Diving Deep Into UC: Uncovering and Resolving Issues in Universal Composability

Céline Chevalier[1,3,4,5,6] and Éric Sageloli[2,3,4,5,6]

[1] CRED, Paris-Panthéon-Assas University, Paris, France
[2] Thales, France
[3] DIENS, École normale supérieure, Paris, France
[4] CNRS, France
[5] PSL University, France
[6] Inria, France

Abstract. Introduced by Canetti in 2001, Universal Composability (UC) is a widely adopted security model that enables the specification and proof of security for a broad range of protocols, offering strong security guarantees. At its core lies the universal composition theorem (UC theorem), which ensures that protocols proven secure within the framework remain secure even when deployed in real-world environments with multiple instances of them.
In this work, we present two key contributions. First, we identify several problems with the UC framework, in particular the UC Theorem. They include counterexamples, limitations that make it unusable for important classes of protocols, and weaknesses in its proof. These problems reveal flaws in nearly all the fundamental concepts of UC.
Second, we propose a revised formulation of the main concepts of UC to address these issues. Although the resulting modifications are nontrivial, our updated definitions are designed to remain as faithful as possible to the structure and intent of the original model.

Keywords: Universal Composability · Composable Security · Protocol Security · Theoretical Foundations

1 Introduction

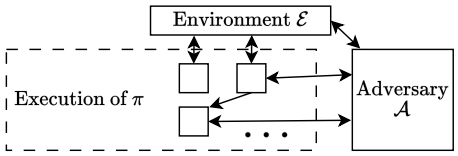
Introduced in 2001 by Canetti in [Can01], Universal Composability (UC) is a security framework renowned for its ability to define and prove the security of a wide range of protocols, as well as for the strong security guarantees it provides. A protocol proven secure within this framework remains secure even when deployed in real-world environments, including in multiple concurrent instances. To illustrate the variety of protocols recently studied under UC, we can cite PAKEs [JKX18, Hes20, CNPR22, BGHJ24], non-interactive zero-knowledge arguments [CDG+25], oblivious transfer [BC15], the WhatsApp backup protocol [DFG+23], the Signal protocol [CJSV22] and the Bitcoin protocol [BMTZ24].

In UC, it is considered that if a protocol contains a vulnerability, this vulnerability should have an impact on some environment of execution, attacked by some adversary. Consequently, when studying a protocol π, we consider its execution alongside two machines: one modeling an adversary and the other modeling the execution environment.

This paper is an extended version of the article [CS26], published in Volume 3, Issue 1 of *IACR Communications in Cryptology*. The main change in this version is the addition of Appendices C, D, and E, which provide missing proofs, constructions and descriptions of issues.

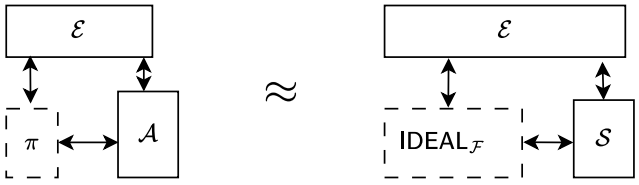
E-mail: eric.sageloli@protonmail.com (Éric Sageloli)

This work is licensed under a "CC BY 4.0" license.
Date of this document: 2026-05-05.



To define the security of a protocol, such as a PAKE [JKX18, Hes20, CNPR22, BGHJ24], or an oblivious transfer [BC15], we must first define a special machine $\mathcal{F}$, known as an ideal functionality. This ideal functionality is then used to define the protocol $\mathsf{IDEAL}_{\mathcal{F}}$, referred to as the ideal protocol associated with $\mathcal{F}$, which represents a idealized version of the protocol (*e.g.*, a ideal PAKE or oblivious transfert). The references listed at the beginning of this paragraph provide examples of ideal functionalities.

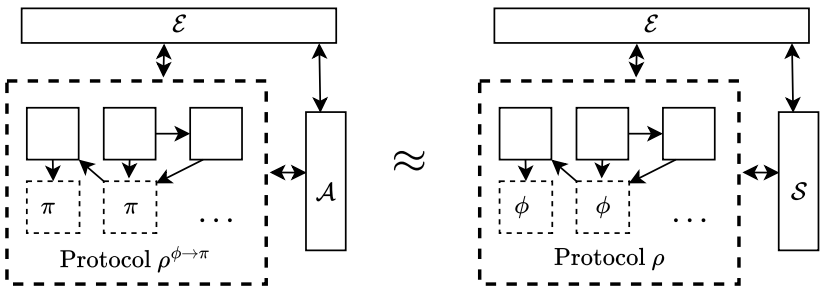
Then, the security of a protocol π is proven by showing that π is "as secure as" $\mathsf{IDEAL}_{\mathcal{F}}$ through a notion called UC-emulation. π is said to UC-emulate $\mathsf{IDEAL}_{\mathcal{F}}$ if, for any adversary $\mathcal{A}$ of π, there is an adversary $\mathcal{S}$ of $\mathsf{IDEAL}_{\mathcal{F}}$ such that no environment $\mathcal{E}$ can efficiently distinguish if it is interacting with π and $\mathcal{A}$ or $\mathsf{IDEAL}_{\mathcal{F}}$ and $\mathcal{S}$.



While the aim of $\mathcal{E}$ is to model any execution environment, such environments are typically not composed of a single machine, nor do they involve only one instance of the protocol. Instead, they can be viewed as protocols ρ with multiple machines, each using multiple instances of the studied protocol.

To justify modeling the environment as a single machine that can only use one instance of the protocol π, UC must ensure that this modeling is sufficient to cover situations involving complex execution environments. This is the goal of the main theorem of the framework, called the universal composition theorem [Can20, (Theorem 22)] (UC theorem).

The UC Theorem states that for protocols ϕ, π, and ρ, under certain hypotheses, if π UC-emulates ϕ, then a modification of ρ, denoted by $\rho^{\phi \to \pi}$, where uses of ϕ are replaced by uses of π, will also UC-emulate ρ.



Taking $\phi = \mathsf{IDEAL}_{\mathcal{F}}$, this implies that if π is secure within a simple execution environment – *i.e.*, if π UC-emulates $\mathsf{IDEAL}_{\mathcal{F}}$ – it remains secure when used in complex execution environments.

Table 1: Issues, Limitations and Weaknesses described in the main body.

Notions	Issues	Limitations	Weaknesses	
Extended Session and Subroutine Respecting	(I.1) (I.2) (I.3)	(L.1)		(Section 5.1)
UC Operator	(I.4)		(W.1)	(Section 5.2)
Subroutine Exposing		(L.2)	(W.2)	(Section 5.3)
Compliance			(W.3) (W.4)	(Section 5.4)

1.1 Our Contributions

As a first and main contribution, we identify several problems with the UC theorem [Can20, (Theorem 22)], classified as follows:

- *Issues (I):* problems that lead to counterexamples of the UC theorem.
- *Limitations (L):* difficulties that limit the applicability of the UC theorem.
- *Weaknesses (W):* gaps in the proof of the UC theorem.

These problems affect all security concepts associated with the UC Theorem, with multiple concerns identified for each. Our focus will be on these concepts within the initial and most detailed formulation of the UC framework that was last updated in 2020 [Can20], which we refer to as "Plain UC ". However, these problems naturally extend to its generalizations, which encompass it. These include UC with joint state [CR03], UC with global setup [CDPW07], and multi-protocol UC [CDT19].

Since the UC Theorem ensures that protocols proven UC-secure maintain their security during composition, our findings reveal that the current UC framework does not guarantee the preservation of security under composition – the very goal of UC.

To re-establish these security guarantees, in a second contribution, we propose an update to each security concept within Plain UC. These revised definitions address the identified issues. To support these updates, we also present an updated UC Theorem.

These updates restrict the applicability of the UC Theorem in certain respects – eliminating some of the issues identified earlier – while simultaneously extending it to protocols that were previously incompatible with the framework.

Finally, although case-by-case verification remains necessary, we believe that protocols proven secure under Plain UC generally satisfy the hypotheses of the UC Theorem in Updated UC.

Moreover, we believe that these updates offer a solid foundation for extending the main generalizations of Plain UC.

1.1.1 Problems in Plain UC and its generalizations.

The identified issues, limitations, and weaknesses presented in the main body exhibit problems with all the properties required by the UC Theorem. They are listed and classified into four categories in Table 1, and outlined in the Technical Overview (Section 2).

As a side contribution, Appendix D describes additional problems related to other concepts, such as the time-complexity notion used in UC and the concept of a Balanced Environment (See Technical Overview, Section 2.8).

1.1.2 Updated UC.

To address the identified problems, we propose updated definitions for Plain UC, as summarized in Table 2, where even similarly named notions exhibit differences. We refer to the framework with these updated definitions as Updated UC.

Table 2: Correspondence between Plain UC and Updated UC notions.

Plain UC [Can20]	Updated UC (this work)
Extended Session [Can20, (Section 3.1.2)]	Extended Session (Definition 3)
–	Panoptic Session (Definition 5)
Subroutine Respecting [Can20, (Section 5.2)]	Sub-party Respecting (Definition 9)
Universal Composition Operator [Can20, (Section 6.1)]	Universal Composition Operator (Section 6.7)
Subroutine Exposing [Can20, (Section 6.1)]	Sid-Exposing (Definition 8)
Compliant [Can20, (Section 6.1)]	Compliant (Definition 10)
–	Disjunction (Definition 6)
–	Separation (Definition 7)
Balanced Environment [Can20, (Section 4.2)]	–
–	Replacement Execution (Section 6.2)

Although most of these updates introduce non-trivial modifications, they remain grounded in the same computational model and core design principles. Moreover, while some updates restrict the framework's generality, this is done deliberately to exclude the counterexamples to the UC Theorem that we identified.

While individual verification is still required, we expect that existing protocols proven secure under Plain UC already meet the assumptions of the UC Theorem as formulated in Updated UC.

The justification for these updates is twofold.

First, we motivate the vast majority of the definitional changes by showing how they resolve specific issues previously identified. These explanations illustrate that, although some modifications are nontrivial, Updated UC should not be viewed as a distinct model, but rather as a necessary refinement of Plain UC to guarantee its correctness.[1]

Second, we provide a new formulation of the UC Theorem based on these updated definitions (and proven in Appendix C). While the main ideas remain those of Plain UC, adapted to Updated UC, our formulation of the UC Theorem relies on a definition of the UC operator that we introduce alongside the presentation of Updated UC. This definition allows the UC Theorem to be stated and proven independently of any specific implementation of the UC operator, thereby adding modularity and simplifying the overall analysis.

1.2 Related Work

Aside from the current version of Plain UC, widely used in this work, we can classify previously known publications into three categories:

- Generalizations of Plain UC, such as UC with joint state [CR03], UC with global setup [CDPW07], and multi-protocol UC [CDT19]. While the problems we have identified in Plain UC can be directly adapted to these generalizations, updating their UC theorems requires additional work. Although further research is needed, we believe that the updates to Plain UC offer a solid foundation for updating these

[1] A few minor changes are not directly related to the identified issues but were found necessary to ensure the correctness of the proof of the new UC Theorem. The fact that these changes are not implemented in Plain UC may itself be the source of additional issues for Updated UC; investigating this is left for future work.

generalizations. This confidence is partly justified by [BCH+20], which demonstrates that the UC theorem from Plain UC can be used to prove the UC theorem for UC with global setup.

- Alternative frameworks to Plain UC that rely on different computational models, such as iUC [CKKR19] (based on the IITM model [KTR20, RKC22]), CC [Mau12], and GNUC [HS15]. Both [HS15] and [KTR20] identify significant issues in older versions of Plain UC (specifically the 2005 and 2013 revisions), leading to counterexamples of the UC theorem. These issues are addressed by the current version of Plain UC that we analyze, and they arise from different ideas than the problems we present in this work. Appendix J of [RKC22] highlights minor issues distinct from those we identified, which can be resolved by slightly refining Plain UC 's protocol-execution definition; our description of Plain UC incorporates those refinements. Given that the problems of Plain UC identified in our work are new, it may be worthwhile to investigate if any of them can be adapted to these alternative frameworks. Finally, we emphasize that our aim is not to create a new framework, but to ensure the correctness of Plain UC by updating its definitions.
- Simplification of Plain UC, such as the one described in [CCL15], which presents a simplified variant within the context of standard multiparty computation. This variant deliberately restricts the expressiveness of Plain UC, for instance, by removing the possibility for protocols to have sub-parties [CCL15, (Section 3.2)]. Since its security is based on embedding this variant into Plain UC, we believe some of the issues we identified may also be relevant to this simplified version. Moreover, our updates to Plain UC pursue different goals: they aim to preserve as much expressiveness as possible.
- Comparison of UC with other frameworks, such as [RKC22] that compare the UC and IITM frameworks. It would be interesting to explore how the updated definitions of UC impact such comparisons.

Acknowledgements

This work was supported in part by the French ANR project SecNISQ (ANR-21-CE47-0014) and through the France 2030 program under the RESQUE project.

We express our deepest appreciation to Thomas Ricosset, whose insightful comments improved the presentation of the paper throughout its creation; Hugo Beguinet, whose feedback enhanced the flow of the main body; Erkan Tairi, for his constructive advice that refined the introduction and technical details; and Paula Celles, for her invaluable assistance in elevating the overall writing quality.

Outline of the paper

Section 2 gives a technical overview of the article.

The following sections elaborate on our contributions. First, Section 3 provides preliminary notations, followed by a description of Plain UC in Section 4. Then, Section 5 identifies various issues, limitations, and weaknesses within Plain UC. Section 6 introduces Updated UC, and Section 7 presents the UC Theorem associated with Updated UC.

The appendices include further contributions:

- Appendix A provides an overview of the appendix structure.
- Appendix B contains the missing details related to the issues, limitations, and weaknesses discussed in the main body.
- Appendix C presents a comprehensive proof of the UC Theorem of Updated UC.

- Appendix D discusses further issues, limitations, and weaknesses that, while not essential for understanding the main content, offer valuable insights. These include analyses of the notion of time complexity within Plain UC, the concept of a balanced environment, and the possibility of specifying the receiver of an external write via a predicate rather than by its extended identity.
- Appendix E defines a UC operator that addresses the problems associated with Plain UC's operator and aligns with our general definition of a UC operator (Definition 13).

Finally, the table of contents is provided at the end of the document.

2 Technical Overview

In this technical overview, we provide a detailed account of the main results of our article. Due to the technical nature of our contributions, some familiarity with Plain UC is required. To facilitate understanding, a comprehensive overview of Plain UC is provided in Section 4. Furthermore, from this point onward and throughout the rest of the article, we focus exclusively on Plain UC and the update proposed in this paper.

2.1 A Flaw in the Extended Session Definition Making the UC Theorem Unusable for Practical Protocols

In Plain UC, the extended session of a protocol π and session identifier s for a fixed execution is a set denoted by $\mathsf{Ext}(\pi, \mathsf{s})$.[2] It is constructed interactively during the execution by observing the ITIs that are invoked. All ITIs with code π and session identifier s, known as the main parties, are included and form the subset $\mathsf{MainParty}(\pi, \mathsf{s}) \subset \mathsf{Ext}(\pi, \mathsf{s})$. Additionally, ITIs that serve to perform computations for the main parties or other members of the extended session are also added. These are referred to as sub-parties and form the subset $\mathsf{ExtSub}(\pi, \mathsf{s}) \subset \mathsf{Ext}(\pi, \mathsf{s})$.

According to the precise definition of an extended session, an ITI N cannot be part of $\mathsf{ExtSub}(\pi, \mathsf{s})$ if it has not been invoked by a member of $\mathsf{Ext}(\pi, \mathsf{s})$ – see the discussion on page 17 in the overview of Plain UC.

In our work, we show that this condition is overly restrictive. In particular, we demonstrate that most ideal protocols are not subroutine respecting, and that the method described in [Can20, Section 5.2, Paragraph "Writing subroutine respecting protocols"] fails – contrary to its stated goal – to generically transform an arbitrary protocol into a subroutine respecting one.

To rectify this, we introduce the concept of "sleeping ITI" in Section 6.1. A sleeping ITI is one that has either not been invoked or has rejected all received messages thus far. In the earlier example, after being invoked by any ITI U that does not belong to $\mathsf{Ext}(\pi, s)$, N remains a sleeping ITI because it rejects U's message. Based on this, we revise the definition of an extended session in Section 6.1.

2.2 Protocols with Dynamic Sub-Parties: A Family of Protocols Unfit for Plain UC

Before delving into the details, we summarize our contribution in a single paragraph. We show that the standard notion of subroutine respecting is insufficiently restrictive, allowing the UC Theorem to be applied to a family of protocols that do not properly fit the UC framework. Specifically, Issue (I.2) (Section 5.1) utilizes one such protocol to construct a counterexample to the UC Theorem.

[2] Formally, the notation should indicate the specific execution being considered.

In simple terms, a protocol π is considered subroutine respecting if for any execution of any protocol – not necessarily π itself – and for any sid s, any communication between an ITI within $\mathsf{Ext}(\pi, \mathsf{s})$ and any other ITI must go through the main parties of $\mathsf{Ext}(\pi, \mathsf{s})$. In particular, for any execution[3]:

- Any input or output sent by an ITI $U \notin \mathsf{Ext}(\pi, \mathsf{s})$ to an ITI in $\mathsf{ExtSub}(\pi, \mathsf{s})$ is rejected.
- No ITI in $\mathsf{ExtSub}(\pi, \mathsf{s})$ sends an input or output to an ITI $U \notin \mathsf{Ext}(\pi, \mathsf{s})$.

This concept is of crucial importance for the UC Theorem because it allows us to "isolate" the members of $\mathsf{ExtSub}(\pi, \mathsf{s})$ from any execution, simulating any other ITI of the execution by an environment $\mathcal{E}_{\mathsf{sim}}$, as shown in Figure 1.[4]

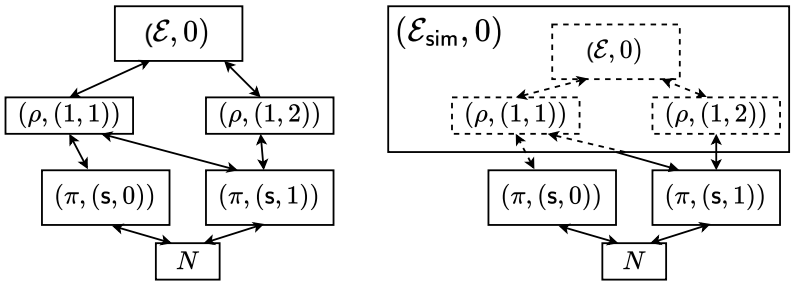


Figure 1: An example of "isolation": all ITIs are simulated except those within $\mathsf{Ext}(\pi, \mathsf{s})$

However, we discovered that the conditions required for being subroutine respecting are not sufficiently restrictive. This lack of constraint allows protocols that are not well-suited for analysis within the UC framework to satisfy the hypotheses of the UC Theorem, leading to counterexamples to the theorem.

We refer to these as protocols with "dynamic subparties". A protocol π has a dynamic sub-party N if there exists a session identifier s such that:

(a) In some executions, $N \in \mathsf{ExtSub}(\pi, \mathsf{s})$.

(b) In other executions, N is not sleeping (i.e., it exists and has not rejected all received messages) and does not belong to $\mathsf{ExtSub}(\pi, \mathsf{s})$.

While we believe that no real-world protocols have dynamic sub-parties, it is possible to construct simple protocols with this characteristic that lead to counterexamples of the UC Theorem. Such a protocol is constructed in Issue (I.2), as detailed in Section 5.1.

These protocols do not fit the UC framework because, during the execution of a protocol π with any adversary and environment, the execution rules prevent instances of condition (b) from occurring. These instances are only observable when considering the execution of another protocol ρ that uses π. As a result, the full behavior of π is unobservable when examining its UC-emulation by another protocol.

In order to exclude protocols with dynamic sub-parties from being considered subroutine respecting, we updated the notion of subroutine respecting protocols to develop the concept of sub-party respecting protocols (see Section 6.5) used in Updated UC. It uses the new concept of Panoptic Session, which is detailed in the next section.

[3]For simplicity, we do not describe all the conditions.

[4]Strictly speaking, achieving isolation requires additional concepts, such as compliance. However, the subroutine respecting notion remains the principal component.

2.3 Panoptic Session: A Static Variant of Extended Session

The concept of an extended session is dynamic because it is tied to a specific execution of the protocol. However, as discussed in the previous section on dynamic sub-parties, it might be beneficial to consider not only the *actual* sub-parties of an execution but also the *potential* sub-parties – that is, those that appear in some execution, without specifying which one.

To capture this idea, we introduce the notion of Panoptic Session (Section 6.3). For a protocol π and session identifier s, the panoptic session $\mathsf{Pan}(\pi, \mathsf{s})$ is defined as the union of all extended sessions $\mathsf{Ext}(\pi, \mathsf{s})$ across all executions of π with any environment and adversary. Therefore, the panoptic session is a static concept that provides information about the potential sub-parties of a protocol π.

For example, in the situation depicted in Figure 2, $\mathsf{Ext}(\pi, 0) = \{((\pi, (0,0)), N_i\}$ for each execution such that $(\pi, (0,0))$ sampled i, and $\{(\pi, (0,0))\} \cup \{N_i : i \in \{1, \ldots, 100\}\} \subset \mathsf{Pan}(\pi, 0)$.

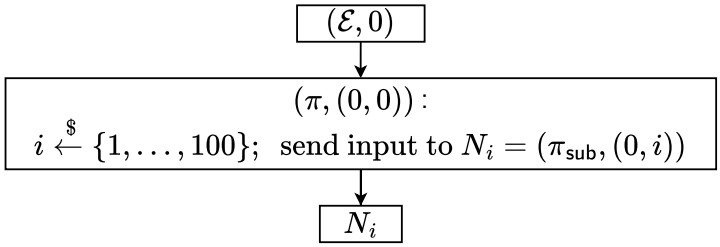


Figure 2: Example to illustrate the difference between Extended and Panoptic Sessions.

We assert that computing the panoptic sessions is straightforward for practical protocols. In fact, for any protocol analyzed under Plain UC, the panoptic sessions are already effectively determined, since checking the subroutine respecting condition requires computing the extended sessions that comprise it.

In the previous section, we explained that panoptic sessions are used to update the notion of subroutine respecting. In the upcoming section, we will demonstrate their applicability in addressing a flaw related to the subroutine exposing notion.

2.4 Problems with the Subroutine Exposing Property

In this section, we identify two flaws in the subroutine-exposing property – a requirement of the UC Theorem – and explain how they are addressed by the sid-exposing property, which updates it in Updated UC.

For a protocol α, the subroutine exposing property requires that for each sid s, one of the ITIs in the extended session $\mathsf{Ext}(\alpha, \mathsf{s})$ can inform – upon request – the adversary whether an ITI N belongs to the extended session $\mathsf{Ext}(\alpha, \mathsf{s})$ at the time of the request. We refer to this ITI as the exposing ITI of the extended session $\mathsf{Ext}(\alpha, \mathsf{s})$, or simply the exposing ITI when the context is clear.[5] It is also required that the extended identity of the exposing ITI for the extended session $\mathsf{Ext}(\alpha, \mathsf{s})$ can be determined using only α and s.

2.4.1 A Flawed Generic Transformation for Adding the Subroutine Exposing Property.

There is a straightforward mechanism, described in [Can20, Section 6.1], that transforms any protocol ϕ into a subroutine-exposing protocol ϕ_{SE}. To our knowledge, this is the only

[5] Note that the exposing ITI is sometimes called a "directory ITI" in Plain UC.

generic method for adding the subroutine-exposing property.

In summary, ϕ_{SE} introduces in any extended session $\mathsf{Ext}(\phi_{\mathsf{SE}}, \mathsf{s})$ a middle-man $M_{\phi,\mathsf{s}}$ which is aware of all ITIs belonging to the extended session $\mathsf{Ext}(\phi_{\mathsf{SE}}, \mathsf{s})$ and can therefore act as an exposing ITI, telling the adversary whether any ITI N belongs to $\mathsf{Ext}(\phi_{\mathsf{SE}}, \mathsf{s})$.

