]_{!\,\mathcal{S}|\mathcal{F}} \in \mathsf{Env}_{R,\sigma\text{-single}}(!\,\mathcal{S}|\mathcal{F})$, and thus, by Lemma C.33, $[\mathcal{H}_\$]_{!\,\mathcal{S}|\mathcal{F}} \in \mathsf{Env}_{R,\sigma\text{-single}}(\mathcal{P})$. Since $\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{R,\sigma\text{-single}}(\mathcal{F})$, this implies that $\Pr\left[C^{\mathcal{F}}_{[\mathcal{H}_\$]_{!\,\mathcal{S}|\mathcal{F}}|\mathcal{S}}\right] =$

$\Pr\left[C^{\mathcal{F}}_{[\mathcal{H}_\$]_{!\,\mathcal{S}|\mathcal{F}}|!\,\mathcal{S}}\right]$ is negligible. (Recall that $[\mathcal{H}_\$]_{!\,\mathcal{S}|\mathcal{F}}$ is a σ-single session environment and that $\mathcal{S}$ is σ-complete. Hence, the systems $[\mathcal{H}_\$]_{!\,\mathcal{S}|\mathcal{F}}|\mathcal{S}$ and $[\mathcal{H}_\$]_{!\,\mathcal{S}|\mathcal{F}}|!\,\mathcal{S}$ behave in exactly the same way.) Finally, because $[\mathcal{H}_\$]_{!\,\mathcal{S}|\mathcal{F}}|!\,\mathcal{S}|\mathcal{F}$ and $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$ behave exactly the same except for a negligible set of runs (i.e. when the runtime bound of $[\mathcal{H}_\$]_{!\,\mathcal{S}|\mathcal{F}}$ is reached), we can conclude that $\Pr\left[C^{\mathcal{F}}_{\mathcal{H}_\$|!\,\mathcal{S}}\right]$ is negligible.

We can now show that $\Pr\left[C^{\mathcal{F}}_{\mathcal{E}|!\,\mathcal{S}}\right]$ is negligible. First we observe that, for every run ρ in $C^{\mathcal{F}}_{\mathcal{E}|!\,\mathcal{S}}(1^\eta, a)$ there is a unique run ρ' in $C^{\mathcal{F}}_{\mathcal{H}_\$|!\,\mathcal{S}}(1^\eta, a)$ with $\Pr[\rho'] = \frac{1}{p_\epsilon(\eta,|a|)} \cdot \Pr[\rho]$.

More formally, we can define an injective function f that maps runs ρ from $C^{\mathcal{F}}_{\mathcal{E}|!\,\mathcal{S}}$ to runs ρ' of $C^{\mathcal{F}}_{\mathcal{H}_\$|!\,\mathcal{S}}$ in the following way: Let ρ be such a run. Then there exists at least one restricting message of $\mathcal{F}$ that is answered incorrectly. Let s be the SID of the instance of $\mathcal{F}$ (in the run ρ) which is the first instance that receives an incorrect answer. Let r' be such that s is the r'-th (unique) SID that is used by $\mathcal{E}$. We define ρ' to be the run of $\mathcal{H}_\$|!\,\mathcal{S}|\mathcal{F}$ where $\mathcal{H}_\$$ chooses $r = r'$ (which happens with probability $p_\epsilon(\eta, |a|)^{-1}$) and the run then continues just as ρ, i.e. all instances in ρ', after r is chosen, use the same random coins as those in ρ. Then ρ' is in $C^{\mathcal{F}}_{\mathcal{H}_\$|!\,\mathcal{S}}$ by construction. Furthermore, it is easy to see that f is injective and that $\Pr[\rho'] = \frac{1}{p_\epsilon(\eta,|a|)} \cdot \Pr[\rho]$. Now, we obtain:

$$\begin{aligned}
\Pr\left[C^{\mathcal{F}}_{\mathcal{E}|!\,\mathcal{S}}\right] &= \sum_{\rho \in C^{\mathcal{F}}_{\mathcal{E}|!\,\mathcal{S}}} \Pr[\rho] \\
&= \sum_{\rho \in C^{\mathcal{F}}_{\mathcal{E}|!\,\mathcal{S}}} p_\epsilon(\eta, |a|) \cdot \Pr[f(\rho)] \\
&\overset{f \text{ injective}}{\leq} \sum_{\rho' \in C^{\mathcal{F}}_{\mathcal{H}_\$|!\,\mathcal{S}}} p_\epsilon(\eta, |a|) \cdot \Pr[\rho'] \\
&= p_\epsilon(\eta, |a|) \cdot \Pr\left[C^{\mathcal{F}}_{\mathcal{H}_\$|!\,\mathcal{S}}\right] .
\end{aligned}$$

Since $\Pr\left[C^{\mathcal{F}}_{\mathcal{H}_\$|!\,\mathcal{S}}\right]$ is negligible, it follows that $\Pr\left[C^{\mathcal{F}}_{\mathcal{E}|!\,\mathcal{S}}\right]$ is bounded by a negligible function as well. This implies $!\,\mathcal{S} \in \mathrm{Sim}^{\mathcal{P}}_{R}(\mathcal{F})$, and thus, concludes the proof of Theorem C.32. □