To the best of our knowledge, most protocols in the UC framework are not subroutine exposing. We substantiate this claim in Appendix B.6 by presenting numerous examples of ideal protocols that lack this property, including those discussed in [Can20, Section 6.1]. Consequently, applying this transformation is essential to endow them with the property and ensure they benefit from the UC Theorem's security guarantees – although its use is almost always implicit.

For the transformation to be sound, it must preserve UC-emulation: whenever π UC-emulates ϕ, it should follow that π_{SE} UC-emulates ϕ_{SE}. However, as shown by Limitation (L.2) (Section 5.3), this preservation is not guaranteed. While [Can20, Section 6.1] does not claim this property, we are not aware of prior works that explicitly analyze why it fails.

More importantly, because most protocols in the literature are not subroutine-exposing and no other generic method exists to achieve this property, such protocols cannot formally benefit from the composable security guarantees offered by the UC Theorem. To ensure they do, the preservation of UC-emulation must be analyzed on a case-by-case basis for each protocol[6]

We illustrate the problem with a simple example depicted in Figure 3. In this example, ϕ^* performs cryptographic operations without utilizing any subroutines, whereas the protocol π^* executes the same operations using two specialized subroutines. These subroutines are selected based on the type of input – for example, α might use an algorithm optimized for inputs of type 1, while β is more efficient for inputs of type 2.

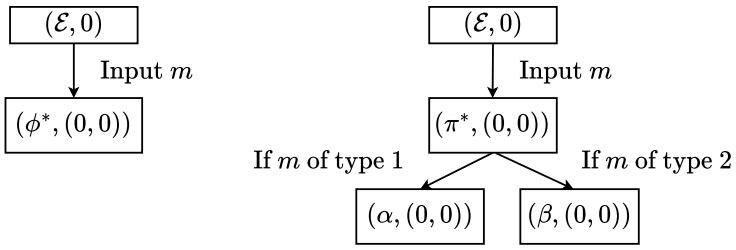


Figure 3: Protocols ϕ^* and π^* used to describe a flaw in the transformation $\mu \to \mu_{\mathsf{SE}}$.

By definition of π^*, $(\alpha, (0,0))$ (*resp.* $(\beta, (0,0))$) is part of the extended session $\mathsf{Ext}(\pi, 0)$ if and only if the environment sent to $(\pi, (0,0))$ an input of type 1 (*resp.* of type 2). This implies that an adversary against π^*_{SE} can leverage the exposing ITI to determine whether the environment sent an input of type 1 or 2. In contrast, an adversary against ϕ^*_{SE} does not gain additional information by using the exposing ITI.

This asymmetry in the information provided by the transformation illustrates why one cannot conclude that π^*_{SE} UC-emulates ϕ^*_{SE} simply because π^* UC-emulates ϕ^*.

To resolve this problem, we leverage the Panoptic Session concept to refine the subroutine exposing property, yielding the sid-exposing property of Updated UC (Section 6.4). Under this notion, the exposing ITI can only determine whether an ITI is in $\mathsf{Pan}(\pi, 0)$ rather than in $\mathsf{Ext}(\pi, 0)$. Informally, the "updated" exposing ITI – its equivalent in the sid-exposing property – reveals if a subroutine *could* belong to $\mathsf{Ext}(\pi, 0)$, whereas the "original" exposing ITI indicates if it *actually* belongs to $\mathsf{Ext}(\pi, 0)$. In particular, the the

[6]We do not claim that such an analysis is complex, but rather that it is required – and generally absent from the literature.

updated exposing ITI provides only static information: it does not need to interact with the ITIs of the current execution.[7] Consequently, the updated exposing ITI do not need to be part of the protocol : it can be defined simply as a Turing machine, which the adversary may simulate if desired. This is illustrated by Figure 4.

Applying this updated definition to the example in Figure 3, we see that it is sid-exposing, with no need for any transformation. The updated exposing ITI cannot help distinguish the type of input received from the environment: it answers affirmatively for both $(\alpha, (0,0))$ and $(\beta, (0,0))$, since both belong to $\mathsf{Pan}(\pi, 0)$. Hence, our sid-exposing property enables a broader class of protocols to be analyzed within the UC framework.

More generally, all the protocols presented in Appendix B.6 are already sid-exposing, without any modification required, and we believe this also holds for most protocols analyzed within the UC framework in the literature.

As a result, our new notion eliminates the need for a transformation such as $\pi \mapsto \pi_{\mathsf{SE}}$, along with its associated limitations.

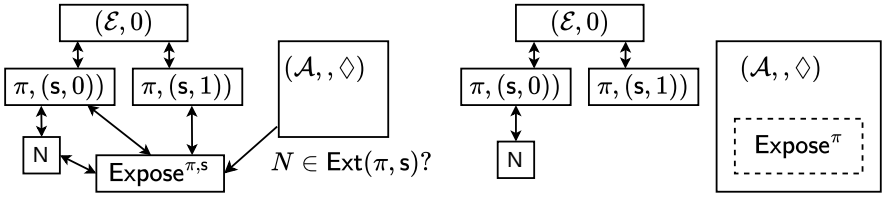


Figure 4: On the left-hand side, an execution illustrates the exposing ITI of the subroutine exposing property of Plain UC. On the right-hand side, an execution shows how an adversary can simulate the exposing ITI of the sid-exposing property of Updated UC.

2.4.2 A Problematic Dependence on the Session Identifier.

As stated in the definition of the subroutine exposing property, the exposing ITI for an extended session $\mathsf{Ext}(\pi, \mathsf{s})$ requires knowledge of both π and s to be used.

However, Weakness (W.2) (Section 5.3) shows that this requirement compromises the soundness of the UC Theorem's proof. Specifically, one core construction of the demonstration implicitly assumes knowledge of some sids without any guarantee of such knowledge. Moreover, attempting to guess these sids can result in an exponential time complexity.

The sid-exposing notion in Updated UC (Section 6.4) addresses this by making the exposing ITI depend only on the protocol π. More precisely, when given an ITI N[8], it returns the sid s of the panoptic session $\mathsf{Pan}(\pi, \mathsf{s})$ containing N, if such a sid exists.[9]

2.5 Extending Security Definitions Beyond Protocol Executions

In Plain UC, subroutine-respecting and compliance properties require that certain conditions apply to every protocol execution. However, the UC Theorem's proof for Plain UC studies some executions of extended systems that are not protocol executions – recall that a protocol execution is just a special case of an extended-system execution. We call these *replacement executions*.

[7] In contrast, the exposing ITI of the subroutine exposing notion provides dynamic information tied to the current execution.

[8] Formally, the input is an extended identity.

[9] We ensure uniqueness by requiring $\mathsf{Pan}(\pi, \mathsf{s}) \cap \mathsf{Pan}(\pi, \mathsf{s}') = \emptyset$ for $\mathsf{s} \neq \mathsf{s}'$, this is the separation property (Definition 7).

Informally, a replacement execution of a protocol ρ is one in which some subroutines with code ϕ are substituted by other subroutines with code π. Figure 5 illustrates this: the left-hand side shows a normal execution, while the right-hand side depicts a replacement execution.

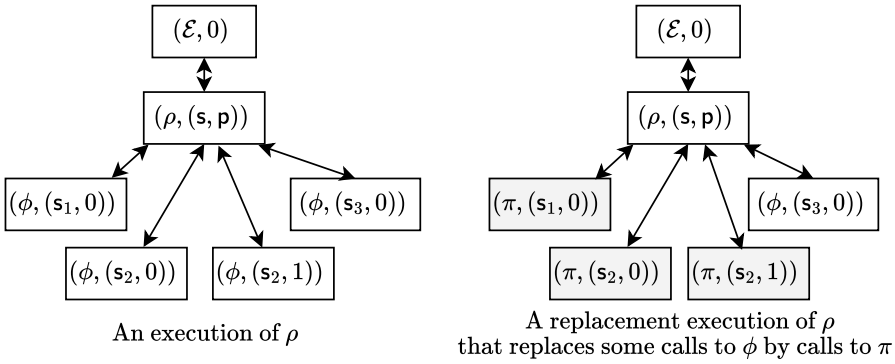


Figure 5: Illustration of a replacement execution.

In this work, we demonstrate that the conditions required by the subroutine respecting and compliance notions should apply not only to protocol executions but also to certain replacement executions.

In particular, Issue (I.3) (Section 5.1) exploits this omission in the subroutine respecting notion to construct a counterexample to the UC Theorem of Plain UC.[10]

In Updated UC, we update both the subroutine respecting and compliance notions to enforce their conditions on these additional executions as well. To that end, we formally define the notion of replacement execution in Section 6.2.[11]

2.6 Identifying Concerns in the UC Operator and Reimagining It for Enhanced Modularity

As explained in the introduction, the UC Theorem requires the construction of a protocol $\rho^{\phi \to \pi}$ for protocols ρ, ϕ, and π. Simply stated, $\rho^{\phi \to \pi}$ is a modification of ρ that replaces any call[12] to ϕ by a call to π. We refer to the function that, given ρ, ϕ, π, returns $\rho^{\phi \to \pi}$ as the UC Operator.

While the purpose of the UC Operator can be simply described, its detailed construction [Can20, (Section 6.1)] is intricate, demanding careful attention to various challenges. This intricacy is the origin of two main contributions of our article. First, we identified a subtle issue in its conception, leading to a complete counterexample of the UC Theorem (Issue (I.4) in Section 5.2). Second, we found a problem in how the UC Operator is used within the proof of the UC Theorem (Weakness (W.1) in Section 5.2).

To mitigate these two problems, it would be sufficient to create a new UC Operator and revisit the proof of the UC Theorem. However, we believe these issues illustrate that the UC Theorem should not directly depend on any specific UC Operator construction. Instead, a more modular formulation is necessary.

[10] Regarding the compliance, the corresponding problem we found is treated as a weakness (Weakness (W.4) in Section 5.4), since we have not attempted to build a counterexample based on it.

[11] A related notion already appears in the proof of the UC Theorem [Can20, (Theorem 22)] within Plain UC; however, it was used only within that proof and not in the definition of any protocol property, such as subroutine respecting or compliance.

[12] Here, we simplify the explanation by omitting the concept of top-level sessions [Can20, (Section 6.1)], which is not essential for understanding and will not be required for the rest of the document. More formally, we should speak of any call to a top-level main party of π.

To achieve this, we introduce in Section 6.7 a general definition of the UC Operator. Our definition outlines the necessary properties for a UC Operator, regardless of how it is precisely constructed. This enables us to provide a formulation of the UC Theorem that depends solely on this abstract definition (Theorem 1). Any operator that satisfies this definition can then be utilized, resulting in a "concrete" version of the UC Theorem using this operator (Corollary 1).

This approach enhances modularity, offering several advantages. The proof of the UC Theorem becomes simpler because it does not need to address the construction nuances of a particular UC Operator, effectively addressing Weakness (W.1). Moreover, the correctness of the theorem remains unaffected by any potential errors in an individual UC Operator.

As a supplementary contribution, we provide in Appendix E the construction of a UC Operator that conforms to our definition. Notably, our construction is unaffected by Issue (I.4).

2.7 Revealing Hidden Scope Constraints in the UC Theorem

The UC Theorem states that for protocols ϕ, π, and ρ, under specific hypotheses, if π UC-emulates ϕ, then $\rho^{\phi\to\pi}$ UC-emulates ρ. Recall that $\rho^{\phi\to\pi}$ is a modification of ρ where instances of ϕ are replaced by instances of π.

In this section, we explain that the UC Theorem does not impose sufficient restrictions on which protocols ρ can be used.

Indeed, while some restrictions apply to ρ, it is possible for ρ to invoke ITIs with both codes ϕ and π. In $\rho^{\phi\to\pi}$, ITIs with code ϕ are replaced with π, whereas ITIs with code π remain unchanged. Consequently, ITIs N with code π in $\rho^{\phi\to\pi}$ can be classified into two types:

- Type 1: N was invoked due to a replacement $\phi \to \pi$,
- Type 2: N is not the result of such a replacement.

This is illustrated by Figure 6.

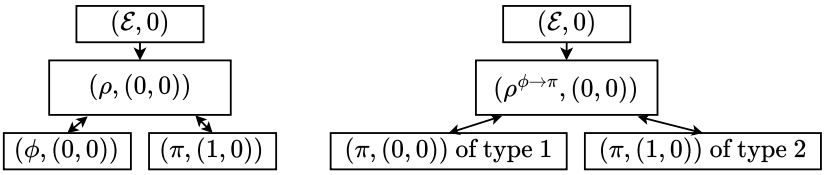


Figure 6: An illustration of the two types of ITIs with code π described in Section 2.7.

The existence of these two types of ITIs gives rise to a subtle weakness in the proof of the UC Theorem (Weakness (W.3), Section 5.4). Specifically, the proof requires one adversary to distinguish between Type 1 and Type 2 ITIs, yet it lacks the information to make this distinction.[13]

As a result, contrary to its stated claim, the UC Theorem applies only to protocols ρ whose executions exclude any Type 2 ITI.[14] This restriction still applies to Updated UC, and is captured by the update of the compliance notion (see Section 6.6).

[13] In specific cases – where additional information about ρ is available – this restriction could be relaxed to distinguish between Type 1 and Type 2. However, such information is not always available, which creates the problem.

[14] It also formally applies to protocols excluding any Type 1 ITI, but that case is vacuous – $\rho^{\phi\to\pi}$ would never replace ϕ with π.

2.8 Additional Contributions

As a side contribution, we discuss additional problems in Appendix D. These are listed in Table 3 and cover three concepts not addressed by the problems described in the main body.[15] These concepts are:

- The ability of a machine to *designate a message receiver using a predicate* rather than specifying its full code and identity [Can20, Section 3.1.2].
- The *time-complexity* notion used in Plain UC [Can20, Section 3.2].
- The concept of *balanced environment* [Can20, Section 4.2], proposed to prevent situations where the adversary is not allocated sufficient execution time.

Table 3: Additional issues, limitations, and weaknesses described in the article. They complete the main problems listed in Table 1.

Notions	Issues	Limitations	Weaknesses
Extended Session Subroutine Respecting		(L.3)	
UC Operator	(I.5) (I.6)		
Predicates to Designate Receiver	(I.7)	(L.4)	
Time Complexity			(W.6) (W.7)
Balanced Environment			(W.8)
Other			(W.5)

To address these additional problems, we implement the following changes in Updated UC:

- We remove the ability to use predicates to designate receivers. This simplifies our presentation, and we have not identified any protocols proven UC-secure that rely on them.
- Regarding the issues with the time complexity notion in Plain UC, which are not central to our article, we adopt a simpler notion, leaving a more general treatment for future work.
- Since we study a weakness related to the notion of balanced environments in Appendix D.6, and because time-complexity concerns are not central to our work, we do not provide a direct analogue of the balanced environment concept from Plain UC. However, our definition of a PPT adversary includes guarantees that ensure sufficient computation time.

3 Preliminary Notations

The set of binary words is denoted by $\{0,1\}^*$ and the empty word by ϵ. In both Plain UC and Updated UC, we need to represent pairs and additional symbols – such as $\Diamond$ or $\perp$ – on the tapes of Turing Machines, which can normally only contain binary words. We achieve this by implicitly using an encoding to represent them within $\{0,1\}^*$.

[15] Note that the category "other" used in the table refers to a weakness in the proof of the UC Theorem that is not directly related to any specific concept of UC. It addresses a deficiency in the definition of an adversary.

4 Overview of Plain UC

In this section, we recall some fundamental concepts of Plain UC, which will largely be utilized in Updated UC as well.

We made slight adjustments to the presentation, particularly in the definition of protocol execution. In addition, two more significant changes are made compared to Plain UC. First, as stated in the technical details (Section 2.8), we do not allow ITMs to designate recipients using predicates. Second, we use an alternative notion of time complexity along with a slightly modified notion of parametrized systems.

Because they are not necessary for understanding the problems of Plain UC discussed later, we omit the original time complexity definitions from the main body. However, they are recalled and discussed in Appendix D.5.

Finally, note that some notions – significantly revised in Updated UC – are not described here. Instead, they will be introduced in the sections that address the corresponding issues in Plain UC. These include concepts such as subroutine respecting, subroutine exposing, and compliance.

ITM, External Write, Protocol. The machines evoked in introduction are Interactive Turing Machines (ITM) [Can20, (Definition 4)]. They are special Turing machines that can send messages to each other through an operation called *external-write*, described below. We generally denote the code of an ITM using lowercase Greek letters.

An ITM μ contains multiples tapes, including:

- A read-only *identity tape* containing its code μ and an identity $\mathsf{id} \in \{0,1\}^*$. This pair (μ, id) is known as its *extended identity*.
- An *activation tape* containing 0 or 1, used during execution to designate which ITM can perform computations – only one ITM at a time.
- A read-only, read-once *randomness tape*, initialized with random bits.
- An *outgoing message tape* used to (1) send messages via the external-write operation, and (2) indicate the execution result (see the paragraph "Extended System and Execution of an Extended System" page 15).
- Three externally-writable tapes (*i.e.* writable only by other ITMs through the external-write operation) for receiving messages sent by other ITMs, with one tape for each type of message: an *input*, a *subroutine-output* (often abbreviated as *output*), and a *backdoor* tape.

Before executing an external-write instruction, an ITM must specify all information regarding the message to be sent. This is achievied by writing a tuple $(\mathsf{forced\text{-}write}, \mathsf{extid}_{\mathsf{rec}}, \mathsf{tape}, \mathsf{reveal\text{-}ext\text{-}id}, \mathsf{extid}_{\mathsf{send}}, \mathsf{m})$ on the outgoing message tape, where:

- $\mathsf{extid}_{\mathsf{rec}}$ is the extended identity of the recipient.
- $\mathsf{tape} \in \{\mathsf{input}, \mathsf{subroutine\text{-}output}, \mathsf{backdoor}\}$ is the destination tape.
- $\mathsf{m} \in \{0,1\}^*$ is the message to be sent, commonly referred to as input, output, or backdoor, depending on the chosen destination tape.
- $\mathsf{reveal\text{-}ext\text{-}id} \in \{0,1\}$ is a flag indicating whether the sender's external identity is sent with m.
- $\mathsf{forced\text{-}write} \in \{0,1\}$ is a flag indicating whether the recipient ITI should be created if it does not exist when the external-write is applied.

We denote by $\mathsf{SetExtWrites}$ the set of such tuples. The complete behavior of this instruction is described in paragraph "Extended System and Execution of an Extended System", page 15

We often use the term protocol to designate an ITM or a parametrized system (defined in paragraph "Parametrized Systems and Execution of a Parametrized System" page 16).

Configuration, ITI. A *configuration* [Can20, (Definition 5)] of an ITM consists of its current tape, the contents of all its tapes, and the location of the head on each tape. A configuration is active if the activation tape is set to 1, otherwise it is inactive.

An instance [Can20, (Definition 5)] of an ITM (ITI) μ consists of the content of its identity tape, whose is its extended identity $M = (\mu, \mathsf{id})$. By abusing notation, we often speak of "ITI" instead of "extended identity". For example, if an algorithm has an input which is an extended identity, we say that its input is an ITI. We generally denote ITIs using uppercase Latin letters.

Extended System and Execution of an Extended System. An extended system of ITMs [Can20, (section 3.1.2)] is a pair $(\mu, \mathcal{C})$, where μ is an ITM called the initial ITM and $\mathcal{C} : \{0,1\}^* \to \{\mathsf{allow}, \mathsf{disallow}\} \cup \mathsf{SetExtWrites}$ is a function called the control function, which role is described below.

To execute an extended system $(\mu, \mathcal{C})$ with an input $z \in \{0,1\}^*$, the ITI $(\mu, 0)$, called the initial ITI, is first activated – *i.e.* 1 is written on its activation tape –, with z written on its input tape. During execution, there is no distinct ITIs with same extended identity and only the activated ITI can perform operations. An ITI will be deactivated in favor of another ITI in two occasions: when it makes an external-write or when it deactivates itself. We describe these two situations below and conclude the paragraph by defining the end and result of an execution.

External-write. When activated, an ITI can send inputs, outputs, or backdoors to another ITI via the external-write operation. If the recipient ITI does not exist and forced-write is set, the recipient ITI is created in its initial state and receives the message in the corresponding input, output or backdoor tape. If the recipient ITI does not exist and forced-write is unset, the initial ITI $(\mu, 0)$ is instead activated, with no message. The value of flag reveal-ext-id is written with the message and, if it is set, the extended identity of the sender is also written.[16] The effect of an external-write can be modified by the control function $\mathcal{C}$. When an external-write operation is made, $\mathcal{C}$ receives the history of all consecutive configurations of each activated ITIs and can decide to allow, disallow or replace the external-write operation with another. If the control function allows the external-write, nothing is changed. If it is disallowed, initial ITI $(\mu, 0)$ is activated next, with no message. If it is replaced by another external-write operation, the new external-write operation is applied.

Self-Deactivation, Reject and Halt. An activated ITI can deactivate itself. When this is done, the next activated ITI is the initial ITI $(\mu, 0)$, with no message. A *reject* [Can20, (Section 3.1.2)] is a specific case of deactivation, where the state of the ITI as well as all its non read-once tables are returned in their state prior to the last activation. Moreover, each ITI can enter in *halt* state. Upon entering the halt state, an ITI deactivates and will reject any subsequent messages.

End of Execution. The execution ends when the initial ITI halts. The result of the execution is the first bit of the outgoing message tape. We sometimes say "the environment returns b" to indicate the termination and result of the execution.

Extended Execution. By an *extended execution*, we mean the execution of an extended system.

Execution Prefix. An execution prefix of an extended execution is a complete description of the extended execution from its start up to a given point in time.

Protocol Execution. The most important example of extended execution is the protocol execution [Can20, (Section 4.1)].

Let $\alpha, \mathcal{A}, \mathcal{E}$ three ITMs. We will refer to any ITI different of $(\mathcal{E}, 0)$ and $(\mathcal{A}, \Diamond)$ as a

[16] The various elements written with the message are encoded in a way that ensures they are clearly separated. For example, it is not possible to perform an external-write with reveal-ext-id unset such that the recipient would interpret it as one with reveal-ext-id set.

protocol ITI.[17] The execution of the protocol α with adversary $\mathcal{A}$ and environment $\mathcal{E}$ is the execution of the extended system $(\mathcal{E}, \mathcal{C}_{\mathsf{EXEC}}^{\alpha,\mathcal{A},\mathcal{E}})$, the control function ensuring some restrictions:

- Unless explicitly stated otherwise, the identity of any recipient of an external-write operation must be structured as a pair $(\mathsf{s}, \mathsf{p}) \in \{0,1\}^* \times \{0,1\}^*$ where s is called the *session identifier* (sid) and p the *party identifier* (pid).
- $(\mathcal{E}, 0)$ always set forced-write and can only send inputs. When $(\mathcal{E}, 0)$ sends input to an ITI with identity $\Diamond$, the recipient ITI is replaced by $(\mathcal{A}, \Diamond)$. When $(\mathcal{E}, 0)$ sends an input to an ITI with identity distinct of $\Diamond$, the recipient's code is replaced by α. When $\mathcal{E}$ sends an input to an ITI with identity distinct of $\Diamond$, the identity must be structured as a couple (s, p) and it must consistently use the same sid s, usually denoted as $\mathsf{s}_{\mathsf{test}}$ and called the test sid.
  Finally, we require that $(\mathcal{E}, 0)$ always sets reveal-ext-id, but it has impersonation power: in all its external-write operations, $\mathcal{E}$ can use a sender ITI other than $(\mathcal{E}, 0)$.[18]
- $(\mathcal{A}, \Diamond)$ always unsets forced-write, reveal-ext-id and can only send backdoors.[19] If it sends a backdoor to an ITI with identity 0, the recipient is replaced by $(\mathcal{E}, 0)$.
- Protocol ITIs always set reveal-ext-id and forced-write.[20] [21] When a protocol ITI sends a backdoor, the recipient ITI is replaced by $(\mathcal{A}, \Diamond)$. When a protocol ITI with code α and sid $\mathsf{s}_{\mathsf{test}}$ (the test sid) sends an output to an ITI M that (1) does not have code α and sid $\mathsf{s}_{\mathsf{test}}$, and (2) does not exist in the system, the message is written to the output tape of $(\mathcal{E}, 0)$, along with the target extended identity and the sender's identity – but not the sender's code.[22]

Protocol Execution Prefix. A α-execution prefix is a prefix of the execution of α. A protocol execution prefix is a α-execution prefix for some protocol α.

Parametrized Systems and Execution of a Parametrized System. As mentioned at the beginning of this section, the definition of parametrized system we use is slightly different from that of Plain UC [Can20, (Section 3.2)].

The UC-emulation property is asymptotic, both in terms of the size of messages and with respect to a security parameter $k \in \mathbb{N}$. The notion of parametrized system is a generalization of the notion of an ITM that takes into account this security parameter. Essentially, it is a sequence of ITMs indexed by the security parameter.

More formally, a parametrized system is a sequence of ITMs $\boldsymbol{\phi} = (\phi_k)_{k\in\mathbb{N}}$ such that the code of each ϕ_k can be written in polynomial time in k. More precisely, there exists a Turing machine and a polynomial P such that, given k, it returns the code ϕ_k in fewer than $P(k)$ operations. Parametrized systems will generally be written using bold lowercase Greek letters.

The execution of a parametrized system $\boldsymbol{\phi}$ is defined as the execution of a protocol, with slight differences:

- All the inputs to $(\mathcal{E}, 0)$ must be pairs (k, z), for $k \in \mathbb{N}$ and $z \in \{0,1\}^*$. k is called the security parameter of the execution.

[17] Note that in plain UC, the identity of adversary is normally $(\mathsf{s}_{\mathsf{test}}, \Diamond)$ and not $\Diamond$. This is a cosmetic change.

[18] This simplifies the original formulation where the environment could set or unset this flag. It is essentially equivalent because the impersonation power allows the environment to conceal its extended identity.

[19] [RKC22, Appendix J.1] explains why reveal-ext-id needs to be unset.

[20] In the original formulation, forced-write could be unset for inputs and outputs. However, since we removed the ability to use predicates to designate the receiver, we do not consider this option relevant.

[21] In the original formulation, forced-write needs to be unset for backdoor and thus the adversary cannot be invoked by a protocol ITI. This might be because executions are made with balanced environments [Can20, Section 4.2], where the adversary is invoked before any protocol sends a backdoor. As we do not restrict to such environment, we take in account the possibility for a protocol ITI to invoke the adversary.

[22] Note that in this situation, the sender identity takes the form $(\mathsf{s}_{\mathsf{test}}, \mathsf{p})$ for some $\mathsf{p} \in \{0,1\}^*$.

- The adversary has a new read-only tape containing the security parameter of the execution.[23]
- When the security parameter of the execution is k, the rules of execution of ϕ_k are applied.

We denote by $\mathcal{C}_{\mathsf{EXEC}}^{\boldsymbol{\phi},\mathcal{A},\mathcal{E}}$ the control function and by $\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{A},\mathcal{E}}(k,z)$ the binary random-variable representing the result of the execution of $\boldsymbol{\phi}$ with adversary $\mathcal{A}$ and environment $\mathcal{E}$ on input (k,z).

Subroutine, Extended Session. Let us consider M and M' two ITIs in an execution prefix. M' is a *subroutine* [Can20, (Section 3.1.2)] of M if M has passed input to M' and M' has not rejected it, or if M' has passed output to M and M has not rejected it.

The *extended session* [Can20, (Section 3.1.2)] of a protocol can be viewed as the set of ITIs that are executed by an instance of the protocol. More formally, considering μ an ITM, a sid $\mathsf{s} \in \{0,1\}^*$ and some prefix of execution θ of a protocol, the extended session $\mathsf{Ext}(\mu,\mathsf{s},\theta)$ is a set constructed inductively by inspecting the ITIs by order of invocation: each newly invoked ITI M is added to the extended session if (a) M has code μ and sid s (such an ITI is called a main party); or (b) M is a subroutine of an ITI that is already in the extended session; or (c) M is invoked by an ITI that is a member of the extended session, but not a main party. Note that determining whether an ITI belongs to an extended session only requires examining the first message received by the ITI. This is because ITIs are processed in the "order of invocation" and that we focus on "newly invoked ITI". Therefore, it is equivalent to replace item (b) by: "it is invoked with an input by a main party and does not reject the input".

The ITIs with code μ and sid s are called the main parties of $\mathsf{Ext}(\mu,\mathsf{s},\theta)$ and form a set $\mathsf{MainParty}(\mu,\mathsf{s})$. The members of the extended session that are not main parties are called *sub-parties* and form a set $\mathsf{ExtSub}(\mu,\mathsf{s},\theta)$.

We will often omit the prefix θ from $\mathsf{Ext}(\mu,\mathsf{s},\theta)$ when clear from context. In the context of an execution, we speak of an extended session of μ to means any extended session $\mathsf{Ext}(\mu,\mathsf{s})$ for some sid $\mathsf{s} \in \{0,1\}^*$.

Finally, note that in the execution of a protocol μ with test sid $\mathsf{s}_{\mathsf{test}}$, the extended session $\mathsf{Ext}(\mu,\mathsf{s}_{\mathsf{test}})$ is referred to as the extended test session.

Probabilistic Polynomial Time ITM. We introduce here the notions of PPT (Probabilistic Polynomial Time) adversary, environment, and protocol – more formally defined in Appendix C.2.

These definitions differ from the notion of PPT ITM given in Plain UC [Can20, (Section 3.2)]. They are partially inspired by the concept of a polynomial time machine from [CCL15, (Section 2.1)] and focused on the context of parametrized system execution.

An adversary is PPT if there exists a polynomial P such that at any moment in any possible execution, its time of execution is bounded by $\sum_i P(|w_i|+k)$ where $(w_i)_i$ is the sequence of external-writes it received and k the security parameter written on its dedicated tape. Thus, the adversary can always perform at least $n_M P(k)$ operations, where n_M is the number of messages it received.

An environment is PPT if there exists a polynomial P such that at any moment in any possible execution, its execution time is bounded by $P(k+|z|)$, where $(k,z) \in \mathbb{N} \cup \{0,1\}^*$ is its input, given at its first activation.

A protocol $\boldsymbol{\alpha}$ is PPT if for any PPT environment $\mathcal{E}$ and PPT adversary $\mathcal{A}$, there exists a polynomial R such that any execution of $\boldsymbol{\alpha}$ with environment $\mathcal{E}$, adversary $\mathcal{A}$ and input $(k,z) \in \mathbb{N} \cup \{0,1\}^*$ has an execution time bounded by $R(k+|z|)$.

UC-emulation. The definition of UC-emulation relation we give is similar to that in Plain UC, but it incorporates our variants of parametrized system and PPT notions for

[23] This differs from the usual formulation of Plain UC, and we believe this variant is more practical.

adversary and environment. Additionally, we do not consider the notion of balanced environment [Can20, (Section 4.2)].[24]

To be defined, the UC-emulation relation requires three preliminary notions.

Let ξ a predicate on extended identities. An environment is said ξ-*identity-bounded* [Can20, (Section 4.2)] if it only impersonates extended identities M for which $\xi(M)$ is true.

A binary probability distribution ensemble [Can20, (Section 4.2)] is an infinite set $X = \{X(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ of binary probability distributions, indexed by $\mathbb{N}\times\{0,1\}^*$.

Two binary probability distribution ensembles X and Y are *indistinguishable* [Can20, (Definition 8)] – written $X \approx Y$ – if for any $c,d\in\mathbb{N}$, there exists $k_0\in\mathbb{N}$ such that for all $k > k_0$ and all $z\in\bigcup_{\kappa\leq k^d}\{0,1\}^\kappa$, we have: $|\Pr[X(k,z)=1]-\Pr[Y(k,z)=1]| < k^{-c}$. They are said to be *perfectly indistinguishable* if we always have $\Pr[X(k,z)=1]=\Pr[Y(k,z)=1]$. The execution of a parametrized system $\boldsymbol{\rho}$ naturally gives a binary probability distribution ensemble $(\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{A},\mathcal{E}}(k,z))_{k\in\mathbb{N},z\in\{0,1\}^*}$ that is used in the definition of UC-emulation.

Definition 1 (UC-emulation [Can20, (Definition 9)])**.** We consider $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ two PPT parametrized systems and ξ a predicate on extended identities. We say that $\boldsymbol{\pi}$ UC-emulates $\boldsymbol{\phi}$ with respect to ξ-identity-bounded environments (or, $\boldsymbol{\pi}$ ξ-UC-emulates $\boldsymbol{\phi}$) if for any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{S}$ such that for any PPT, ξ-identity-bounded environment $\mathcal{E}$, we have $\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{A},\mathcal{E}}\approx\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S},\mathcal{E}}$. If ξ is the predicate which is true for all ITIs, then $\boldsymbol{\pi}$ is said to UC-emulate $\boldsymbol{\phi}$. The UC-emulation is said to be perfect if the two probability distributions ensembles are perfectly indistinguishable. If this relation is only verified for a fixed adversary $\mathcal{A}$, we say that $\boldsymbol{\pi}$ ξ-UC-emulates $\boldsymbol{\phi}$ with respect to $\mathcal{A}$.

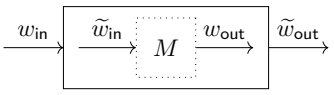
In the description of our counter-examples, we will consider UC-emulation for ITMs π and ϕ instead of parametrized systems. When we state that π UC-emulates ϕ, it should be understood as $\boldsymbol{\pi}$ UC-emulates $\boldsymbol{\phi}$ for $\pi_k=\pi$ and $\phi_k=\phi$ for each $k\in\mathbb{N}$.

Dummy Adversary. In plain UC, an adversary $\mathcal{D}$, called the dummy adversary [Can20, Section 4.3.1], is of special importance as it can be shown that UC-emulation with respect to $\mathcal{D}$ implies UC-emulation [Can20, (Claim 11)] (*i.e.* UC-emulation with respect to any PPT adversary).

Specifically, the dummy adversary proceeds as follows. When activated with an input "$(\mathsf{m},(\mu,\mathsf{id}))$" from the environment $(\mathcal{E},0)$, where m is a message, and (μ,id) an extended identity, $(\mathcal{D},\Diamond)$ sends backdoor "m" to ITI (μ,id). When activated with a backdoor "m" sent by an ITI (μ,id), $(\mathcal{D},\Diamond)$ sends backdoor "$(\mathsf{m},(\mu,\mathsf{id}))$" to the environment $(\mathcal{E},0)$.

$\mathcal{D}$ is a PPT adversary, as its execution time for each activation is linear in the size of the received message.

Universal Composition Operator (UC Operator). The UC operator $\rho^{\phi\to\pi}$ is defined in [Can20, (Section 6.1)] and relies on the notion of *structured protocol* [Can20, (Section 5.1)]. A structured protocol is an ITM whose functioning can be divided into two parts: a body, which is an ITI, and a shell that simulates the body and can modify the external-writes it receives or intends to send. In the following figure, a structured ITI N is shown with a body M. We observe that N can modify both the external-writes that M receives and those that M intends to send. The shell is responsible for simulating the body and applying these modifications.



In a nutshell, the ITIs of an extended session $\mathsf{Ext}(\rho^{\phi\to\pi},\mathsf{s})$ are structured ITIs whose bodies correspond to ITIs from the extended session $\mathsf{Ext}(\rho,\mathsf{s})$ or from extended sessions of π. The shells of these ITIs are used to replace codes in certain sent and received

[24] This is justified by a weakness described in Appendix D.6.

external-write operations, in such a way that the execution of $\rho^{\phi\to\pi}$ corresponds to an execution of ρ in messages to sub-parties with code π are replaced by messages to ITIs with the same identity but code ϕ.

We now provide a more detailed explanation. To simplify it, we make the following three assumptions. First, no execution of ρ invokes a main party of π. Second, for any sids $\mathsf{s} \neq \mathsf{s}'$ and $\mu \in \{\phi, \pi\}$, $\mathsf{ExtSub}(\mu, \mathsf{s}) \cap \mathsf{ExtSub}(\mu, \mathsf{s}') = \emptyset$. Third, for any sids s, s', $\mathsf{Ext}(\phi, \mathsf{s}) \cap \mathsf{Ext}(\pi, \mathsf{s}') = \emptyset$.

To describe the rules of replacement, we call an ITI *outside* if it does not belong to any extended session of π. The rules are as follows:

- *Replacements $\phi \to \pi$:* Inputs or outputs from an outside ITI to a main party of an extended session $\mathsf{Ext}(\phi, \mathsf{s})$ are redirected to the main party of $\mathsf{Ext}(\pi, \mathsf{s})$ with the same pid.
- *Replacements $\pi \to \phi$:* When a main party of an extended session $\mathsf{Ext}(\pi, \mathsf{s})$ sends an output to an ITI M outside this extended session, the external-write is modified so that the sender code, originally π, is replaced by ϕ.

To apply these replacements, the shells store information through local variables and communicate with other shells. More precisely, any message sent by a structured protocol to another one can be seen as a couple $(\mathsf{m}_{\mathsf{body}}, \mathsf{m}_{\mathsf{shell}})$ where $\mathsf{m}_{\mathsf{body}}$ is aimed to the body of the recipient (we call it a body message) and $\mathsf{m}_{\mathsf{shell}}$ to its shell (we call it a shell message). The only aim of the shell messages is to communicate enough information to ensure good application of the replacements detailed above.

Finally, note that when applied to parametrized systems $\boldsymbol{\rho}, \boldsymbol{\phi}$ and $\boldsymbol{\pi}$, $\boldsymbol{\rho}^{\boldsymbol{\phi}\to\boldsymbol{\pi}}$ is itself a parametrized system with $\left(\rho^{\phi\to\pi}\right)_k = \rho_k^{\phi_k\to\pi_k}$.

UC Theorem. The UC Theorem [Can20, (Theorem 22)] states that, under certain hypotheses on PPT parametrized systems $\boldsymbol{\rho}$, $\boldsymbol{\phi}$, $\boldsymbol{\pi}$ and on a predicate ξ on extended identities, if $\boldsymbol{\pi}$ ξ-UC-emulates $\boldsymbol{\phi}$, then $\boldsymbol{\rho}^{\boldsymbol{\phi}\to\boldsymbol{\pi}}$ UC-emulates $\boldsymbol{\rho}$.

Concretely, they need to satisfy some conditions : (1) $\boldsymbol{\phi}$ and $\boldsymbol{\pi}$ are both subroutine respecting [Can20, (Definition 19)] and subroutine exposing [Can20, (Definition 21)], and (2) $\boldsymbol{\rho}$ is $(\boldsymbol{\pi}, \boldsymbol{\phi}, \xi)$-compliant [Can20, (Section 6.1)]. The subroutine respecting, subroutine exposing, and compliance notions are respectively linked to the problems of the UC Theorem described in sections 5.1, 5.3 and 5.4. They are detailed within those sections.

5 Problems Within The Universal Composition Theorem

This section provides a detailed account of the issues, limitations, and weaknesses in the UC theorem of Plain UC [Can20, (Theorem 22)] listed in Table 1.

While these points help in understanding the changes introduced by Updated UC, they are also of independent interest.

5.1 Problems Related to Extended Sessions and Subroutine Respecting Property

We begin by reviewing the concept of subroutine respecting protocols.

A protocol μ is said *subroutine respecting* [Can20, (Definition 19)] if, in any prefix of execution of any protocol α and for any sid $\mathsf{s} \in \{0,1\}^*$, certain restrictions apply to communications with the extended session $\mathsf{Ext}(\mu, \mathsf{s})$ – we emphasize that we are not restricted to executions of μ itself, but rather consider how μ behaves within the execution of any protocol α. For the problems we are addressing, we will focus on these four restrictions: (1) the main parties of $\mathsf{Ext}(\mu, \mathsf{s})$ reject any output sent by an ITI outside $\mathsf{Ext}(\mu, \mathsf{s})$; (2) the sub-parties of $\mathsf{Ext}(\mu, \mathsf{s})$ reject any input or output sent by an ITI outside

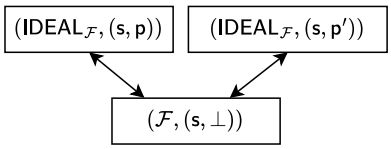


Figure 7: Illustration of an ideal protocol.

$\mathsf{Ext}(\mu, \mathsf{s})$; (3) no member of $\mathsf{Ext}(\mu, \mathsf{s})$ sends an input to an existing ITI that is not already a member of $\mathsf{Ext}(\mu, \mathsf{s})$; and (4) no sub-party of $\mathsf{Ext}(\mu, \mathsf{s})$ sends an output to an existing ITI that is not already a member of $\mathsf{Ext}(\mu, \mathsf{s})$. Moreover, neither of the required conditions impose any restriction on received or sent backdoors.

(L.1) Extended Session Definition Excludes Too Many ITIs. The notions of extended session and of subroutine respecting protocol are overly restrictive.

Looking at the definition of an extended session recalled in Section 4, page 17, we observe that an ITI which is neither a main party nor invoked by a member of the extended session will never become part of that session, even if it rejects the message that invoked it. We will show that this property implies that a broad class of protocols that should be subroutine respecting are, in fact, not subroutine respecting. This class consists of:

- most of the ideal protocols associated with ideal functionalities [Can20, Section 5.3];
- certain protocols constructed according to the method described in [Can20, Section 5.2, Paragraph "Writing subroutine respecting protocols"], whose stated goal was precisely to ensure that the resulting protocols are subroutine respecting.

We demonstrate this here for the case of ideal protocols; the second case is addressed in Appendix B.1.1. First, let's recall the necessary notions about the ideal protocols. For any ITM $\mathcal{F}$, we associate a protocol $\mathsf{IDEAL}_\mathcal{F}$, referred to as the ideal protocol of $\mathcal{F}$ (in this context, $\mathcal{F}$ is called an ideal functionality). By definition of the ideal protocol $\mathsf{IDEAL}_\mathcal{F}$ [Can20, Section 5.3], for any $\mathsf{s} \in \{0,1\}^*$, the ITIs that may appear in an extended session $(\mathsf{IDEAL}_\mathcal{F}, \mathsf{s})$ are of two types:

- The main parties of $(\mathsf{IDEAL}_\mathcal{F}, \mathsf{s})$, which act as relays between the environment and the ideal functionality $(\mathcal{F}, (\mathsf{s}, \bot))$. In particular, when a main party $(\mathsf{IDEAL}_\mathcal{F}, (\mathsf{s}, \mathsf{p}))$ receives an input "m" from an ITI M, they send the input "(m, M)" to $(\mathcal{F}, (\mathsf{s}, \bot))$ ([Can20, Item (1) of Section 5.3]).
- The ideal functionality ITI $(\mathcal{F}, (\mathsf{s}, \bot))$, and its sub-parties.

This is illustrated in Figure 7.

Finally, we make a mild assumption on $\mathcal{F}$, which is satisfied by most ideal functionalities studied in the literature: there exist a message $\tilde{\mathsf{m}} \in \{0,1\}^*$ and a session identifier $\tilde{\mathsf{s}} \in \{0,1\}^*$ such that $(\mathcal{F}, (\tilde{\mathsf{s}}, \bot))$ rejects when invoked with input message $\tilde{\mathsf{m}}$. Indeed, the inputs that such functionalities accept generally need to follow a specific format, which makes it easy to find such a message $\tilde{\mathsf{m}}$.

Proposition 1. $\mathsf{IDEAL}_\mathcal{F}$ *is not subroutine respecting.*

Proof. Consider an ITI $(\rho, (\tilde{\mathsf{s}}, 0)), \rho \notin \{\mathsf{IDEAL}_\mathcal{F}, \mathcal{F}\}$, which, upon its first activation, sends the input "$\tilde{\mathsf{m}}$" to $(\mathcal{F}, (\tilde{\mathsf{s}}, \bot))$, and upon its second activation, sends the input "0" to $(\mathsf{IDEAL}_\mathcal{F}, (\tilde{\mathsf{s}}, 0))$. Next, consider an environment $\mathcal{E}$ which, at its first activation, sends the input "0" to the protocol with identifier $(\tilde{\mathsf{s}}, 0)$, and at its second activation, again sends the input "0" to the protocol with identifier $(\tilde{\mathsf{s}}, 0)$. Let θ be a prefix of the execution of ρ with environment $\mathcal{E}$, some adversary, and some input.

We first note that $(\mathcal{F}, (\tilde{\mathsf{s}}, \bot)) \notin \mathsf{Ext}(\mathsf{IDEAL}_\mathcal{F}, \tilde{\mathsf{s}}, \theta)$. This follows directly from the definition of extended sessions, since $(\mathcal{F}, (\tilde{\mathsf{s}}, \bot))$ is invoked by an input from $(\rho, (\tilde{\mathsf{s}}, 0))$, which itself does not belong to $\mathsf{Ext}(\mathsf{IDEAL}_\mathcal{F}, \tilde{\mathsf{s}}, \theta)$.

We now use this observation to show that $\mathsf{IDEAL}_{\mathcal{F}}$ violates one of the conditions required for a protocol to be subroutine respecting.

Let us analyze the execution more closely. After being invoked by $(\rho, (\tilde{\mathsf{s}}, 0))$, $(\mathcal{F}, (\tilde{\mathsf{s}}, \perp))$ rejects, since it is invoked by an input with message $\tilde{\mathsf{m}}$ (by the assumption made on $\mathcal{F}$). We now describe the subsequent sequence of activations and messages:

- The environment $(\mathcal{E}, 0)$ is activated again, without receiving any message.
- Upon activation, the environment $(\mathcal{E}, 0)$ sends input "$\mathsf{0}$" to $(\rho, (\tilde{\mathsf{s}}, 0))$.
- In response, $(\rho, (\tilde{\mathsf{s}}, 0))$ invokes $(\mathsf{IDEAL}_{\mathcal{F}}, (\tilde{\mathsf{s}}, 0))$ with input "$\mathsf{0}$".
- Then, $(\mathsf{IDEAL}_{\mathcal{F}}, (\tilde{\mathsf{s}}, 0))$ sends input "$(\mathsf{0}, (\rho, (\tilde{\mathsf{s}}, 0)))$" to $(\mathcal{F}, (\tilde{\mathsf{s}}, \perp))$.

This final input demonstrates that $\mathsf{IDEAL}_{\mathcal{F}}$ is not subroutine respecting, since a main party of $\mathsf{Ext}(\mathsf{IDEAL}_{\mathcal{F}}, \tilde{\mathsf{s}}, \theta)$ sends an input to an existing ITI that does not belong to $\mathsf{Ext}(\mathsf{IDEAL}_{\mathcal{F}}, \tilde{\mathsf{s}}, , \theta)$, thereby violating condition (3) of the subroutine respecting property. □

To address this issue, Updated UC adopts an updated definition of the extended session (Section 6.1). The main change in this definition is that the fact that an ITI has rejected all messages it has received since its invocation no longer prevents it from later belonging to an extended session. As a result, Proposition 1 does not apply under this updated definition.

Note that this issue is not directly related to the other problems discussed in this section (Section 5.1). In particular, addressing it by adopting the updated definition of extended sessions from Updated UC does not resolve the other concerns identified in this section.

(I.1) Subroutine Respecting Property and Backdoors.

We recall that in the execution of a protocol, the adversary cannot invoke ITIs. Nevertheless, once an ITI has been invoked it becomes accessible to the adversary – even if it rejected all received messages.

Additionally, in the full definition of the subroutine respecting notion [Can20, Definition 19], no explicit condition is imposed on backdoor messages sent to main parties. The only message-related requirements in that definition concern inputs and outputs.

Together, these observations yield the following counterexample to the UC theorem.

We create an ITM π such that:

- For any s, p, $(\pi, (\mathsf{s}, \mathsf{p}))$ always rejects any output it receives.
- For any $(\mathsf{s}, \mathsf{p}) \neq (0, 0)$, $(\pi, (\mathsf{s}, \mathsf{p}))$ always rejects any message it receives (input, output or backdoor).
- At reception of its first input, $(\pi, (0, 0))$ initializes a variable $\mathsf{ReceivedInput}$, then deactivates and rejects any subsequent input, output or backdoor it receives.
- At reception of a backdoor message, if $\mathsf{ReceivedInput}$ is initialized, $(\pi, (\mathsf{s}, \mathsf{p}))$ rejects it (as already implied by the previous item). Else, if $\mathsf{ReceivedInput}$ is uninitialized, $(\pi, (\mathsf{s}, \mathsf{p}))$ sends an output to an ITI with code $\mu \neq \pi$.

It can be readily verified that π is subroutine respecting. Indeed, in the execution of any protocol,[25] every ITI with code π has no subroutines and rejects all output messages.

Moreover, in any execution of π with an adversary and an environment, all main parties of the extended test session are invoked through inputs originating from the environment. In particular, if $(\pi, (0, 0))$ exists in the execution, it must have initialized the variable $\mathsf{ReceivedInput}$ and therefore rejects any subsequent input, output, or backdoor message. As a consequence, one can show that π UC-emulates the protocol ϕ that rejects any incoming message.

However, we can straightforwardly construct a protocol ρ such that $\rho^{\phi \to \pi}$ does not UC-emulate ρ. This can be achieved by exploiting the fact that ρ may invoke $(\phi, (0, 0))$

[25] Recall that the conditions ensuring that a protocol π is subroutine respecting must hold not only in executions of π itself, but also in executions of any protocol ρ in which π may appear as a sub-party.

through an output message, together with the behavior of $(\pi, (0,0))$ upon receiving a backdoor message when $\mathsf{ReceivedInput}$ is uninitialized.

Although we believe this issue does not affect real-world protocols, it highlights the need to refine the notion of subroutine respecting protocols. This refinement is introduced in Section 6.5.

(I.2) The Case of "Dynamic Sub-parties". We introduce two concepts to explain this issue. An ITI M is said to be *awake* in an execution prefix if it has been invoked and has not rejected all received messages. Additionally, we say that an ITI M is a *dynamic sub-party* of a protocol ϕ if there is a sid s and two execution prefixes $\theta_{\mathsf{in}}, \theta_{\mathsf{out}}$ such that $M \in \mathsf{Ext}(\phi, \mathsf{s}, \theta_{\mathsf{in}})$, but $M \notin \mathsf{Ext}(\phi, \mathsf{s}, \theta_{\mathsf{out}})$ and M is awake in θ_{out}.

Dynamic sub-parties do not align well with Plain UC because, during the execution of ϕ with an environment and adversary, the execution rules indicate that if M exists, it will always be a sub-party of the extended test session. Consequently, these executions do not consider scenarios where M is awake but outside the extended session. In particular, some protocols with dynamic sub-parties could lead to counterexamples of the UC Theorem.

The main difficulty in constructing a protocol with dynamic sub-parties that can be used to contradict the UC Theorem lies in finding one that is subroutine respecting. Intuitively, one might expect this property to exclude any protocol with dynamic sub-parties, given the multiple constraints it imposes on the sub-parties of a protocol. However, the definition of a dynamic sub-party is not inherently incompatible with the requirements of a subroutine respecting protocol. For a protocol ϕ with a dynamic sub-party M to be subroutine respecting, the following must hold:

- M satisfies all requirements imposed on sub-parties in every execution of any protocol in which M appears as a sub-party of (ϕ, s) for some s;
- M is not required to satisfy these requirements in executions of protocols in which M does not appear as a sub-party of (ϕ, s) for any s.

We present such a protocol ϕ in Figure 8. We claim that it is a subroutine respecting protocol with a dynamic sub-party $(\alpha, (0,0))$. Specifically:

- $(\alpha, (0,0))$ is a sub-party of $\mathsf{Ext}(\phi, 0)$ if it is invoked with an input by $(\phi, (0,0))$.
- $(\alpha, (0,0))$ is awake and not a sub-party of $\mathsf{Ext}(\phi, 0)$ if it is invoked with an input by another ITI. Moreover, it is never a sub-party of $\mathsf{Ext}(\phi, \mathsf{s})$ for $\mathsf{s} \neq 0$.

We explain in this sidebar the key reason why ϕ is subroutine respecting.

> We fix the execution of any protocol[26]. In this execution, note that if $(\alpha, (0,0))$ is not invoked by an input from $(\phi, (0,0))$, it sends an input to $(\phi, (0,0))$. This input can be interpreted as a "warning" to inform $(\phi, (0,0))$ that $(\alpha, (0,0))$ is awake and does not belong to $\mathsf{Ext}(\phi, 0)$. Once $(\phi, (0,0))$ is "warned", it refrains from sending any input to $(\alpha, (0,0))$, thereby maintaining the conditions of the subroutine respecting property[27].

In Appendix B.2, we show that there exists a protocol π that UC-emulates a slight modification of ϕ, and a protocol ρ such that all the conditions of the UC theorem are satisfied. However, $\rho^{\phi \to \pi}$ does not UC-emulate ρ, thereby contradicting the UC theorem.

The core reason for the counterexample lies in the difference in the observable behavior[28] of ϕ when executed alone versus as a subroutine within another protocol. We detail it in the following sidebar.

[26] Recall that the conditions ensuring that a protocol ϕ is subroutine respecting must hold not only in executions of ϕ itself, but also in executions of any protocol in which ϕ may appear as a sub-party.

[27] Note that a complete proof (for a slight modification of ϕ) is provided in Appendix B.2.

[28] *i.e.*, observable by the environment.

ρ is a protocol distinct of α and ϕ.
Moreover, all external writes are made with flags reveal-ext-id and forced-write set.

(ϕ, id)

local variable: boolean FirstInputMade, initialized to false
if id $\neq (0,0)$ **then reject**
At reception of an input m from ITI M :
if FirstInputMade is false **then**
  FirstInputMade $\leftarrow$ true
  if $M \neq (\alpha, (0,0))$
    $\wedge$ first bit of randomness is 0 **then**
      send input "0" to $(\alpha, (0,0))$
  else
    send output "0" to $(\rho, (0,0))$
send output "0" to $(\rho, (0,0))$
At reception of output m from ITI M :
if $M = (\alpha, (0,0))$ **then**
  send output "0" to $(\rho, (0,0))$
else reject
At reception of backdoor : **reject**

(α, id)

local variable: boolean FirstInputMade, initialized to false
if id $\neq (0,0)$ **then reject**
At reception of input m from ITI M :
if FirstInputMade is false **then**
  FirstInputMade $\leftarrow$ true
  if $M = (\phi, (0,0))$ **then**
    send output "0" to $(\phi, (0,0))$
  else
    send input "0" to $(\phi, (0,0))$
else reject
At reception of backdoor : **reject**
At reception of output : **reject**

Figure 8: Algorithms of ϕ with dynamic sub-party α, for Issue (I.2).

In any execution of ϕ with test sid 0, $(\phi, (0,0))$ behaves essentially like a protocol that sends an output to $(\rho, (0,0))$ upon receiving any input – regardless of whether it creates the subroutine $(\alpha, (0,0))$. In this context, the creation of the subroutine has no observable effect.

However, this changes when ϕ is used as a subroutine by a protocol. A new observable behavior arises. Suppose, for example, that $(\rho, (0,0))$ first sends an input to $(\phi, (0,0))$, and then sends an output to $(\alpha, (0,0))$. The result of this second input, sent to $(\alpha, (0,0))$, depends on the first bit of randomness used by ϕ:

- If it is 0, then $(\alpha, (0,0))$ has already been invoked by $(\phi, (0,0))$. As a consequence, $(\alpha, (0,0))$ rejects the input.
- If it is 1, then this message actually invokes $(\alpha, (0,0))$. Because $(\alpha, (0,0))$ is invoked by an ITI distinct from $(\phi, (0,0))$ – namely, $(\rho, (0,0))$ – it subsequently sends an input to $(\phi, (0,0))$, which in turn sends an output to $(\rho, (0,0))$.

To address this issue, Updated UC's variant of the subroutine respecting property (Section 6.1) excludes protocols with dynamic sub-parties. This is achieved through the introduction of the Panoptic Session concept, described in Section 6.3.

(I.3) Subroutine Respecting Property and Universal Composition Operator. This issue is particularly subtle, and we will only provide a brief explanation of the mechanism it exploits, with a full counterexample of the UC theorem presented in Appendix B.3.

When considering the UC theorem for protocols ρ, ϕ, π, an important hypothesis is that π is subroutine respecting. This property gives restrictions to the ITIs with code π and its sub-parties in any protocol execution.

In demonstrating the UC theorem, it's crucial to verify these conditions within the

execution of $\rho^{\phi \to \pi}$. However, such executions use structured protocols to simulate special kind of executions of ρ, where instances of ϕ are substituted with instances of π. In these executions, we can demonstrate that the simulated extended sessions of π may encounter situations that do not arise in any real extended sessions of π during any protocol execution. By definition, the subroutine respecting property of π does not cover such cases, and this absence of constraints can be exploited to construct a counterexample to the UC theorem.

While we did not find a concrete protocol with this issue, it is crucial to address it to ensure the correctness of the UC Theorem.

In Updated UC, the notion of sub-party respecting (Section 6.5) addresses this issue by requiring that its restrictions hold not only for any protocol execution but also for any extended execution.

5.2 Problems Related to the Universal Composition Operator

In this section, we present an issue and a weakness linked to the UC operator [Can20, (Section 6.1, item 2)].

The issue highlights the necessity of replacing the UC operator, while the weakness underscores the value of having a modular formulation of the UC theorem that does not rely on a specific UC Operator implementation. These considerations are further discussed for Updated UC in Section 6.7.

(I.4) An Issue in Handling Outputs Forwarded to the Environment.

In this section, we examine a subtle mechanism required by the UC operator that was omitted from the overview for simplicity (see Section 4, page 18). We show that the UC operator in Plain UC does not implement this mechanism correctly, which we exploit to construct a counterexample to the UC theorem.

Recall that, in the execution of a protocol ρ with environment $\mathcal{E}$ and some adversary, if a main party $(\rho, (\mathsf{s}_{\mathsf{test}}, \mathsf{p}))$ of the extended test session sends an output to an ITI M that does not exist in the execution, the output is instead sent to $(\mathcal{E}, 0)$. Additionally, the identity $(\mathsf{s}_{\mathsf{test}}, \mathsf{p})$ and the extended identity of M are also forwarded to $(\mathcal{E}, 0)$. Recall further that the ITIs of an extended session $\mathsf{Ext}(\rho^{\phi \to \pi}, \mathsf{s})$ are structured ITIs whose bodies correspond either to ITIs of $\mathsf{Ext}(\rho, \mathsf{s})$ or of extended sessions of π.

As a result, when the body of a structured ITI attempts to send such an output to an ITI M, it is crucial for the shell to determine whether a structured ITI S with body M exists in the execution. If such a structured ITI S exists, the message must be directed to S. Otherwise, the recipient must remain unmodified, as the output will be forwarded to $(\mathcal{E}, 0)$ along with the identity of the intended recipient. In this latter case, using S as the recipient instead of M would allow the environment to distinguish between executions of $\rho^{\phi \to \pi}$ and ρ.

However, the definition of $\rho^{\phi \to \pi}$ does not address this situation correctly. In an execution of $\rho^{\phi \to \pi}$ with test sid $\mathsf{s}_{\mathsf{test}}$, when the body of a main party of $\mathsf{Ext}\left(\rho^{\phi \to \pi}, \mathsf{s}_{\mathsf{test}}\right)$ attempts to send an output to an ITI $(\alpha, (\mathsf{s}', \mathsf{p}'))$, the procedure described in [Can20, (Section 6.1, item 2)] is as follows:

- If $(\alpha, \mathsf{s}') \neq (\rho, \mathsf{s}_{\mathsf{test}})$, then $\left(\rho^{\phi \to \pi}, (\mathsf{s}_{\mathsf{test}}, \mathsf{p})\right)$ sends the output to $(\alpha, (\mathsf{s}', \mathsf{p}'))$.
- Otherwise, if $(\alpha, \mathsf{s}') = (\rho, \mathsf{s}_{\mathsf{test}})$, then the output is sent to $\left(\rho^{\phi \to \pi}, (\mathsf{s}_{\mathsf{test}}, \mathsf{p}')\right)$ – *i.e.*, the structured ITI with body $(\rho, (\mathsf{s}_{\mathsf{test}}, \mathsf{p}'))$.

Consequently, $\left(\rho^{\phi \to \pi}, (\mathsf{s}_{\mathsf{test}}, \mathsf{p})\right)$ behaves as if, whenever $(\alpha, \mathsf{s}') \neq (\rho, \mathsf{s}_{\mathsf{test}})$, no structured ITI with body $(\alpha, (\mathsf{s}', \mathsf{p}'))$ exists. This assumption is incorrect and leads to counterexamples to the UC theorem, as detailed in Appendix B.4.

To correctly select the recipient in this situation, $\left(\rho^{\phi \to \pi}, (\mathsf{s}_{\mathsf{test}}, \mathsf{p})\right)$ must be able to determine whether a structured ITI with body M is present in the execution. To enable this, the UC operator proposed in Updated UC uses a centralized approach: it employs a

special ITI that tracks exchanged messages and maintains global awareness of instantiated structured protocols.

(W.1) Proof of UC theorem and structured ITIs.

In the proof of the UC Theorem [Can20, (Theorem 22)] for protocols ρ, ϕ, and π, the objective is to establish the existence of a PPT adversary $\mathcal{S}$ such that the following relation [Can20, (Equation 2, Section 6.1)] holds for any PPT environment $\mathcal{E}$: $\mathsf{EXEC}_{\rho^{\phi\to\pi},\mathcal{D},\mathcal{E}} \approx \mathsf{EXEC}_{\rho,\mathcal{S},\mathcal{E}}$.

In this section, we briefly highlight a subtle flaw in the definition of $\mathcal{S}$ [Can20, Figure 10], which renders the proof of the UC Theorem incorrect. A full description of the problem and its resolution is provided in Appendix B.5. However, the precise description of it is not our main focus here. What matters is that both the problem and its resolution depend on the specific construction of the UC operator $\rho^{\phi\to\pi}$ – in particular, on the fact that $\rho^{\phi\to\pi}$ is a structured protocol and on its shell.

We argue that the existence of this problem stems from the inherent complexity of the proof, which simultaneously involves several subtle notions, including the internal structure of the UC operator. We believe it would be advantageous to formulate the UC Theorem in a way that does not rely on a specific operator construction. Such a formulation offers two key benefits: it simplifies the analysis of the proof and makes it easier to accommodate alternative operator constructions. This is precisely the approach adopted in Updated UC: we define the UC operator abstractly, by highlighting the conditions it must satisfy, rather than relying on a specific instantiation.

We now briefly describe the flaw. In essence, it arises when the simulator $\mathcal{S}$ receives an input of the form (m, M)—where m is a message and M an ITI—and performs certain operations and checks directly on M (see Items 1.(a), 1.(b), and 1.(c) in [Can20, Figure 10]). These steps should instead be handled as follows. First, $\mathcal{S}$ should reject whenever M is not a structured protocol sharing the same shell as $\rho^{\phi\to\pi}$. Second, if this condition holds and M indeed represents such a structured protocol, the corresponding operations and verifications must be performed on the body of M, rather than on M itself.

As shown in Appendix B.5, without this correction, the proof of the UC Theorem does not hold. That appendix also provides a detailed account of the required modifications.

5.3 Problems Related to the Subroutine Exposing Property

This section presents a limitation and a weakness related to the *subroutine exposing property* ([Can20, (Definition 21)]), which is required for ϕ and π to apply the UC theorem.

To ease the reading, we recall the definition of a subroutine exposing protocol as provided in the technical overview (Section 2.4). For a protocol α, the subroutine exposing property requires that for each sid s, one of the ITIs in the extended session $\mathsf{Ext}(\alpha, \mathsf{s})$ can inform – upon request – the adversary whether an ITI M belongs to the extended session $\mathsf{Ext}(\alpha, \mathsf{s})$ at the time of the request. We refer to this ITI as the exposing ITI of the extended session $\mathsf{Ext}(\alpha, \mathsf{s})$, or simply the exposing ITI when the context is clear. It is also required that the extended identity of the exposing ITI for the extended session $\mathsf{Ext}(\alpha, \mathsf{s})$ can be determined using only α and s.[29]

(L.2) Adding the Subroutine Exposing Property to a Protocol Does Not Preserve UC-Emulation.

To improve readability, we restate here part of the explanation previously given in the technical overview (Section 2.4.1). Nevertheless, we encourage the reader to refer to that section for the full context of this limitation.

There is a straightforward mechanism, described in [Can20, (Section 6.1)], that transforms any protocol ϕ into a subroutine exposing protocol ϕ_{SE}. In summary, ϕ_{SE} introduces,

[29] The exposing ITI is sometimes called a "directory ITI" in Plain UC.

All external writes are made with flags reveal-ext-id and forced-write set.

(π, id)	(α, id)
if $\mathsf{id} \neq (0,0)$ **then reject**	**reject**
At reception of an output or a backdoor : **reject**	
At reception of an input m from an ITI M :	(ϕ, id)
if it is first activation $\wedge\, m = \mathsf{0}$ **then send input** "0" to $(\alpha, (0,0))$	**reject**
else reject	

Figure 9: Algorithms of π, α and ϕ for Limitation (L.2).

in any extended session $\mathsf{Ext}(\phi_{\mathsf{SE}}, \mathsf{s})$, a middle-man $M_{\phi,\mathsf{s}}$ that is aware of all ITIs belonging to the extended session and can therefore act as an exposing ITI, telling the adversary whether any ITI N belongs to $\mathsf{Ext}(\phi_{\mathsf{SE}}, \mathsf{s})$.

To the best of our knowledge, most protocols in the UC framework are not subroutine exposing. We substantiate this claim in Appendix B.6 by exhibiting several ideal protocols that lack this property, including the ideal functionalities $\mathcal{F}_{\mathsf{AUTH}}$, $\mathcal{F}^{l}_{\mathsf{SMT}}$, $\mathcal{F}_{\mathsf{SYN}}$ and $\mathcal{F}_{\mathsf{NC}}$ of [Can20]. Consequently, applying the transformation is essential to endow them with it and to ensure they benefit from the security guarantees of the UC theorem.

For the transformation to be sound, it must preserve UC-emulation: whenever π UC-emulates ϕ, we should also have that π_{SE} UC-emulates ϕ_{SE}. However, this preservation is not guaranteed. We demonstrate this with a simple, intentionally constructed counterexample.

Let π and ϕ be the protocols described in Figure 9. Since neither π nor ϕ produces any output or interacts with the adversary, we know that π UC-emulates ϕ. Furthermore, both protocols are subroutine respecting. As a result, the only missing property for applying the UC theorem is subroutine exposing. Nevertheless, π_{SE} does not UC-emulate ϕ_{SE}. To see this, observe that $(\alpha, (0,0))$ belongs to $\mathsf{Ext}(\pi, 0)$ if and only if the environment invoked $(\pi, (0,0))$ with input "0". This implies that an adversary of π_{SE} can use the subroutine exposing ITI to determine whether the environment invoked the protocol with identifier $(0,0)$ using input "0". In contrast, an adversary of ϕ_{SE} lacks such a mechanism.

This problem can arise in real-world protocols, where the existence of an ITI depends on an action taken by the environment. An example is given in the technical overview (Section 2.4).

The implications of this limitation are twofold. First, it shows that the subroutine-exposing property must be explicitly verified in protocols analyzed under Plain UC, since the generic transformation does not preserve UC-emulation – a verification that, to the best of our knowledge, is generally omitted. Second, it reveals that certain realistic examples of UC-emulation cannot be directly leveraged within the UC Theorem.

In Updated UC, we adopt a variant of subroutine exposing (Section 6.4) that eliminates the need for such a transformation and enables the analysis of more protocols, including the example from the technical overview (Section 2.4).

(W.2) Determining the sids of Existing Sessions of ϕ. In the proof of UC theorem [Can20, (Theorem 22)] for protocols ρ, ϕ, and π,[30] the author constructs a PPT adversary $\mathcal{S}$ [Can20, (Figure 10)] of ρ that behaves differently when receiving a backdoor from an ITI M, depending on whether there exists a sid s such that $M \in \mathsf{Ext}(\phi, \mathsf{s})$.

$\mathcal{S}$ relies on the subroutine exposing property of ϕ to determine if an ITI belongs to an extended session of ϕ during the execution of ρ (see [Can20, (2.(a) of Figure 10)]). Since

[30] In discussing this weakness, we simplify the explanation by omitting the concept of top-level sessions [Can20, (Section 6.1)], which is not essential for understanding and will not be required for the rest of the document.

using the exposing ITI of an extended session $\mathsf{Ext}(\phi, \mathsf{s})$ generally requires knowing s, $\mathcal{S}$ must know the sids of the existing extended sessions of ϕ in the current execution of ρ. However, no mechanism is provided to identify them. This is critical because the number of possible sids could be exponential, meaning that the adversary would not be PPT if required to test all possible sids.

For instance, consider a protocol ρ that, upon activation, samples a random $\mathsf{s} \in \{0,1\}^k$ and then invokes $(\phi, (\mathsf{s}, 0))$: the number of possible sids is exponential in the security parameter k.

We believe than this weakness cannot be solved in Plain UC without additional properties. A direct solution could be to only consider protocols ρ that informs the adversary of these sids. Nevertheless, we think this would overly restrict the applicability of the UC theorem.

In Updated UC, this weakness does not exist due to our use of a variant of the subroutine exposing property, described in Section 6.4.

5.4 Problem Related to the Compliance Property

This section presents a weakness related to the *compliance* property ([Can20, (Section 6.1)]), which is required to apply the UC theorem. It highlights that this notion is not restrictive enough.

Compliance is the sole requirement concerning the protocol ρ in the UC Theorem, aside from being PPT. Simply put, for protocols ϕ and π, and ξ a predicate on extended identities, ρ is (ϕ, π, ξ)-compliant [Can20, (Section 6.1)] if, in any execution[31] of ρ: (1) some ITIs satisfy ξ; and (2) if two external-write made by protocol ITIs have target codes π and ϕ, they have distinct target sids.

(W.3) Identifying Whether π is the Result of a Replacement.

In the proof of the UC theorem [Can20, (Theorem 22)] for protocols ρ, ϕ, and π,[32] the author constructs a PPT adversary $\mathcal{S}$ [Can20, (Figure 10)] for ρ that can behave in two ways upon receiving an input (m, M) from the environment $(\mathcal{E}, 0)$: either as the dummy adversary or as a copy of an adversary $\mathcal{S}_\pi$ of ϕ such that $\mathsf{EXEC}_{\pi,\mathcal{D},\mathcal{E}} \approx \mathsf{EXEC}_{\phi,\mathcal{S}_\pi,\mathcal{E}}$.

The compliance notion does not prohibit ρ from invoking ITMs with code π. In this setting, an ITI M with body $(\pi, (\mathsf{s}, \mathsf{p}))$ appearing in an extended session of $\rho^{\phi \to \pi}$ can be classified into two types:

- Type 1: M was invoked as the result of a replacement $\phi \to \pi$,
- Type 2: M is not the result of a replacement.

We explain in the next paragraph that the existence of these two types leads to a problem in the proof of the UC Theorem.

As a consequence, in Updated UC we propose updating the compliance notion to exclude protocols ρ that can invoke main parties of π, thereby preventing the occurrence of Type 2 ITIs. As a result, only Type 1 ITIs may appear, which avoids the problem. This restriction is explicitly enforced in the compliance definition introduced in Section 6.6.

Detailed explanation of the problem. Consider two protocols ϕ and π such that π UC-emulates ϕ, with the simulator corresponding to the dummy adversary $\mathcal{D}$ denoted by $\mathcal{S}_\pi$. For simplicity, assume that ϕ and π never send inputs, and that all outputs they produce are directed towards a protocol ρ.

The UC-emulation of ϕ by π is illustrated in Figure 10 (identifiers omitted for simplicity):

[31] Note that our simplifications of protocol execution (Section 4, page 15) reduces the complexity of this notion, particularly because: (1) $(\mathcal{E}, 0)$ always sets reveal-ext-id, and (2) all protocol ITIs set flag forced-write.

[32] Same as note 30 on page 26.

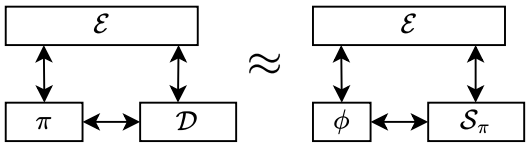


Figure 10: UC-emulation of ϕ by π. We omit the identifiers for simplicity.

Furthermore, assume that in any execution of ρ, ρ interacts exclusively with a single main party of π. Under this assumption, in any execution of $\rho^{\phi\to\pi}$, no replacements $\phi \to \pi$ occur.

We further assume that ϕ, π, and ρ satisfy all additional conditions required for the UC theorem to apply. The UC-emulation of ρ by $\rho^{\phi\to\pi}$ is depicted in Figure 11, where $\mathcal{S}$ [Can20, (Figure 10)] is the simulator constructed in the proof of the UC theorem.

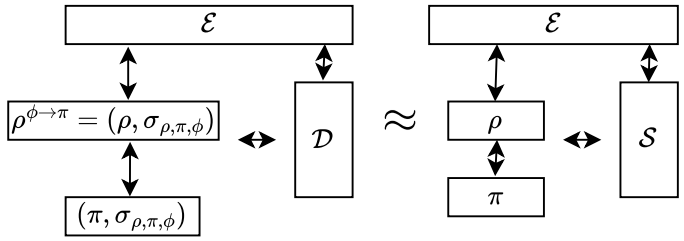


Figure 11: UC-emulation of ρ by $\rho^{\phi\to\pi}$, with simulator $\mathcal{S}$. We omit the identifiers for simplicity.

In this situation, Figure 11 clearly illustrates that, for the indistinguishability of Figure 11 to hold, $\mathcal{S}$ should essentially behave like the dummy adversary $\mathcal{D}$. Indeed, the protocols in both executions are essentially identical, except for the presence or absence of shells; in particular, no replacement $\phi \to \pi$ is performed by $\rho^{\phi\to\pi}$.

Specifically, upon receiving an input (m, M), where $M = \pi$, $\mathcal{S}$ should simply forward a backdoor m to π (we continue to omit identifiers for simplicity).

However, according to the current construction of $\mathcal{S}$ (specifically, Item 1.(b) of [Can20, (Figure 10)]), upon receiving such an input (m, M), $\mathcal{S}$ can behave as a copy of $\mathcal{S}_\pi$. There is then no guarantee that the behavior of this copy of $\mathcal{S}_\pi$ led to a result indistinguishable from sending the backdoor m to π. This discrepancy is precisely where the problem lies: even if subtle, it opens the door to environments that can easily distinguish the left and right executions of Figure 11, thus invalidating the proof of the UC Theorem.

(W.4) Compliance Property and Universal Composition Operator.

In Issue (I.3) (page 23), we have shown that the conditions required by the subroutine respecting notion should be verified not only in protocol executions but also in more general forms of execution.

A similar problem arises with the conditions of the compliance notion. We classify this as a weakness rather than an issue, as we have not provided a counterexample that exploits it.

Given the similarity of the two problems, one might logically attempt to address it in a similar manner—by requiring the conditions of the compliance property to hold not only for any protocol execution but also for any extended execution.

However, the conditions required by the subroutine respecting and compliance properties differ significantly, and directly applying this solution would be overly restrictive – making it nearly impossible for a protocol to be compliant.

To update the compliance notion without being overly restrictive, we need a more precise characterization of the set of extended execution that should be considered. To that end, Updated UC introduces a new kind of execution, called replacement execution (see Section 6.2).

6 Updated UC: An Update of Plain UC

This section introduces Updated UC, an update to the Plain UC framework that revises several core definitions to address the issues identified in earlier sections.

Each modification in Updated UC will be motivated by specific problems in Plain UC described in the preceding sections.

For clarity of exposition, the universal composition theorem is presented separately in Section 7.

6.1 A New Notion of Extended Session

As shown in Limitation (L.1) (Section 5.1), the fact that an ITI rejects a message should not prevent it from belonging to an extended session. In Updated UC, we modify the definition of extended session to account for this.

First, we introduce the notion of a sleeping ITI: an ITI is considered sleeping if it does not exist (*i.e.* it has not been invoked), or if it has always rejected every message it has received. Note that this implies that, upon receiving a message, a sleeping ITI reacts as if it were handling its first message.

Then, when constructing the extended session of a protocol, we no longer focus solely on messages received by newly invoked ITIs, as in the original definition (*i.e.*, ITIs that did not exist before receiving the message). Instead, we inspect each sleeping ITI that receives a message.

Definition 2 (Sleep)**.** An ITI M is sleeping in a prefix of execution θ of an extended system if one of the following holds:

- M does not exist in θ (*i.e.* it has not been invoked).
- M has rejected every message it received – including the most recent one. That is, if the prefix ends with a message being delivered to M, we use the full configuration of M (as recorded in θ) to determine whether it rejects the message.

An ITI M is awake if it is not sleeping.

Definition 3 (Main parties and extended session)**.** Let α be an ITM and $\mathsf{s} \in \{0,1\}^*$ a sid. The main parties of $\mathsf{Ext}(\alpha, \mathsf{s})$ are defined as $\mathsf{MainParty}(\alpha, \mathsf{s}) = \{(\alpha, (\mathsf{s}, \mathsf{p})) : \mathsf{p} \in \{0,1\}^*\}$.

Given a prefix θ of extended execution, the extended session $\mathsf{Ext}(\alpha, \mathsf{s}, \theta)$ is defined as a set of ITIs initialized [33] to $\mathsf{MainParty}(\alpha, \mathsf{s})$, and constructed inductively by inspecting each input or output received by a sleeping ITI, except the inputs sent by the environment. When a sleeping ITI M receives an input or output with sender an ITI N:

- If the identifier of M is not of the form $(\mathsf{s}', \mathsf{p}')$ for some $\mathsf{s}', \mathsf{p}' \in \{0,1\}^*$, nothing is done. In the next items, we assume this condition holds.
- If $M \in \mathsf{MainParty}(\alpha, \mathsf{s})$, then M already belongs to $\mathsf{Ext}(\alpha, \mathsf{s}, \theta)$.
- If it is an input and $N \in \mathsf{Ext}(\alpha, \mathsf{s}, \theta)$, then M is added to $\mathsf{Ext}(\alpha, \mathsf{s}, \theta)$.
- If it is an output and $N \in \mathsf{Ext}(\alpha, \mathsf{s}, \theta) \setminus \mathsf{MainParty}(\alpha, \mathsf{s})$, then M is added to $\mathsf{Ext}(\alpha, \mathsf{s}, \theta)$.
- Otherwise, nothing is done.

We define $\mathsf{ExtSub}(\alpha, \mathsf{s}, \theta) = \mathsf{Ext}(\alpha, \mathsf{s}, \theta) \setminus \mathsf{MainParty}(\alpha, \mathsf{s})$, and call the ITIs in this set the sub-parties.

When the context is clear, we omit θ and simply write $\mathsf{Ext}(\alpha, \mathsf{s})$ and $\mathsf{ExtSub}(\alpha, \mathsf{s})$.

[33] In the original definition, $\mathsf{MainParty}(\alpha, \mathsf{s}) = \{(\alpha, (\mathsf{s}, \mathsf{p})) : \mathsf{p} \in \{0,1\}^*\}$ is not a subset of $\mathsf{Ext}(\alpha, \mathsf{s})$, since extended identities $(\alpha, (\mathsf{s}, \mathsf{p}))$ that have not been invoked are not regarded as main parties. Our change is a convenient convention.

6.2 Replacement Execution: A (Not Entirely) New Notion of Execution

Updated UC introduces a new notion of execution that depends on two protocols ϕ, π and a number $\ell \in \mathbb{N}$. Simply put, the (ϕ, ℓ, π)-execution of a protocol ρ corresponds to executing ρ while replacing the first ℓ calls to ϕ by calls to π, and informing the adversary of the sids of the replaced extended sessions.

Although variants of such executions already appear in the proof of the UC theorem [Can20, (Theorem 22)] within Plain UC, they were not used to define any protocol property. However, ignoring such executions leads to problems such as Issue (I.3). and Weakness (W.4). In Updated UC, the replacement executions will be used to update the compliance notion (Section 6.6) and define the notion of UC Operator (Section 6.7). Moreover, the proof of the UC Theorem made in Appendix C will extensively use them.

Replacement Execution of an ITM.

Definition 4 ((ϕ, ℓ, π)-execution)**.** Let ρ, ϕ, and π be three ITMs, $\ell \in \mathbb{N} \cup \{\infty\}$, and let $\mathcal{A}, \mathcal{E}$ be two ITMs.

The (ϕ, ℓ, π)-execution of ρ with environment $\mathcal{E}$ and adversary $\mathcal{A}$ is defined as the execution of the extended system $(\mathcal{E}, \mathcal{C}^{\rho,\mathcal{A},\mathcal{E}}_{(\phi,\ell,\pi)\text{-}\mathsf{EXEC}})$, where the control function $\mathcal{C}^{\rho,\mathcal{A},\mathcal{E}}_{(\phi,\ell,\pi)\text{-}\mathsf{EXEC}}$ is defined similarly to $\mathcal{C}^{\rho,\mathcal{A},\mathcal{E}}_{\mathsf{EXEC}}$, with some differences listed below:

1. *Tracking the sids of replaced extended sessions:* The execution maintains a set $\mathsf{ReplacedSession} \subseteq \{0,1\}^*$ (initialized to $\varnothing$) that tracks the sids of replaced sessions – as detailed in next items. Its size is bounded by ℓ.
   In the following items, we will denote an ITI as "outside" if it is neither the environment $(\mathcal{E}, 0)$ nor the adversary $(\mathcal{A}, \Diamond)$, and if it does not belong to:
   - $\mathsf{Ext}(\phi, \mathsf{s})$ for any $\mathsf{s} \in \{0,1\}^*$,
   - $\mathsf{Ext}(\pi, \mathsf{s})$ for any $\mathsf{s} \in \mathsf{ReplacedSession}$.
2. *Replacement $\phi \to \pi$:* When an input or output is sent from an outside ITI to an ITI $(\phi, (\mathsf{s}, \mathsf{p}))$ for some $\mathsf{s}, \mathsf{p} \in \{0,1\}^*$:
   - If $\mathsf{s} \in \mathsf{ReplacedSession}$ or if $|\mathsf{ReplacedSession}| < \ell$, the message is instead sent to $(\pi, (\mathsf{s}, \mathsf{p}))$ and s is added to $\mathsf{ReplacedSession}$ (if not already present).
   - Otherwise, the message is not redirected and the execution continues.
3. *Replacement $\pi \to \phi$:* When an output is sent from an ITI $(\pi, (\mathsf{s}, \mathsf{p}))$ for some pid $\mathsf{p} \in \{0,1\}^*$ and sid $\mathsf{s} \in \mathsf{ReplacedSession}$, to an ITI $N \notin \mathsf{Ext}(\pi, \mathsf{s})$, the sender ITI $(\pi, (\mathsf{s}, \mathsf{p}))$ is replaced by $(\phi, (\mathsf{s}, \mathsf{p}))$.
4. *Adversary learns* $\mathsf{ReplacedSession}$*:* Whenever a message m is sent to the adversary, the control function replaces it with the pair $(\mathsf{ReplacedSessions}, m)$. This ensures the adversary always knows the current value of $\mathsf{ReplacedSessions}$.

We denote by $(\phi, \ell, \pi)\text{-}\mathsf{EXEC}_{\rho,\mathcal{A},\mathcal{E}}(z)$ the random variable representing the result of executing $(\mathcal{E}, \mathcal{C}^{\rho,\mathcal{A},\mathcal{E}}_{(\phi,\ell,\pi)\text{-}\mathsf{EXEC}})$ on input z.

Note that replacement execution is a generalization of the standard notion of protocol execution. Indeed, for any protocol ρ, a protocol execution of ρ corresponds to a $(\phi, 0, \pi)$-execution of ρ. The only difference is that, in the replacement execution, any message received by the adversary is a pair whose first element is the set $\varnothing$.

Note also that a replacement execution differs from applying the UC operator in two key respects:

- In a replacement execution, modifications are performed through control functions, which can directly substitute any external operation with another. In contrast, the UC operator must explicitly alter the involved protocols to simulate such changes.
- A replacement execution may perform partial replacements, whereas the UC operator applies a complete substitution.

Furthermore, the replacement execution should not be viewed as a substitute for the UC operator itself. Rather, it serves as a technical tool used to define certain notions within Updated UC. A closely related concept already appears in the proof of the UC Theorem in Plain UC [Can20, (Theorem 22)].

Replacement Execution of a Parametrized System.

Let $\ell \in \mathbb{N} \cup \{\infty\}$, and let $\boldsymbol{\rho}$, $\boldsymbol{\phi}$, and $\boldsymbol{\pi}$ be three parametrized systems, and $\mathcal{A}, \mathcal{E}$ two ITMs. The $(\boldsymbol{\phi}, \ell, \boldsymbol{\pi})$-execution of $\boldsymbol{\rho}$ with adversary $\mathcal{A}$ and environment $\mathcal{E}$ is defined analogously, with the following differences:

- All inputs to $\mathcal{E}$ must be pairs (k, z) with $k \in \mathbb{N}$ and $z \in \{0,1\}^*$. The value k is the security parameter.
- The adversary has read-only access to the security parameter.
- When the security parameter is k, the execution proceeds as a (ϕ_k, ℓ, π_k)-execution of ρ_k.

We denote the corresponding control function by $\mathcal{C}^{\boldsymbol{\rho},\mathcal{A},\mathcal{E}}_{(\boldsymbol{\phi},\ell,\boldsymbol{\pi})\text{-}\mathsf{EXEC}}$ and the result of the execution by $(\boldsymbol{\phi}, \ell, \boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{A},\mathcal{E}}(k, z)$.

PPT Triplet of Parametrized Systems.

In Section 4, we defined the notion of a PPT protocol in the context of standard protocol execution. We now define an analogous notion for replacement executions.

A triplet $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ of parametrized systems is said to be PPT if, for every PPT environment $\mathcal{E}$ and PPT adversary $\mathcal{A}$, there exists a polynomial R such that for any $\ell \in \mathbb{N}$ and any input $(k, z) \in \mathbb{N} \times \{0,1\}^*$, any $(\boldsymbol{\phi}, \ell, \boldsymbol{\pi})$-execution of $\boldsymbol{\rho}$ with environment $\mathcal{E}$ and adversary $\mathcal{A}$ completes in time bounded by $R(k + |z|)$.

Replacement Execution Prefixes.

Just as we defined prefixes of protocol executions (Section 4, page 16), we now define prefixes for replacement executions.

A (ρ, ϕ, ℓ, π)-execution prefix is a prefix of an (ϕ, ℓ, π)-execution of ρ. We refer to any such prefix as a *replacement execution prefix*.

6.3 Panoptic Session

To define our variants of the subroutine respecting and subroutine exposing properties, we introduce a new construction whose use will be central in avoiding several issues identified in the first part of the paper—primarily Issue (I.2), Limitation (L.2), and Weakness (W.2).

Definition 5 (Panoptic session)**.** Let α be an ITM, and let s be a sid. We define the panoptic session of α with sid s, as

$$\mathsf{Pan}(\alpha, \mathsf{s}) = \bigcup_{\theta} \mathsf{Ext}(\alpha, \mathsf{s}, \theta),$$

where the union is taken over all α-execution prefixes with test sid s.

We define the set of panoptic sub-parties as

$$\mathsf{PanSub}(\alpha, \mathsf{s}) = \mathsf{Pan}(\alpha, \mathsf{s}) \setminus \mathsf{MainParty}(\alpha, \mathsf{s})\ .$$

Additionally, for an ITM α, we define

$$\mathsf{Pan}(\alpha) = \{\beta \text{ ITM} : \exists \mathsf{id}, \mathsf{s} \in \{0,1\}^*,\ (\beta, \mathsf{id}) \in \mathsf{Pan}(\alpha, \mathsf{s})\}.$$

We note that computing a panoptic session is no more difficult than computing the extended sessions already required by Plain UC (for example, to verify the subroutine respecting property), as it is simply a union over such sessions.

We also use this notion to define a restriction on the parameterized systems $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ considered in the UC Theorem. This restriction is particularly needed to formulate our update to the subroutine exposing notion and is employed in the proof of the UC Theorem in Updated UC. Although we do not know whether this condition can be relaxed in our setting, we argue that it is reasonable, as in practical settings $\boldsymbol{\phi}$ is typically an ideal protocol that does not share sub-parties with $\boldsymbol{\pi}$.

Definition 6 (Disjunction)**.** Two protocols π and ϕ are said to be disjoint if, for any sids $\mathsf{s}, \mathsf{s}' \in \{0,1\}^*$, we have $\mathsf{Pan}(\pi, \mathsf{s}) \cap \mathsf{Pan}(\phi, \mathsf{s}') = \varnothing$. Two parametrized systems $\boldsymbol{\phi}$ and $\boldsymbol{\pi}$ are said to be disjoint if, for all $k \in \mathbb{N}$, ϕ_k and π_k are disjoint.

6.4 Sid-Exposing

To address the two issues discussed in Section 5.3 regarding the subroutine exposing property, we introduce a new concept called *sid-exposing*.

As a prerequisite for a protocol α to be sid-exposing, we require that all its panoptic sessions be separated, as formally defined below.[34]

Definition 7 (Separation)**.** A protocol α is said to be *separated* if for any two distinct sids $\mathsf{s} \neq \mathsf{s}'$, $\mathsf{Pan}(\alpha, \mathsf{s}) \cap \mathsf{Pan}(\alpha, \mathsf{s}') = \varnothing$. A parametrized system $\boldsymbol{\alpha}$ is separated if each α_k is separated for all $k \in \mathbb{N}$.

Assuming that α is separated, we observe that any ITI M belongs to at most one set $\mathsf{Pan}(\alpha, \mathsf{s})$ for some unique $\mathsf{s} \in \{0,1\}^*$, or to none of them.

The sid-exposing property asks that we can efficiently determine, given an ITI M, the corresponding s, if it exists.

Definition 8 (Sid-Exposing)**.** Let α be a separated protocol and s a sid. We say that α is *sid-exposing* if there exists a PPT Turing machine (TM) Expose^{α}, called the *exposing TM* of α, which takes as input an ITI M and returns the unique sid s such that $M \in \mathsf{Pan}(\alpha, \mathsf{s})$, if such a sid exists, and $\perp$ otherwise.

A separated parametrized system $\boldsymbol{\alpha}$ is sid-exposing if there exists a PPT Turing machine $\mathsf{PSExpose}^{\boldsymbol{\alpha}}$ whose inputs are pairs $(k, M) \in \mathbb{N} \times \mathsf{SetITI}$, and such that $\mathsf{PSExpose}^{\boldsymbol{\alpha}}(k, \cdot)$ is an exposing TM of α_k for each $k \in \mathbb{N}$.

We believe that this property is generally easy to verify, and we are not aware of any protocol in the literature that would fail to be sid-exposing.

This notion presents three main advantages over the subroutine exposing notion, effectively addressing the problems detailed in Section 5.3.

First, the exposing TM Expose^{α} is not associated with a specific sid. In contrast, in the subroutine exposing property, there is a distinct exposing ITI for each sid. This resolves Weakness (W.2).

Second, the notion is *non-interactive*: the adversary can simulate Expose^{α} locally without interacting with the protocol. This makes it more practical and avoids the need for the transformation discussed in (L.2).

Finally, this property reveals strictly less information to the adversary. Subroutine exposing allows the adversary to test whether an ITM M exists in the *current execution*, whereas sid-exposing only reveals whether M exists in *some execution* (possibly not the current one). A positive consequence is that the UC theorem in Updated UC can apply to concrete protocols such as π^* and ϕ^* discussed in the technical details (Section 2.4).

[34] As with the disjunction property, we do not know whether this condition can be relaxed in our setting. We argue that it is reasonable, as we know no protocols in the literature under Plain UC that violate it. Generally, the subroutine of an extended session (α, s) have sid s too, which directly imply the separation property.

6.5 Sub-party Respecting Protocol

As shown in Section 5.1, the subroutine respecting definition lacks a condition about backdoors (I.1), should not account only for protocol executions (I.3), and allows dynamic sub-parties (I.2).

In this section, we introduce a more robust concept called "sub-party respecting" to address these three issues. To achieve this, the sub-party respecting notion incorporates the following key changes compared to the original subroutine respecting definition:

- It prevents Issue (I.1) by adding a condition about the reception of backdoors by sleeping ITIs.
- To address Issue (I.2), we replace extended sessions with panoptic sessions and add a requirement regarding the reception of outputs by sleeping ITIs. Together, these measures prevent protocols with dynamic sub-parties to be sub-party respecting.
- To address Issue (I.3), the conditions are evaluated on any extended execution, not just any protocol execution.

Definition 9 (Sub-party respecting)**.** A protocol α is said to be sub-party respecting if, for any prefix θ of any extended execution, and for each sid s, the following conditions hold:

spr.1 All sleeping ITIs $N \in \mathsf{Pan}(\alpha, \mathsf{s})$ reject any backdoor.

spr.2 All sleeping ITIs $N \in \mathsf{PanSub}(\alpha, \mathsf{s})$ reject any output whose sender extended identity is an ITI $M \in \mathsf{MainParty}(\alpha, \mathsf{s})$.

spr.3 All ITIs of $\mathsf{PanSub}(\alpha, \mathsf{s})$ reject any input whose sender extended identity[35] is an ITI $M \notin \mathsf{Pan}(\alpha, \mathsf{s})$.

spr.4 All ITIs of $\mathsf{Pan}(\alpha, \mathsf{s})$ reject any output whose sender extended identity is an ITI $M \notin \mathsf{Pan}(\alpha, \mathsf{s})$.

spr.5 No ITI of $\mathsf{PanSub}(\alpha, \mathsf{s})$ sends an output to an ITI $M \notin \mathsf{Pan}(\alpha, \mathsf{s})$.

spr.6 No ITI of $\mathsf{Pan}(\alpha, \mathsf{s})$ sends an input to an ITI $M \notin \mathsf{Pan}(\alpha, \mathsf{s})$.

A parametrized system $\boldsymbol{\alpha}$ is sub-party respecting if α_k is sub-party respecting for each $k \in \mathbb{N}$.

Note that it may seem abstract to require these conditions for any prefix of any extended execution. A more concrete and informal interpretation is that they must hold for any possible internal state and any possible message (including those forged with impersonated sender identities).

6.6 Compliance

In Plain UC, compliance [Can20, (Section 6.1)] is the only property that constrains the protocol $\boldsymbol{\rho}$ when applying the UC theorem to protocols $\boldsymbol{\rho}$, $\boldsymbol{\pi}$, and $\boldsymbol{\phi}$[36]. The same principle applies in Updated UC, which employs a slight variant of this notion, as specified in Definition 10.

Our variant introduces two main changes. First, its conditions are not evaluated on any protocol execution, but rather on some replacement executions – this is motivated by Issue (I.3), Section 5.1. Second, to address Weakness (W.3) (also in Section 5.1), we forbid the presence of ITIs of Type 2, using the terminology introduced in that weakness.

[35] Here, we specifically mean the sender's extended identity as indicated at the reception of the message. It is important to note that there is no guarantee the message was actually sent by an ITI with this extended identity, as impersonations are possible.

[36] Except time-execution constraints.

Three additional changes are not directly related to the issues identified in Plain UC but are necessary to ensure the correctness of the UC Theorem in Updated UC. Determining whether the absence of these conditions causes problems within Plain UC is left for future work.

- We require that for each security parameter k, $\rho_k \notin \mathsf{Pan}(\phi_k) \cup \mathsf{Pan}(\pi_k)$. This assumption is reasonable, as $\boldsymbol{\rho}$ typically denotes a "larger protocol" that uses $\boldsymbol{\pi}$ or $\boldsymbol{\phi}$ as subroutines.
- Two additional conditions concerning certain output messages (Item 4 in the definition) is also required. We are not aware of any protocols studied in the literature that fail to satisfy it.

Definition 10 (Compliance)**.** Let π, ϕ, and ρ be protocols such that π and ϕ are separated and disjoint, and let ξ be a predicate on extended identities. The protocol ρ is said to be (π, ϕ, ξ)-compliant if, for every $\ell \in \mathbb{N}$ and every (ρ, ϕ, ℓ, π)-execution prefix θ, the following hold:

1. $\rho \notin \mathsf{Pan}(\phi) \cup \mathsf{Pan}(\pi)$.
2. There is no sid s such that an ITI $M \notin \mathsf{Ext}(\pi, \mathsf{s}, \theta)$ makes an external-write[37] targeting a recipient $N \in \mathsf{MainParty}(\pi, \mathsf{s})$.
3. The extended identities of all ITIs M in θ that send inputs to ITIs with code ϕ satisfy the predicate ξ.[38]
4. There is no sid s such that an ITI $M \in \mathsf{MainParty}(\pi, \mathsf{s})$ intends to send an output to an ITI of $\mathsf{PanSub}(\phi, \mathsf{s})$, that lead to a replacement $\pi \to \phi$ by the control function.
5. There is no sids s, s' such that an ITI $M \in \mathsf{MainParty}(\rho, \mathsf{s})$ sends an output to an ITI of $\mathsf{PanSub}(\phi, \mathsf{s}')$ or $\mathsf{PanSub}(\pi, \mathsf{s}')$.

If $\xi = \text{true}$, we simply say that ρ is (π, ϕ)-compliant.

Let $\boldsymbol{\pi}$, $\boldsymbol{\phi}$, and $\boldsymbol{\rho}$ be parametrized systems. $\boldsymbol{\rho}$ is said to be $(\boldsymbol{\pi}, \boldsymbol{\phi}, \xi)$-compliant if, for every $k \in \mathbb{N}$, the protocol ρ_k is (π_k, ϕ_k, ξ)-compliant.

6.7 Universal Composition Operator and Extension of UC-emulation

To address the concerns related to the universal composition operator (UC operator) in Plain UC (Section 5.2), we propose a general definition of the UC operator. This clarifies the concept and allows us to state the UC Theorem independently of any particular UC operator implementation, making it more general and modular. Since this definition relies on an extension of the UC-emulation relation, we first introduce this extension and then present the definition.

Additionally, we provide in Appendix E a new construction of UC operator that adheres to our definition and mitigates the issues identified with the UC operator of Plain UC. We will conclude this section by explaining the underlying idea behind its construction.

6.7.1 Extension of UC-emulation Relation.

Using the definitions of Section 6.1, we observe that for any parametrized systems $\boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{\rho}$, ITMs $\mathcal{E}, \mathcal{A}$, and $k \in \mathbb{N}, z \in \{0,1\}^*$, we can construct two binary probability distributions ensembles:

$$(\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{A},\mathcal{E}}(k,z))_{k\in\mathbb{N},z\in\{0,1\}^*} \quad \text{and} \quad ((\boldsymbol{\phi},\infty,\boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{A},\mathcal{E}}(k,z))_{k\in\mathbb{N},z\in\{0,1\}^*}.$$

We want to use them to define a generalization of UC-emulation, adapted to $(\boldsymbol{\phi}, \infty, \boldsymbol{\pi})$-executions. We want this extension to retain desirable properties of UC-emulation, notably

[37] Here, we specifically refer to the external-write instruction issued by M, before any potential modification by the control function.

[38] Same as footnote 37.

transitivity, and the equivalence between UC-emulation and UC-emulation with respect to the dummy adversary [Can20, (Claim 11)].

To this end, we define $\mathcal{D}^{\text{DROP}}$, an adaptation of the dummy adversary $\mathcal{D}$ for (ϕ, ∞, π)-executions. When $\mathcal{D}^{\text{DROP}}$ receives a message $(\mathsf{ReplacedSession}, m)$, it behaves exactly like $\mathcal{D}$ on the message m. $\mathcal{D}^{\text{DROP}}$ is said to be "dropping", as defined below:

Definition 11 (Dropping Adversary)**.** An adversary $\mathcal{A}$ is said to be "dropping" if, upon receiving a message of the form $(\mathsf{ReplacedSession}, m)$ — with $\mathsf{ReplacedSession}$ a set of sids and m a message — it only uses m, thereby "dropping" $\mathsf{ReplacedSession}$.

In our extension of UC-emulation, we restrict attention to dropping adversaries in $(\boldsymbol{\phi}, \infty, \boldsymbol{\pi})$-executions. Otherwise, a generalization of [Can20, (Claim 11)] would fail.

Definition 12 (Extension of UC-emulation)**.** Let $\boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{\rho}, \boldsymbol{\upsilon}$ be four PPT parametrized systems, and let ξ be a predicate on extended identities. Suppose $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ is a PPT triplet.

We say that $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ UC-emulates $\boldsymbol{\upsilon}$ with respect to ξ-identity-bounded environments (or: $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ ξ-UC-emulates $\boldsymbol{\upsilon}$) if for any dropping PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{S}$ such that for any PPT ξ-identity-bounded environment $\mathcal{E}$, we have $(\boldsymbol{\phi}, \infty, \boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\boldsymbol{\upsilon},\mathcal{S},\mathcal{E}}$.

We say that $\boldsymbol{\upsilon}$ UC-emulates $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ with respect to ξ-identity-bounded environments (or: $\boldsymbol{\upsilon}$ ξ-UC-emulates $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$) if for any PPT adversary $\mathcal{A}$, there exists a dropping PPT adversary $\mathcal{S}$ such that for any PPT ξ-identity-bounded environment $\mathcal{E}$, we have $\mathsf{EXEC}_{\boldsymbol{\upsilon},\mathcal{A},\mathcal{E}} \approx (\boldsymbol{\phi}, \infty, \boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{S},\mathcal{E}}$.

The UC-emulation is said to be perfect if the two distribution ensembles are perfectly indistinguishable. If the relation only holds for a specific adversary $\mathcal{A}$, we say that UC-emulation is with respect to $\mathcal{A}$.

UC-emulation for a fixed predicate ξ is transitive. For example, if $\boldsymbol{\alpha}$ ξ-UC-emulates $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ and $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ ξ-UC-emulates $\boldsymbol{\upsilon}$, then $\boldsymbol{\alpha}$ ξ-UC-emulates $\boldsymbol{\upsilon}$.

We also generalize [Can20, (Claim 11)] to this setting:

Proposition 2 (UC-emulation with respect to $\mathcal{D}^{\text{DROP}}$ is equivalent to UC-emulation)**.** *Let $\boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{\rho}, \boldsymbol{\upsilon}$ be four PPT parametrized systems and ξ a predicate on extended identities. Suppose $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ is a PPT triplet. Then, $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ ξ-UC-emulates $\boldsymbol{\upsilon}$ if and only if $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ ξ-UC-emulates $\boldsymbol{\upsilon}$ with respect to $\mathcal{D}^{\text{DROP}}$.*

Proof. Direct adaptation of the proof of [Can20, (Claim 11)]. □

6.7.2 Definition of UC Operator.

We now define what we mean by a UC operator. Note that the conditions imposed on $\boldsymbol{\rho}$, $\boldsymbol{\phi}$, and $\boldsymbol{\pi}$ correspond to those required in our UC theorem.

Definition 13 (UC Operator)**.** Let $\mathsf{PPT\text{-}Param}$ be the set of PPT parametrized systems. Let $S \subset \mathsf{PPT\text{-}Param}^3$ and $\mathsf{UC\text{-}OP} : S \to \mathsf{PPT\text{-}Param}$ a function. We say that $\mathsf{UC\text{-}OP}$ is a S-Universal Composition Operator (or S-UC Operator) if, for any $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi}) \in S$ such that:

- $\boldsymbol{\pi}, \boldsymbol{\phi}$ are separated, sub-party respecting, sid-exposing, and disjoint,
- $\boldsymbol{\rho}$ is $(\boldsymbol{\pi}, \boldsymbol{\phi})$-compliant,
- $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ is a PPT triplet,

Then, $\mathsf{UC\text{-}OP}(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ perfectly UC-emulates $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$.

6.7.3 Description of our UC Operator Implementation.

As described in Issue (I.4) (page 24), the UC operator used in Plain UC suffers from flaws that lead to counterexamples to the UC theorem.[39]. In Appendix E, we describe a revised UC operator that meets our definition. Here, we summarize the core concept.

[39] An additional issue is discussed in Appendix D.1

In every extended session $\mathsf{Ext}(\rho^{\phi\to\pi}, \mathsf{s}_{\mathsf{test}})$, we introduce an ITI $\mathsf{Coord}\,(\mathsf{s}_{\mathsf{test}})$, referred to as the coordinator. This ITI acts as an active middle-man within the session, observing nearly all messages and modifying them as needed to ensure correct replacements. In particular, it records enough information to avoid the problem described in Issue (I.4).

7 Universal Composition Theorem

This section states the UC theorem of Updated UC. It is both more general and more modular than the UC Theorem of Plain UC [Can20, (Theorem 22)], as it does not rely on a particular UC Operator implementation. Any choice of a UC operator (Definition 13) – including the one provided in Appendix E – immediately yields a concrete instantiation of the theorem, analogous to the UC Theorem in Plain UC.

A complete proof is provided in Appendix C. Its overall structure follows that of the original proof in Plain UC, building on the changes introduced in Updated UC to address the issues identified in this work.

Theorem 1 (Universal Composition Theorem)**.** *Let $\boldsymbol{\pi}$, $\boldsymbol{\phi}$, and $\boldsymbol{\rho}$ be three PPT parametrized systems, and let ξ be a predicate on extended identities. Assume that $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ is a PPT triplet, and that $\boldsymbol{\phi}$ and $\boldsymbol{\pi}$ are sub-party respecting, separated, disjoint, and sid-exposing. Further assume that $\boldsymbol{\rho}$ is $(\boldsymbol{\pi}, \boldsymbol{\phi}, \xi)$-compliant and that $\boldsymbol{\pi}$ ξ-UC-emulates $\boldsymbol{\phi}$. Then, $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ UC-emulates $\boldsymbol{\rho}$.*

Sketch of proof. The complete proof of the theorem is given in the extended version [CS25, Appendix C]. Its blueprint follows closely the original proof [Can20, (Theorem 22)], but incorporates the changes introduced in Updated UC to address the issues identified in this work, and is more detailed. In this proof sketch, we briefly explain the main points of the argument.

Aim of the proof Using UC-emulation with respect to the dropping dummy adversary $\mathcal{D}^{\text{DROP}}$, we are looking for a PPT adversary $\mathcal{S}_\rho$ such that, for any PPT environment $\mathcal{E}$:

$$(\boldsymbol{\phi}, \infty, \boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho}, \mathcal{D}^{\text{DROP}}, \mathcal{E}} \approx \mathsf{EXEC}_{\boldsymbol{\rho}, \mathcal{S}_\rho, \mathcal{E}}\ . \tag{1}$$

The fact that $\boldsymbol{\pi}$ ξ-UC-emulates $\boldsymbol{\phi}$ guarantees that there exists a PPT adversary $\mathcal{S}_\phi$ of $\boldsymbol{\phi}$, such that for any ξ-identity-bounded, PPT environment $\mathcal{E}$:

$$\mathsf{EXEC}_{\boldsymbol{\pi}, \mathcal{D}, \mathcal{E}} \approx \mathsf{EXEC}_{\boldsymbol{\phi}, \mathcal{S}_\phi, \mathcal{E}}\ . \tag{2}$$

We construct a PPT adversary $\mathcal{S}_\rho$ from $\mathcal{S}_\phi$ and then show that it satisfies Equation 1.

Description of $\mathcal{S}_\rho$ The complete code of the adversary $\mathcal{S}_\rho$ is provided in the extended version [CS25].

Before giving a summarized description of $\mathcal{S}_\rho$, we note that it simulates multiple variants of $\mathcal{S}_\phi$, each slightly modified. For each $\mathsf{s} \in \{0,1\}^*$, the adversary $\mathcal{S}_{\phi,\mathsf{s}}$ behaves identically to $\mathcal{S}_\phi$, with one exception: when $\mathcal{S}_\phi$ attempts to send a backdoor to an ITI $M \notin \mathsf{Pan}(\phi, \mathsf{s})$, $\mathcal{S}_{\phi,\mathsf{s}}$ deactivates instead of sending the message.

The behavior of $\mathcal{S}_\rho$, represented in Figure 12, can be summarized as follows:

- Upon receiving an input (m, M) from the environment, $\mathcal{S}_\rho$ uses the sid-exposing property of both $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ in order to compute $\mathsf{positionM}_\pi = \mathsf{PSExpose}^{\boldsymbol{\pi}}(k, M)$, and $\mathsf{positionM}_\phi = \mathsf{PSExpose}^{\boldsymbol{\phi}}(k, M)$. It then proceeds as follows:
  - If $\mathsf{positionM}_\pi \in \{0,1\}^*$, then $\mathcal{S}_\rho$ uses the simulation of $\mathcal{S}_{\phi,\mathsf{positionM}}$ to process the input (m, M).

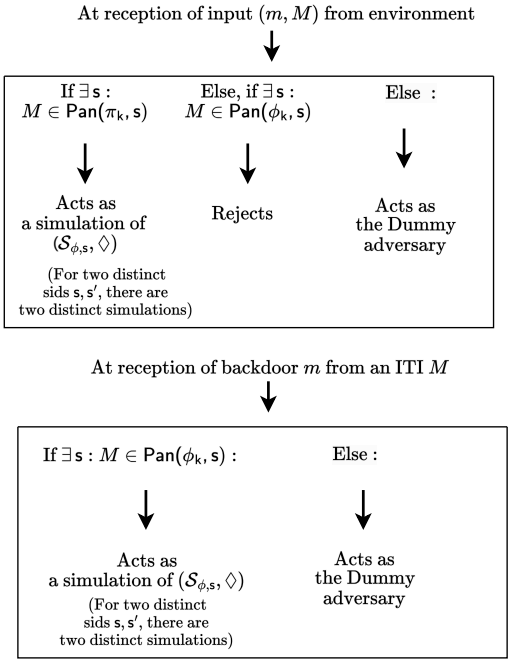


Figure 12: Simplified representation of the adversary $(\mathcal{S}_\rho, \Diamond)$ for security parameter k.

  - Else, if $\mathsf{positionM}_\phi \in \{0,1\}^*$, then $\mathcal{S}_\rho$ rejects.[40]
  - Otherwise, if $\mathsf{positionM}_\phi = \mathsf{positionM}_\pi = \perp$, then $\mathcal{S}_\rho$ acts as the dummy adversary and sends backdoor m to M.
- Upon receiving a backdoor m from some ITI M, $\mathcal{S}_\rho$ uses the sid-exposing property of $\boldsymbol{\phi}$ to compute $\mathsf{positionM}_\phi = \mathsf{PSExpose}^{\boldsymbol{\phi}}(k, M)$. It then proceeds as follows:
  - If $\mathsf{positionM}_\phi \in \{0,1\}^*$, then $\mathcal{S}_\rho$ uses the simulation of $\mathcal{S}_{\phi,\mathsf{positionM}}$ to process the backdoor m received from M.
  - Otherwise, $\mathcal{S}_\rho$ acts as the dummy adversary and sends the backdoor (m, M) to the environment.

Note that $\mathcal{S}_\rho$ is PPT because $\mathcal{S}_\phi$ is PPT, and both $\mathsf{Expose}^{\boldsymbol{\pi}}$ and $\mathsf{Expose}^{\boldsymbol{\phi}}$ are PPT Turing machines.

Proof of Equation (1) The proof proceeds by contraposition. We suppose that there exists a PPT environment $\mathcal{E}_\rho$ such that Equation (1) is false. By definition, this means: $\exists c, d \in \mathbb{N}$ such that $\forall k_0 \in \mathbb{N}, \exists k > k_0, \exists z \in \bigcup_{0<a\leq k^d}\{0,1\}^a$:

$$\Big|\Pr\big[(\boldsymbol{\phi}, \infty, \boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{D}^{\mathsf{DROP}},\mathcal{E}_\rho}(k,z) = 1\big] - \Pr\big[\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{S}_\rho,\mathcal{E}_\rho}(k,z) = 1\big]\Big| \geq k^{-c}\,. \qquad (3)$$

We then use this environment $\mathcal{E}_\rho$ to construct a ξ-identity-bounded PPT environment $\mathcal{E}_\phi$ for which Equation (2) fails. More precisely, we show in the extended version [CS25, Appendix C, Lemma 1] that $\mathcal{E}_\phi$ satisfies the following central result:

[40] This case does not appear in the simulator used in the proof of the UC Theorem of Plain UC [Can20, (Theorem 22)]. Its omission is the source of Weakness (W.5), as described in Section D.3, for readers familiar with the proof of the plain UC Theorem.

There is a polynomial Q increasing on $[0,\infty[$, with values in $]0,\infty[$, such that, for any $(k,z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d}\{0,1\}^a$:

$$\Big|\Pr\big[\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi}(k,z)=1\big]-\Pr\big[\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi}(k,z)=1\big]\Big| = \frac{1}{Q(k)}\left|\Pr\big[(\boldsymbol{\phi},\infty,\boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{D}^{\mathsf{DROP}},\mathcal{E}_\rho}(k,z)=1\big]-\Pr\big[\mathsf{EXEC}_{\rho,\mathcal{S}_\rho,\mathcal{E}_\rho}(k,z)=1\big]\right|.$$

Using this result together with Equation (3), we can conclude that Equation (2) is false. Indeed, they imply that $\forall k_0, \exists k > k_0, \exists z \in \bigcup_{0<a\leq k^d}\{0,1\}^a$ such that:

$$\Big|\Pr\big[\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi}(k,z)=1\big]-\Pr\big[\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi}(k,z)=1\big]\Big| \geq \frac{k^{-c}}{Q(k)}\,.$$

Which completes the proof of the UC Theorem.

Note that establishing this central result requires showing that multiple executions have the same output distribution. More precisely, we perform an in-depth analysis in the extended version [CS25, Appendix C, Lemmas 2 and 3] to show that for $(k,z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d}\{0,1\}^a$ and $\ell \in \{0,\cdots,Q(k)-2\}$:

$$\begin{aligned}
\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi^\ell}(k,z) &= \mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi^{\ell+1}}(k,z)\,,\\
\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi^{Q(k)-1}}(k,z) &= (\boldsymbol{\phi},\infty,\boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{D}^{\mathsf{DROP}},\mathcal{E}_\rho}(k,z)\,,\\
\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi^0}(k,z) &= \mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{S}_\rho,\mathcal{E}_\rho}(k,z)\,.
\end{aligned}$$

These equalities correspond to statements appearing in the proof of the UC Theorem in Plain UC, namely:

- Equation [Can20, (Equation (7), page 65)];
- The equivalences between inequalities [Can20, (Equation (4), page 64)] and [Can20, (Equation (6), page 65)], claimed on page 65 with the sentence "Consequently, inequality Equation (4) can be rewritten as".

These statements are asserted merely on the basis of an inspection of the involved ITMs. However, detailed demonstrations are not provided, and we believe that this omission substantially contributed to the counterexamples we identified. □

Corollary 1 (Universal Composition Theorem with a UC Operator)**.** *Let $S \subset \mathsf{PPT\text{-}Param}^3$ and let $\mathsf{UC\text{-}OP}$ be an S-UC operator. Let $\boldsymbol{\pi}$, $\boldsymbol{\phi}$, and $\boldsymbol{\rho}$ be parametrized systems such that $(\boldsymbol{\rho},\boldsymbol{\phi},\boldsymbol{\pi}) \in S$, and let ξ be a predicate on extended identities. Under the same assumptions as Theorem 1, $\mathsf{UC\text{-}OP}\,(\boldsymbol{\rho},\boldsymbol{\phi},\boldsymbol{\pi})$ UC-emulates $\boldsymbol{\rho}$.*

Proof. Immediate from transitivity of UC-emulation and Proposition 2. □

In typical use cases of Updated UC, the user aims to show that a protocol $\boldsymbol{\pi}$ UC-emulates another protocol $\boldsymbol{\phi}$, and wishes to preserve this security under composition. To that end, it suffices to verify that $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ are PPT, disjoint, separated, sid-exposing, and sub-party respecting. This ensures that the UC theorem applies to most natural choices of a PPT protocol $\boldsymbol{\rho}$, while the compliance and PPT-triplet requirements act as sanity conditions to exclude problematic edge cases.

References

[BC15] Olivier Blazy and Céline Chevalier. Generic construction of UC-secure oblivious transfer. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 2015: 13th International Conference on Applied Cryptography and Network Security*, volume 9092 of *Lecture Notes in Computer Science*, pages 65–86, New York, NY, USA, June 2–5, 2015. Springer, Cham, Switzerland. doi:10.1007/978-3-319-28166-7_4.

[BCH+20] Christian Badertscher, Ran Canetti, Julia Hesse, Björn Tackmann, and Vassilis Zikas. Universal composition with global subroutines: Capturing global setup within plain UC. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 1–30, Durham, NC, USA, November 16–19, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-64381-2_1.

[BDFH25] Ward Beullens, Lucas Dodgson, Sebastian H. Faller, and Julia Hesse. The 2Hash OPRF framework and efficient post-quantum instantiations. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part VIII*, volume 15608 of *Lecture Notes in Computer Science*, pages 332–362, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland. doi:10.1007/978-3-031-91101-9_12.

[BGHJ24] Manuel Barbosa, Kai Gellert, Julia Hesse, and Stanislaw Jarecki. Bare PAKE: Universally composable key exchange from just passwords. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part II*, volume 14921 of *Lecture Notes in Computer Science*, pages 183–217, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-68379-4_6.

[BMTZ24] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: A composable treatment. *Journal of Cryptology*, 37(2):18, April 2024. doi:10.1007/s00145-024-09493-7.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. doi:10.1109/SFCS.2001.959888.

[Can18] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, version 20181231:043505, 2018. https://eprint.iacr.org/archive/2000/067/20181231:043505.

[Can20] Ran Canetti. Universally Composable Security. *J. ACM*, 67(5):28:1–28:94, 2020. doi:10.1145/3402457.

[CCL15] Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 3–22, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-48000-7_1.

[CDG+25] Megan Chen, Pousali Dey, Chaya Ganesh, Pratyay Mukherjee, Pratik Sarkar, and Swagata Sasmal. Universally composable non-interactive zero-knowledge

from sigma protocols via a new straight-line compiler, 2025. doi:10.1007/978-3-031-91820-9.

[CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85, Amsterdam, The Netherlands, February 21–24, 2007. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-70936-7_4.

[CDT19] Jan Camenisch, Manu Drijvers, and Björn Tackmann. Multi-protocol UC and its use for building modular and efficient protocols. Cryptology ePrint Archive, Report 2019/065, 2019. URL: https://eprint.iacr.org/2019/065.

[CJSV22] Ran Canetti, Palak Jain, Marika Swanberg, and Mayank Varia. Universally composable end-to-end secure messaging. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 3–33, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-15979-4_1.

[CKKR19] Jan Camenisch, Stephan Krenn, Ralf Küsters, and Daniel Rausch. iUC: Flexible universal composability made simple. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 191–221, Kobe, Japan, December 8–12, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-34618-8_7.

[CNPR22] Cas Cremers, Moni Naor, Shahar Paz, and Eyal Ronen. CHIP and CRISP: Protecting all parties against compromise through identity-binding PAKEs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 668–698, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-15979-4_23.

[CR03] Ran Canetti and Tal Rabin. Universal composition with joint state. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 265–281, Santa Barbara, CA, USA, August 17–21, 2003. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-45146-4_16.

[CS25] Céline Chevalier and Éric Sageloli. Diving deep into UC: Uncovering and resolving issues in universal composability. Cryptology ePrint Archive, Report 2025/934, 2025. URL: https://eprint.iacr.org/2025/934.

[CS26] Céline Chevalier and Éric Sageloli. Diving deep into UC. *IACR Communications in Cryptology*, 3(1), 2026. doi:10.62056/a6n56chdj.

[DFG+23] Gareth T. Davies, Sebastian H. Faller, Kai Gellert, Tobias Handirk, Julia Hesse, Máté Horváth, and Tibor Jager. Security analysis of the WhatsApp end-to-end encrypted backup protocol. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 330–361, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-38551-3_11.

[Hes20] Julia Hesse. Separating symmetric and asymmetric password-authenticated key exchange. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20: 12th International Conference on Security in Communication Networks*, volume 12238 of *Lecture Notes in Computer Science*, pages 579–599, Amalfi, Italy, September 14–16, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-57990-6_29.

[HR25] Julia Hesse and Michael Rosenberg. PAKE combiners and efficient post-quantum instantiations. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part II*, volume 15602 of *Lecture Notes in Computer Science*, pages 395–420, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland. doi:10.1007/978-3-031-91124-8_14.

[HS15] Dennis Hofheinz and Victor Shoup. GNUC: A new universal composability framework. *Journal of Cryptology*, 28(3):423–508, July 2015. doi:10.1007/s00145-013-9160-y.

[HSS23] Julia Hesse, Nitin Singh, and Alessandro Sorniotti. How to bind anonymous credentials to humans. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023: 32nd USENIX Security Symposium*, pages 3047–3064, Anaheim, CA, USA, August 9–11, 2023. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/hesse.

[HUM09] Dennis Hofheinz, Dominique Unruh, and Jörn Müller-Quade. Polynomial runtime and composability. Cryptology ePrint Archive, Report 2009/023, 2009. URL: https://eprint.iacr.org/2009/023.

[JKX18] Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 456–486, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-78372-7_15.

[KTR20] Ralf Küsters, Max Tuengerthal, and Daniel Rausch. The IITM model: A simple and expressive model for universal composability. *Journal of Cryptology*, 33(4):1461–1584, October 2020. doi:10.1007/s00145-020-09352-1.

[LL25] You Lyu and Shengli Liu. Hybrid password authentication key exchange in the UC framework. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part II*, volume 15602 of *Lecture Notes in Computer Science*, pages 421–450, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland. doi:10.1007/978-3-031-91124-8_15.

[Mau12] Ueli Maurer. Constructive cryptography – a new paradigm for security definitions and proofs. In Sebastian Mödersheim and Catuscia Palamidessi, editors, *Theory of Security and Applications*, pages 33–56, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. doi:10.1007/978-3-642-27375-9_3.

[RKC22] Daniel Rausch, Ralf Küsters, and Céline Chevalier. Embedding the UC model into the IITM model. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 242–272, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-07085-3_9.

A Quick Guide to Appendices

In addition to this guide, the appendices contain five sections that can be read in any order. They remain independent except for shared conventions and notations:

- Notations and conventions necessary for Appendices C and E are detailed in Section C.1.
- For Appendices B and D, relevant notations and conventions are provided in Section B.1.

Let's briefly explain each appendix:

- Appendix B contains complete counterexamples and proofs for Issues (I.2), (I.3), and (I.4) discussed in the paper.
- Appendix C gives a detailed proof of the UC Theorem. A detailed description of its subsections is given at the beginning of the appendix.
- Appendix D presents further discussion on subtleties of Plain UC and additional issues, limitations, and weaknesses. In particular, it explains the rationale for avoiding predicates to designate receivers, using an alternative time-complexity notion, and omitting the balanced environment concept. A more detailed presentation is given at the beginning of the appendix.
- Appendix E details a construction of a UC operator that satisfies Definition 13.

Finally, the complete table of contents appears at the end of the document.

B Details About Issues, Limitations and Weaknesses Described in the Main Body

This section provides complete counterexamples and detailed proofs for Issue (I.2), Issue (I.3), and Issue (I.4) introduced in the main body.

Preliminary notations and conventions relevant to these constructions are given in Appendix B.1.

B.1 Additional Notations and Conventions

Additional Notations for ITIs.

Let SetITI denote the set of all ITIs.

For an ITI M, we denote its code by $\mathsf{code}(M)$ and its identifier by $\mathsf{id}(M)$. If this identifier is a pair $(\mathsf{s}, \mathsf{p}) \in \{0,1\}^* \times \{0,1\}^*$, we write $\mathsf{sid}(M)$ for its sid and $\mathsf{pid}(M)$ for its pid.

For any external-write w, $\mathsf{mess}(w)$ denotes its message and $\mathsf{tape}(w)$ its destination tape.

Additional Conventions and Instructions for Algorithms.

Local Functions. Each local function has access to the local variables of the main function and may modify them.

Sending Messages. The instructions **apply** and **send** both indicate that the ITI performs an external-write instruction w. More precisely, **apply** is used in the form "**apply** w", and **send** appears as "**send** 'm' to tape t of ITI N with flags (...)". If the context is clear, flags may be omitted. When the destination tape t is known, we often use **send input** , **send output** , or **send backdoor** .

Deactivation, Reject, and Halt. The instruction **deactivate** deactivates the ITI, **reject** causes it to reject, and **halt** places it in a halt state. Once halted, the ITI deactivates and rejects any subsequent messages.

Parsing. The instruction "**checkorchalt** m is (...)" checks whether a message m matches the specified pattern. If so, execution proceeds; otherwise, the ITI halts. This instruction is typically used to parse m as a tuple and halt if parsing fails. The instruction **checkorreject** performs the same check but rejects instead of halting.

Ending Execution. In environment algorithms, for $b \in \{0,1\}$, the instruction "**return** b" causes $\mathcal{E}$ to write b to the outgoing message tape and halt. This instruction thus terminates the execution with result b. The result of an execution with environment $\mathcal{E}$ is sometimes referred to as the "result of $(\mathcal{E},0)$".

Default Behavior. If an ITM's algorithm omits a response to certain events (*e.g.* it specifies a reaction to inputs but not to outputs or backdoors), the ITM is assumed to reject by default.

Use of the New Extended Session Definition. Due to the limitation highlighted in (L.1) (Section 5.1), we adopt our own extended session definition (Definition 3).

No Use of Parametrized Systems. The counterexamples we construct are not parametrized systems, but rather ITMs.

Although UC-emulation is formally defined only for parametrized systems, the notion naturally extends to ITMs. For any ITMs π and ϕ, we define $\pi_k = \pi$ and $\phi_k = \phi$ for all k, and apply the UC-emulation definition to the parametrized systems $\boldsymbol{\pi} = (\pi_k)_k$ and $\boldsymbol{\phi} = (\phi_k)_k$.

Simplification of the Subroutine Exposing Property. Certain protocols ϕ we will construct require the subroutine exposing property. In every such case, the exposing ITI for the extended session $\mathsf{Ext}(\phi,\mathsf{s})$ will be $(\phi,(\mathsf{s},0))$, and each ITI $(\phi,(\mathsf{s},\mathsf{p}))$ with $\mathsf{p} \neq 0$ will halt upon its first activation.

In our counterexamples, these latter ITIs are not monitored by the exposing ITI, as implementing such functionality would compromise readability and inflate the size of the algorithm.

Nevertheless, this feature could be added without affecting any relevant properties: an ITI $(\phi,(\mathsf{s},\mathsf{p}))$ may notify $(\phi,(\mathsf{s},0))$ of its invocation by sending it an output. This difference has no impact on any of the properties considered in our counterexamples.

Use of Flags. Unless explicitly stated otherwise, the following conventions apply:

- Messages exchanged between protocol ITIs (*i.e.* any ITI except the adversary and environment) have both flags forced-write and reveal-ext-id set.
- Backdoors sent by the adversary have both forced-write and reveal-ext-id unset.
- Inputs sent by the environment have both flags forced-write and reveal-ext-id set.

Use of Autoreferences. For clarity, we permit the code of an ITI to include its own code—despite this being formally undefined.

This abstraction simplifies the frequent situation involving two ITIs, say (ψ,id) and (α,id'), such that:

- (ψ,id) sends a message to (α,id'),
- and (α,id') sends a message back to (ψ,id).

In this scenario, the code of ψ would embed that of α, which in turn embeds the code of ψ, introducing a circular definition.

To circumvent this, we rely on the fact that an ITI has access to its own code via its identity tape, as defined in [Can20, (Section 3.1.1)]. We implement the following workaround in ψ:

- Replace every occurrence of ψ in its own code with a reserved sequence seq.

- Modify ψ's code so that, at runtime, the ITI replaces seq with the code stored on its identity tape.

Structured Protocols' sid and pid Match Their Body.

For structured protocols such as the UC operator $\rho^{\phi \to \pi}$, we require their sid and pid to to match those of their body.

This convention is not respected in Plain UC, as pointed out in Issue (I.5) (see Appendix D.1.1), and leads to a counterexample of the UC Theorem.

Compliance. We omit the full definition of compliance from the main body of the article; it is reproduced here for completeness.

Definition 14 (Compliance, ([Can20, (Section 6.1)])). A protocol ρ is called (π, ϕ, ξ)-compliant if:

- All inputs sent by any ITI of any extended session of ρ are made with the flag $\mathsf{forced\text{-}write}$ set. All outputs sent to any ITI of any extended session of ρ have the flag $\mathsf{reveal\text{-}ext\text{-}id}$ set.
- No two external-write instructions, out of all the external-write instructions made by the members of an extended session of ρ, where one instruction has target code π, and the other instruction has target code ϕ, have the same target sid.
- The extended identities of all the ITIs in any extended session of ρ, that pass inputs to ITIs with code either π or ϕ, satisfy the predicate ξ.

When $\xi = \xi_{\mathsf{true}}$, the always-true predicate, we say that the protocol is (π, ϕ)-compliant.

B.1.1 (L.1): Detailed proof that the shell code [Can20, (Section 5.2)] does not turn any protocol into a subroutine respecting one

Consider two ITIs $M = (\pi, (s, p))$ and $N = (\nu, (s, p))$, where the only hypothesis we make is the following:

(H) When M is invoked with an input m_1, M sends an input m_2 to N.

We denote by σ_π the shell construction defined in [Can20, (Section 5.2)], paragraph "Writing subroutine respecting protocols".

We will show that the structured protocol (π, σ_π), with body π and shell σ_π, is not subroutine respecting.

First, we introduce some notations for clarity: For any protocol μ, we denote by (μ, σ_π) the structured protocol with body μ and shell σ_π. We adopt the convention that the sid of such a structured protocol is a pair $(s_{\mathsf{body}}, s_{\mathsf{shell}})$.

We also define:

- M_{σ_π} as the structured ITI with shell σ_π, body π, shell sid $\top$, body sid s, and body pid p;[41]
- N_{σ_π} as the structured ITI with shell σ_π, body ν, shell sid $\top$, body sid s, and body pid p.

We now proceed to prove that (π, σ_π) is not subroutine respecting. To this end, we describe an execution of a protocol ρ with an environment $\mathcal{E}$ and some adversary, in which one of the subroutine-respecting conditions is violated.

For clarity and to keep the explanation concise, we define the actions of $\mathcal{E}$ and ρ as the execution progresses.[42]

1. At first activation, $(\mathcal{E}, 0)$ sends an input to $(\rho, (s, p))$.
2. $(\rho, (s, p))$ then sends an input to N_{σ_π}.

[41] We do not specify the value of the shell pid because we could not find its definition in the shell specification. We believe it can be set to $\top$, as is done for the shell sid.

[42] If needed, we would be happy to provide their code.

3. This triggers the first activation of N_{σ_π}. By definition of the shell σ_π [Can20, Section 5.2], Condition 1.(c) applies: N_{σ_π} halts and remains inactive for all subsequent activations. In particular, *Because N_{σ_π} has been invoked by an ITI that do not belong to an extended session of (π, σ), N_{σ_π} is outside of the extended session* $((\pi, \sigma_\pi), (\mathsf{s}, \top))$.
4. As N_{σ_π} has halted, the environment $(\mathcal{E}, 0)$ is activated again. It sends another input to $(\rho, (s, p))$.
5. $(\rho, (s, p))$ is activated for the second time. We define its behavior so that it sends an input m_1 to M_{σ_π}.
6. This is the first activation of M_{σ_π}. According to the shell specification, Condition 1.(a) applies, setting the main-party flag. Then, Condition 2 applies: since the flag main-party is set and the message is an input, the message m_1 is directly transferred to the body M.
7. By hypothesis **(H)**, the body M intends to send an input m_2 to N. As a result, according to the last paragraph of Condition 3 of the shell specification, M_{σ_π} produces an instruction to deliver input m_2 to N_{σ_π}. *This contradicts the subroutine-property of (π, σ_π), as M_{σ_π}, a main party of the extended session $((\pi, \sigma_\pi), (s, \top))$, sent an input to N_{σ_π}, an ITI that is outside the extended session* $((\pi, \sigma_\pi), (s, \top))$.[43]

B.2 Details Regarding Issue (I.2)

In this section, we present a comprehensive counterexample for Issue (I.2).

Figure 13 defines the protocols ϕ, π, and α – adaptations of those introduced in the discussion of Issue (I.2) (Section 5.1) – modified to satisfy the conditions required for the application of the UC theorem. These protocols rely on an additional protocol $\rho \notin \{\pi, \phi, \alpha\}$, whose definition is given in Figure 14.

Proposition 3. *ϕ and π are subroutine respecting.*

Proof. It is clear that π is subroutine respecting. We therefore focus on ϕ, and fix an execution of an arbitrary protocol (not necessarily π) with any adversary and environment. For a sid $\mathsf{s} \neq 0$, the extended session $\mathsf{Ext}(\phi, \mathsf{s})$ contains no sub-party, so we focus on $(\phi, 0)$. The extended session $\mathsf{Ext}(\phi, 0)$ can have one sub-party $(\alpha, (0, 0))$, depending on which ITI sends first an input to $(\alpha, (0, 0))$. More precisely, there are two cases:

- Case 1: if $(\alpha, (0, 0))$ is activated by an input from $(\phi, (0, 0))$, it is a sub-party of $(\phi, 0)$. We thus needs to ensure that:
  - $(\alpha, (0, 0))$ does not send inputs or outputs to ITIs outside the extended session $\mathsf{Ext}(\phi, 0)$. This is indeed the case, as $(\alpha, (0, 0))$ only sends an input to $(\phi, (0, 0))$ then halts.
  - $(\alpha, (0, 0))$ rejects any inputs or outputs from ITIs outside the extended session $\mathsf{Ext}(\phi, 0)$. This is indeed the case, as $(\alpha, (0, 0))$ halts after its first activation.
- Case 2: if $(\alpha, (0, 0))$ is activated by an input from an ITI that is not $(\phi, (0, 0))$, we need to ensure that $(\alpha, (0, 0))$ does not send outputs to main parties of the extended session $\mathsf{Ext}(\phi, 0)$, or input to the sub-parties of the extended session $\mathsf{Ext}(\phi, 0)$. This is the case, as it only sends an input to $(\phi, (0, 0))$ and then halts.

□

Proposition 4. *ϕ and π are subroutine exposing.*

Proof. It is direct by construction, the exposing ITI of session (π, s) (*resp.* (ϕ, s)) being $(\pi, (s, 0))$ (*resp.* $(\phi, (\mathsf{s}, 0))$). □

Proposition 5. *π UC-emulates ϕ.*

[43] More precisely, this contradicts the condition 4 of the subroutine respecting definition [Can20, (Definition 19)].

$(\phi, (\mathsf{s}, \mathsf{p}))$

local variable: boolean $\mathsf{alphaIsSubparty}$, initialized to false
boolean $\mathsf{FirstInputMade}$, initialized to false
1 : **if** $(\mathsf{s}, \mathsf{p}) \neq (0,0) \wedge \neg\mathsf{IsBackdoor}$ **then**
2 : **reject**
3 : At reception of an input m from an ITI M :
4 : **if** $\mathsf{FirstInputMade}$ is false **then**
5 : $\mathsf{FirstInputMade} \leftarrow \mathsf{true}$
6 : **if** $M \neq (\alpha, (0,0))$ $\wedge$ first bit of randomness is 0 **then**
7 : $\mathsf{alphaIsSubparty} \leftarrow$ true
8 : **send input** "0" to $(\alpha, (0,0))$
9 : **else**
10 : **send output** "0" to $(\rho, (0,0))$
11 : **send output** "0" to $(\rho, (0,0))$
12 : At reception of an output m from an ITI M :
13 : **if** $M = (\alpha, (0,0))$ **then**
14 : **send output** "0" to $(\rho, (0,0))$
15 : **else reject**
16 : At reception of a backdoor :
// subroutine exposing interface:
17 : **if** $\mathsf{p} \neq 0$ **then reject**
18 : **checkorreject** message can be parsed as an ITI M
19 : **if** $M = (\phi, (\mathsf{s}, \mathsf{p}))$ **then**
20 : **send backdoor** "true" to adversary
21 : **if** $(\mathsf{s}, \mathsf{p}) = (0,0) \wedge M = (\alpha, (0,0))$ **then**
22 : **send backdoor** "alphaIsSubparty" to adversary
23 : **send backdoor** "false" to adversary

$(\pi, (\mathsf{s}, \mathsf{p}))$

1 : **if** $(\mathsf{s}, \mathsf{p}) \neq (0,0) \wedge \neg\mathsf{IsBackdoor}$ **then**
2 : **reject**
3 : At reception of an input m from an ITI M :
4 : **send output** "0" to $(\rho, (0,0))$
5 : **reject**
6 : At reception of output : **reject**
7 : At reception of a backdoor :
8 : // subroutine exposing interface:
9 : **if** $\mathsf{p} \neq 0$ **then reject**
10 : **checkorreject** message can be parsed as an ITI M
11 : **if** $M = (\phi, (\mathsf{s}, \mathsf{p}))$ **then**
12 : **send backdoor** "true" to adversary
13 : **send backdoor** "false" to adversary

$(\alpha, (\mathsf{s}, \mathsf{p}))$

1 : **local variable:** boolean $\mathsf{FirstInputMade}$, initialized to false
2 : **if** $(\mathsf{s}, \mathsf{p}) \neq (0,0)$ **then reject**
3 : At reception of an input m from an ITI M :
4 : **if** $\mathsf{FirstInputMade}$ is false **then**
5 : $\mathsf{FirstInputMade} \leftarrow \mathsf{true}$
6 : **if** $M = (\phi, (0,0))$ **then**
7 : **send output** "0" to $(\phi, (0,0))$
8 : **else**
9 : **send input** "0" to $(\phi, (0,0))$
10 : **else reject**
11 : At reception of output : **reject**
12 : At reception of backdoor : **reject**

Figure 13: Definition of ϕ, π and α for Issue (I.2).

Proof. We consider an execution of π with dummy adversary $\mathcal{D}$ and an environment $\mathcal{E}$. We note $\mathsf{s}_{\mathsf{test}}$ the test sid of the execution. The proof is made in two steps.
Step 1: We can assume that $\mathcal{D}$ neither receives nor sends any input or backdoor.

Note that the only interactions between the protocols ϕ, π, and their adversaries occur through requests to the exposing ITI. We define an adversary $\mathcal{S}$ for ϕ by modifying $\mathcal{D}$ only in the way it handles messages related to the subroutine exposing property:

- The code of the exposing ITI $(\pi, (\mathsf{s}_{\mathsf{test}}, 0))$ is replaced by ϕ, both for messages from the exposing ITI to the adversary and from the adversary to the exposing ITI.
- Requests about the presence of $(\pi, (\mathsf{s}_{\mathsf{test}}, 0))$ in the session are replaced by requests about the presence of $(\phi, (\mathsf{s}_{\mathsf{test}}, 0))$.
- Requests about the presence of an ITI M with code α are not sent to the exposing ITI of $(\phi, (\mathsf{s}_{\mathsf{test}}, 0))$; instead, $\mathcal{S}$ simulates the response as if the request were made and the exposing ITI returned false.

It is immediate that $\mathcal{S}$ perfectly simulates the queries made by $\mathcal{D}$ to the exposing ITI $(\pi, (\mathsf{s}_{\mathsf{test}}, 0))$.

Furthermore, the environment $\mathcal{E}$ does not need to send any message to the dummy adversary in order to learn which ITIs belong to $\mathsf{Ext}(\phi, \mathsf{s}_{\mathsf{test}})$: the existence of these ITIs is entirely determined by the input messages sent by $\mathcal{E}$.

We have therefore established that:

- $\mathcal{E}$ cannot exploit any behavioral difference between $\mathcal{D}$ and $\mathcal{S}$ to distinguish the execution of π from that of ϕ.
- $\mathcal{E}$ does not need to interact with $\mathcal{D}$, as it already knows all the information that $\mathcal{D}$ could provide.

Consequently, we may assume without loss of generality that $\mathcal{D}$ never receives or sends any input or backdoor.
Step 2: $\mathcal{E}$ cannot distinguish the execution of π or ϕ by sending inputs.

The only possible observable behavioral difference between $(\phi, (0, 0))$ and $(\pi, (0, 0))$ from the environment's perspective arises from the fact that $(\alpha, (0, 0))$ may belong to the extended session $\mathsf{Ext}(\phi, 0)$, but never belongs to $\mathsf{Ext}(\pi, 0)$.

In particular, in the execution of ϕ with test sid 0, the behavior of $(\alpha, (0, 0))$ upon receiving an input depends on whether it was previously invoked by $(\phi, (0, 0))$.

However, for this behavioral difference to be visible to the environment, it would have to send an input to $(\alpha, (0, 0))$. This is impossible, since the environment is restricted to sending inputs only to main parties. □

Proposition 6. *The protocol ρ defined in Figure 14 is (π, ϕ)-compliant.*

Proof. All conditions can directly be verified. □

We have shown that ϕ and π are subroutine respecting, subroutine exposing, that π UC-emulates ϕ and that ρ is (π, ϕ)-compliant. The UC theorem of Plain UC should thus apply and ensure that $\rho^{\phi \to \pi}$ UC-emulates ρ.

Nevertheless, this conclusion is false, as shown by the next proposition.

Proposition 7. *$\rho^{\phi \to \pi}$ does not UC-emulate ρ.*

Proof. The environment $\widetilde{\mathcal{E}}$ defined in Figure 14 can distinguish them. More precisely, in the execution of ρ with environment $\widetilde{\mathcal{E}}$, any adversary and any input, $\widetilde{\mathcal{E}}$ will return a uniformly random bit, whereas it will always return 1 in the execution of $\rho^{\phi \to \pi}$ with environment $\widetilde{\mathcal{E}}$ any adversary and any input.

Detail of messages sent in the execution of ρ with environment $\widetilde{\mathcal{E}}$, any adversary and any input:

- $\widetilde{\mathcal{E}}$ sends input "0" to $(\rho, (0, 0))$.

N is an ITI with code $\notin \{\phi, \pi, \alpha, \rho\}$

$(\widetilde{\mathcal{E}}, 0)$

1 : At first activation :
2 : **send input** "0" to $(\rho, (0,0))$
3 : At activation with no message :
// This condition occurs if:
// - An ITI deactivates itself.
// - The adversary sends a message
// to a non-existent ITI.
// - The control function disallows
// an external write.
4 : **return** 0
5 : At reception of an output from an ITI M :
6 : **return** 1

(ρ, id)

1 : **if** $\mathsf{id} \neq (0,0)$ **then halt**
2 : At reception of an input :
3 : **if** it is first activation **then**
4 : **send input** "0" to $(\phi, (0,0))$
5 : **else halt**
6 : At reception of an output from an ITI M :
7 : **if** it is second activation **then**
8 : **send input** "0" to $(\alpha, (0,0))$
9 : **else if** it is third activation **then**
10 : **send output** "0" to N
11 : **else halt**

Figure 14: Definition of ρ and $\widetilde{\mathcal{E}}$ for Issue (I.2).

- $(\rho, (0,0))$ sends input "0" to $(\phi, (0,0))$.
- If the first bit of randomness of $(\phi, (0,0))$ is 0:
  1. $(\phi, (0,0))$ sends input "0" to $(\alpha, (0,0))$.
  2. $(\alpha, (0,0))$ sends output "0" to $(\phi, (0,0))$.
  3. $(\phi, (0,0))$ sends output "0" to $(\rho, (0,0))$.
  4. $(\rho, (0,0))$ sends input "0" to $(\alpha, (0,0))$.
  5. Because α halted, it deactivates.
  6. $\widetilde{\mathcal{E}}$ is activated with no message and therefore returns 0.
- If the first bit of randomness of $(\phi, (0,0)$ is 1:
  1. $(\phi, (0,0))$ sends output "0" to $(\rho, (0,0))$.
  2. $(\rho, (0,0))$ sends input "0" to $(\alpha, (0,0))$.
  3. It is the first activation of α by an ITI that is not $(\phi, (0,0))$, it sends input "0" to $(\phi, (0,0))$.
  4. $(\phi, (0,0))$ sends output "0" to $(\rho, (0,0))$.
  5. $(\rho, (0,0))$ sends output "0" to an ITI N that does belong to the extended session $\mathsf{Ext}(\rho, 0)$. The output is thus sent to $\widetilde{\mathcal{E}}$, as well as extended identity N and identity $(0,0)$.
  6. $\widetilde{\mathcal{E}}$ returns 1.

Detail of messages sent in the execution of $\rho^{\phi \to \pi} = (\rho, \sigma_{\pi,\phi})$ with environment $\widetilde{\mathcal{E}}$, any adversary and any input:

- $\widetilde{\mathcal{E}}$ sends input "0" to $((\rho, \sigma_{\pi,\phi}), (0,0))$.
- $((\rho, \sigma_{\pi,\phi}), (0,0))$ sends input with message "$(0, (\texttt{Shell:main:}\ \rho, 0\texttt{;}\ \texttt{subr:}\ \pi, 0))$" to the ITI $((\pi, \sigma_{\pi,\phi}), (0,0))$.
- $((\pi, \sigma_{\pi,\phi}), (0,0))$ sends output with message "$(0, (\texttt{Shell:main:}\ \rho, 0\texttt{;}\ \texttt{subr:}\ \pi, 0))$" to the ITI $((\rho, \sigma_{\pi,\phi}), (0,0))$.

- $((\rho, \sigma_{\pi,\phi}), (0,0))$ sends input with message "$(\mathsf{0}, (\texttt{Shell:main:}\ \rho, 0\texttt{;}\ \texttt{subr:}\ \perp))$" to the ITI $((\alpha, \sigma_{\pi,\phi}), (0,0))$.
- $((\alpha, \sigma_{\pi,\phi}), (0,0))$ sends input with message "$(\mathsf{0}, (\texttt{Shell:main:}\ \rho, 0\texttt{;}\ \texttt{subr:}\ \pi, 0))$" to the ITI $((\pi, \sigma_{\pi,\phi}), (0,0))$.
- $((\pi, \sigma_{\pi,\phi}), (0,0))$ sends output with message "$(\mathsf{0}, (\texttt{Shell:main:}\ \rho, 0\texttt{;}\ \texttt{subr:}\ \pi, 0))$" to the ITI $((\rho, \sigma_{\pi,\phi}), (0,0))$.
- $((\rho, \sigma_{\pi,\phi}), (0,0))$ wants to send output with message "$\mathsf{0}$" to an ITI N that does belong to the extended session $\mathsf{Ext}(\rho, \sigma_{\pi,\phi}, 0)$. The output is thus sent to $\widetilde{\mathcal{E}}$, as well as extended identity N and identity $(0,0)$.
- $\widetilde{\mathcal{E}}$ returns 1.

□

B.3 Details Regarding Issue (I.3)

In this section, we present a comprehensive counterexample to issue (I.3).

We consider the protocols ϕ and π defined in Figure 15, the auxiliary protocols α and β defined in Figure 16, and the protocol ρ defined in Figure 17.

Proposition 8. *ϕ and π are subroutine respecting.*

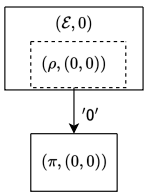
Proof. In any execution of an arbitrary protocol, the only extended session of ϕ that may contain a sub-party is $(\phi, 0)$, with sub-party $(\beta, (0,0))$. By definition, $(\beta, (0,0))$ rejects any message not originating from $(\phi, (0,0))$. It follows that ϕ is subroutine respecting.

We now focus on π. In any execution of an arbitrary protocol, the only session of π that can contain a sub-party is $(\pi, 0)$, with sub-party $(\alpha, (0,0))$. All other main parties $(\pi, (0, \mathsf{s}))$, $\mathsf{s} \neq 0$, only accept backdoor and never send any input or output, so we can focus on the study of $(\pi, (0,0))$ and $(\alpha, (0,0))$.

Let θ be an execution prefix – of an arbitrary protocol with an environment $\mathcal{E}$ and some adversary – such that $(\alpha, (0,0))$ is a sub-party of $(\pi, 0)$. We will show that, within θ, $(\alpha, (0,0))$ never receives an input from any ITI other than $(\pi, (0,0))$. This will complete the proof.

If $(\alpha, (0,0))$ is a sub-party of $(\pi, 0)$, then the condition given in Line 5 of π's definition must hold. In particular, this implies that $(\pi, (0,0))$ exists in θ and received an input "$\mathsf{0}$" from some ITI with extended identity $(\rho, (0,0))$.

However, by construction, ρ never sends inputs to $(\pi, (0,0))$. Therefore, this situation can only arise if θ is a prefix of an execution in which the environment $(\mathcal{E}, 0)$ impersonates $(\rho, (0,0))$ to send the input "$\mathsf{0}$" to $(\pi, (0,0))$:



In this situation, θ is therefore a prefix of an execution of the protocol π with test sid 0. By the rules governing protocol executions, $(\alpha, (0,0))$ can only receive inputs from $(\pi, (0,0))$, since the environment $(\mathcal{E}, 0)$ is not permitted to send inputs to the sub-party $(\alpha, (0,0))$. This completes the proof. □

Proposition 9. *ϕ and π are subroutine exposing.*

Proof. It is direct by construction, the exposing ITI of session (π, s) (*resp.* (ϕ, s)) being $(\pi, (s, 0))$ (*resp.* $(\phi, (\mathsf{s}, 0))$). □

Proposition 10. *π UC-emulates ϕ.*

$(\pi, (\mathsf{s}, \mathsf{p}))$		$(\phi, (\mathsf{s}, \mathsf{p}))$
	local variable: boolean alphaIsSubparty, initialized to false integer cpt, initialized to 0	**local variable:** boolean betaIsSubparty, initialized to false integer cpt, initialized to 0
1 :	**if** $(\mathsf{s}, \mathsf{p}) \neq (0,0) \wedge \neg$IsBackdoor **then**	**if** $(\mathsf{s}, \mathsf{p}) \neq (0,0) \wedge \neg$IsBackdoor **then**
2 :	**reject**	**reject**
3 :	At reception of an input or output :	At reception of an input or output :
4 :	**if** cpt $= 0$ **then**	**if** cpt $= 0$ **then**
5 :	**if** it is input "0" from $(\rho, (0,0))$ **then**	**if** it is input "0" from $(\rho, (0,0))$ **then**
6 :	alphaIsSubparty $\leftarrow$ true	betaIsSubparty $\leftarrow$ true
7 :	cpt $\leftarrow 1$	cpt $\leftarrow 1$
8 :	**send input** "0" to $(\alpha, (0,0))$	**send input** "0" to $(\beta, (0,0))$
9 :	**else reject**	**else reject**
10 :	**elseif** cpt $= 1$ **then**	**elseif** cpt $= 1$ **then**
11 :	**if** it is output "0" from $(\alpha, (0,0))$ **then**	**if** it is output "0" from $(\beta, (0,0))$ **then**
12 :	cpt $\leftarrow 2$	cpt $\leftarrow 2$
13 :	**send output** "0" to $(\rho, (0,0))$	**send output** "0" to $(\rho, (0,0))$
14 :	**else reject**	**else reject**
15 :	**elseif** cpt ≥ 2 **then**	**elseif** cpt ≥ 2 **then**
16 :	**reject**	**reject**
17 :	At reception of a backdoor :	At reception of a backdoor :
	// subroutine exposing interface:	// subroutine exposing interface:
18 :	**if** p $\neq 0$ **then reject**	**if** p $\neq 0$ **then reject**
19 :	**checkorreject** message can be parsed as an ITI M	**checkorreject** message can be parsed as an ITI M
20 :	**if** $M = (\pi, (\mathsf{s}, \mathsf{p}))$ **then**	**if** $M = (\phi, (\mathsf{s}, \mathsf{p}))$ **then**
21 :	**send backdoor** "true" to adversary	**send backdoor** "true" to adversary
22 :	**if** $(\mathsf{s}, \mathsf{p}) = (0,0) \wedge M = (\alpha, (0,0))$ **then**	**if** $(\mathsf{s}, \mathsf{p}) = (0,0) \wedge M = (\beta, (0,0))$ **then**
23 :	**send backdoor** "alphaIsSubparty" to adversary	**send backdoor** "betaIsSubparty" to adversary
24 :	**send backdoor** "false" to adversary	**send backdoor** "false" to adversary

Figure 15: Definition of ϕ, π for Issue (I.3).

(α, id)

1 : **local variable:** integer cpt, initialized to 0
2 : **if** id $\neq (0,0)$ **then reject**
3 : At reception of an input :
4 : **if** cpt $= 0$ **then**
5 : **if** if it is input "0" from $(\pi, (0,0))$ **then**
6 : cpt $\leftarrow 1$
7 : **send output** "0" to $(\pi, (0,0))$
8 : **else reject**
9 : **elseif** cpt $= 1$ **then**
10 : **if** it is input "0" from $(\rho, (0,0))$ **then**
11 : cpt $\leftarrow 2$
12 : **send output** "0" to $(\rho, (0,0))$
13 : **elseif** cpt ≥ 2 **then reject**
14 : At reception of an output or backdoor :
15 : **reject**

(β, id)

local variable: integer cpt, initialized to 0
if id $\neq (0,0)$ **then reject**
At reception of an input :
if cpt $= 0$ **then**
if it is input "0" from $(\phi, (0,0))$ **then**
cpt $\leftarrow 1$
send output "0" to $(\phi, (0,0))$
else reject
elseif cpt ≥ 1 **then reject**
At reception of an output or backdoor :
reject

Figure 16: Definition of α, β for Issue (I.3).

(ρ, id)

1 : At first activation :
2 : **if** id $\neq (0,0)$ **then halt**
3 : **send input** "0" to $(\phi, (0,0))$
4 : At second activation :
5 : **if** it is ouput "0" from $(\phi, (0,0))$ **then**
6 : **send input** "0" to $(\alpha, (0,0))$
7 : **else reject**
8 : At third activation :
9 : **if** it is ouput "0" from $(\alpha, (0,0))$ **then**
// μ is a code distinct of ρ, π, ϕ
10 : **send output** "0" to $(\mu, (0,0))$
11 : **else reject**
12 : For other activations:
13 : **reject**

Figure 17: Definition of ρ for Issue (I.3). All messages are sent with flags forced-write and reveal-ext-id set.

Proof. We consider an execution of π with dummy adversary $\mathcal{D}$ and an environment $\mathcal{E}$. We note $\mathsf{s}_{\mathsf{test}}$ the test sid of the execution.

The proof is made in two steps.

Step 1: We can assume that $\mathcal{D}$ neither receives nor sends any input or backdoor.

Observe that the only interactions between the protocols ϕ, π and their adversaries occur through requests to the exposing ITI. We construct an adversary $\mathcal{S}$ for ϕ as a variant of $\mathcal{D}$, with the only difference being that, whenever $\mathcal{D}$ sends or receives a message related to the subroutine exposing property, $\mathcal{S}$ modifies the message as follows:

- The code of the exposing ITI $(\pi, (\mathsf{s}_{\mathsf{test}}, 0))$ is replaced with ϕ, both for messages from the exposing ITI to the adversary and from the adversary to the exposing ITI.
- Requests about the presence of $(\alpha, (0, 0))$ in the session are replaced by requests about the presence of $(\beta, (0, 0))$.

It is immediate that $\mathcal{S}$ perfectly simulates the requests made by $\mathcal{D}$ to the exposing ITI $(\pi, (\mathsf{s}_{\mathsf{test}}, 0))$, since the subroutine $(\alpha, (0, 0))$ is invoked in the execution of ϕ under exactly the same conditions as $(\beta, (0, 0))$ in the execution of π.

Moreover, the environment $\mathcal{E}$ does not need to interact with the dummy adversary to determine which ITIs belong to $\mathsf{Ext}(\phi, \mathsf{s}_{\mathsf{test}})$, as their existence depends solely on the input messages previously sent by $\mathcal{E}$.

We have therefore shown that:

- $\mathcal{E}$ cannot exploit any behavioral difference between $\mathcal{D}$ and $\mathcal{S}$ to distinguish the execution of π from that of ϕ.
- $\mathcal{E}$ does not need to communicate with $\mathcal{D}$, as it already knows all the information $\mathcal{D}$ could provide.

It follows that we may assume, without loss of generality, that $\mathcal{D}$ neither receives nor sends any input or backdoor.

Step 2: $(\mathcal{E}, 0)$ cannot distinguish the execution of π or ϕ by sending inputs.

The only behavioral difference that $(\mathcal{E}, 0)$ could potentially exploit lies in the second activation of the sub-parties: $(\beta, (0, 0))$ deactivates, whereas $(\alpha, (0, 0))$ sends an output to $(\rho, (0, 0))$ if it receives the input "0" from $(\rho, (0, 0))$. However, by construction, $(\rho, (0, 0))$ never sends such a message. Moreover, the environment's impersonation ability cannot be leveraged in this context, as it is not allowed to send inputs to sub-parties. □

Proposition 11. *The protocol ρ defined in Figure 17 is (π, ϕ)-compliant.*

Proof. It is direct, as ρ never writes to an ITI with code π and that the flag forced-write is always set. □

We have shown that ϕ and π are subroutine respecting, subroutine exposing, that π UC-emulates ϕ and that ρ is (π, ϕ)-compliant. The UC theorem of Plain UC should thus apply and ensure that $\rho^{\phi \to \pi}$ UC-emulates ρ.

Nevertheless, this conclusion is false, as shown by the next proposition.

Proposition 12. *$\rho^{\phi \to \pi}$ does not UC-emulate ρ.*

Proof. Let us consider this environment:

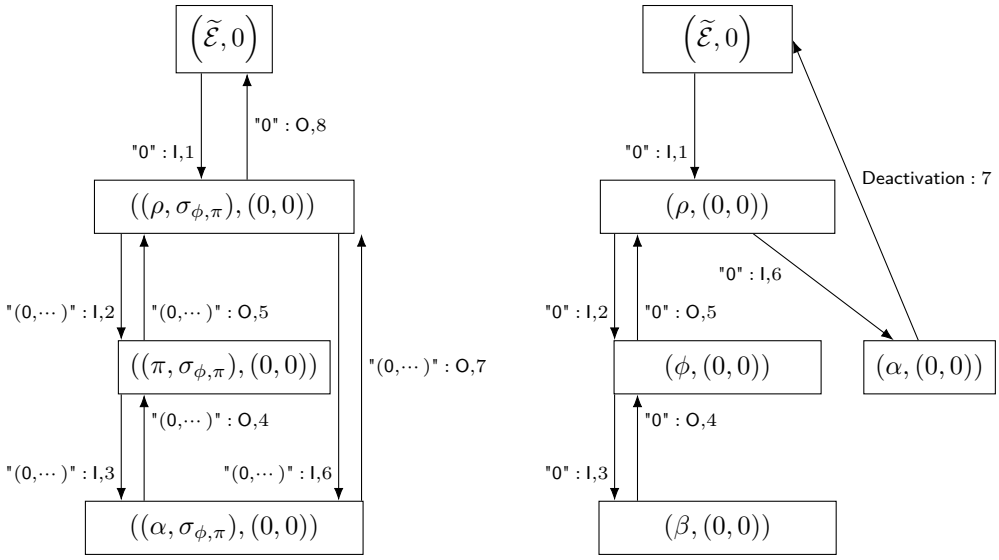


Figure 18: I/O differentiate input from output, and the numbers indicate execution order. The left diagram shows the message sequence in the execution of $\rho^{\phi \to \pi}$ with $\widetilde{\mathcal{E}}$, and the omitted right parts of messages indicated by "$\cdots$" are the shell messages $\mathsf{m}_{\mathsf{shell}}$ as discussed in the first paragraph of Section 5.2. The right diagram shows the sequence of messages in the execution of ρ with $\widetilde{\mathcal{E}}$. The adversary is not used in both executions and is therefore not shown.

$(\widetilde{\mathcal{E}}, 0)$

At first activation :

send input "0" to the protocol with identifier $(0,0)$

At second activation :

if it is activated with no message **then**

// This condition occurs if:

// - An ITI deactivates itself.

// - The adversary sends a message to a non-existent ITI.

// - The control function disallows an external write.

return 1

else return 0

We can see in the Figure 18 that for any adversary, $\widetilde{\mathcal{E}}$ returns 0 for the execution of $\rho^{\phi \to \pi}$ and 1 for the execution of ρ.

□

B.4 Details Regarding Issue (I.4)

In this section, we present a comprehensive counterexample to issue (I.4).

We consider the protocol ρ defined in Figure 19, which is constructed using an auxiliary protocol α.

Additionally, we consider any PPT protocols ϕ and π satisfying the following:

- $\phi \notin \{\pi, \rho, \alpha\}$,

(ρ, id)		(α, id)
1 :	At first activation:	1 : **reject**
2 :	**if** $\mathsf{id} \notin \{(0,0),(0,1)\}$ **then reject**	
3 :	**if** $\mathsf{id} = (0,0)$ **then**	
4 :	**send input** "0" to $(\alpha,(0,0))$	
5 :	**if** $\mathsf{id} = (0,1)$ **then**	
6 :	**send output** "0" to $(\alpha,(0,0))$	
7 :	For other activations:	
8 :	**reject**	

Figure 19: Algorithm ρ used for Issue (I.4).

- ϕ and π are subroutine respecting and subroutine exposing,
- π UC-emulates ϕ.

It is clear that ρ is (π, ϕ)-compliant.

According to the UC theorem in Plain UC, this should imply that $\rho^{\phi \to \pi}$ UC-emulates ρ. However, we will show that this is not the case: $\rho^{\phi \to \pi}$ does not UC-emulate ρ.

Proposition 13. *$\rho^{\phi \to \pi}$ does not UC-emulate ρ.*

Proof. We recall that the UC operator of Plain UC is defined in [Can20, (Section 6.1)]. Considering the environment $\widetilde{\mathcal{E}}$ defined in Figure 20. We can see that, for any adversary $\mathcal{A}$:

- In the execution of $\rho^{\phi \to \pi} = (\rho, \sigma_{\pi,\phi})$ with $\widetilde{\mathcal{E}}$ and the dummy adversary $\mathcal{D}$, using the definition of the replacement operator [Can20, (Section 6.1)], we see:
  - At reception of input 0, $\big((\rho, \sigma_{\pi,\phi}),(0,0)\big)$ sends output "$(0, (\texttt{Shell:main:}\ \rho, 0\texttt{;}\ \texttt{subr:}\ \bot)))$" to $((\alpha, \sigma_{\pi,\phi}),(0,0))$.
  - At reception of this input $((\alpha, \sigma_{\pi,\phi}),(0,0))$ deactivates.
  - $(\widetilde{\mathcal{E}}, 0)$ is then activated, and sends input 0 to $((\rho, \sigma_{\pi,\phi}),(0,1))$.
  - At reception of the input, $((\rho, \sigma_{\pi,\phi}),(0,1))$ initializes its variables main to $(\rho, 0)$ and subr to $\bot$. The body $(\rho,(0,1))$ is then activated with input 0 and intends to send output 0 to $(\alpha,(0,0))$: the condition 2.(c) of the definition of UC operator [Can20, (Section 6.1)] is then verified, which implies that $((\rho, \sigma_{\pi,\phi}),(0,1))$ sends output 0 to $(\alpha,(0,0))$. Because $(\alpha,(0,0))$ does not exist in the extended session $((\rho, \sigma_{\pi,\phi}), 0)$, this message is actually sent to $(\widetilde{\mathcal{E}}, 0)$, as well as the extended identity $(\alpha,(0,0))$ and the identity $(0,1)$.
  - $(\widetilde{\mathcal{E}}, 0)$ receives the output, and thus returns 1.
- In the execution of ρ with $\widetilde{\mathcal{E}}$ and $\mathcal{A}$:
  - At reception of input 0, $(\rho,(0,0))$ sends input 0 to $(\alpha,(0,0))$.
  - At reception of this input, $(\alpha,(0,0))$ deactivates.
  - $\widetilde{\mathcal{E}}$ is then activated, and sends input 0 to $(\rho,(0,1))$.
  - At reception of the input, $(\rho,(0,1))$ sends output 0 to $(\alpha,(0,0))$.
  - At reception this output, $(\alpha,(0,0))$ deactivates.
  - As a result of this deactivation, $\widetilde{\mathcal{E}}$ is activated with no message and returns 0.

□

$\left(\widetilde{\mathcal{E}}, 0\right)$

1 : At first activation:
2 : **send input** "0" to $(\rho, (0,0))$
3 : At second activation:
4 : **send input** "0" to $(\rho, (0,1))$
5 : At third activation:
6 : **if** it is activated with no message **then**
// This condition occurs if:
// - An ITI deactivates itself.
// - The adversary sends a message to a non-existent ITI.
// - The control function disallows an external write.
return 0
7 : **else return** 1

Figure 20: environment $\widetilde{\mathcal{E}}$ used for Issue (I.4).

B.5 Details Regarding Weakness (W.1)

In thus subsection, we explain precisely the problem (W.1) and then describes how it can be mitigated.

Detailed explanation of the problem.
We denote by $\mathcal{S}_\pi$ the simulator corresponding to the dummy adversary in the UC-emulation of ϕ by π, and by $\mathcal{S}$ the simulator constructed in the proof of the UC Theorem [Can20, Figure 10] to ensure that $\rho^{\phi\to\pi}$ UC-emulates ρ.

We are concerned with the following indistinguishability from [Can20, Equation (2) of Section 6.2.1]:

$$\mathsf{EXEC}_{\rho^{\phi\to\pi},\mathcal{D},\mathcal{E}} \approx \mathsf{EXEC}_{\rho,\mathcal{S},\mathcal{E}}. \tag{4}$$

Our goal is to explain that, with the current definition of $\mathcal{S}$, this indistinguishability does not hold for all PPT environments $\mathcal{E}$.

For an ITI M, we define $\zeta(M)$ as the predicate that holds if and only if the code of M corresponds to a structured protocol with the same shell as $\rho^{\phi\to\pi}$ (*i.e.*, the shell $\sigma_{\rho,\pi,\phi}$, following the precise definition of the UC Operator [Can20, (Section 6.1)]).

We make the following assumptions:

(H1) No execution of ρ contains an ITI M such that $\zeta(M)$ holds.

(H2) In the execution of any arbitrary protocol, no ITI M satisfying $\zeta(M)$ belongs to an extended session of π or ϕ.

Both assumptions are quite natural and hold for a large class of protocols ρ, ϕ, and π.

Let m be any input, M_{T} any ITI such that $\zeta(M_{\mathsf{T}})$ holds and that is neither an adversary nor an environment (that is, its identifier is neither 0 nor $\Diamond$), and M_{F} any ITI such that $\zeta(M_{\mathsf{F}})$ does not hold and that is neither an adversary nor the environment.

We will temporarily admit the following two claims, which will be proven at the end of this subsection:

- Claim 1: In any execution of $\rho^{\phi\to\pi}$ with dummy adversary $\mathcal{D}$ and some environment $\mathcal{E}$, when $(\mathcal{E},0)$ sends the input (m, M_{F}) to the adversary $(\mathcal{D},\Diamond)$, the next activation of $(\mathcal{E},0)$ is an activation with no message, resulting from a disallow by the control function.
- Claim 2: In any execution of ρ with simulator $\mathcal{S}$ and some environment $\mathcal{E}$, when $(\mathcal{E},0)$ sends the input (m, M_{T}) to the adversary $(\mathcal{S},\Diamond)$, the next activation of $(\mathcal{E},0)$ is an activation with no message, resulting from a disallow by the control function.

We now consider $\mathcal{E}$ a PPT environment such that Equation (4) is verified. Since, for every ITI M, either $\zeta(M)$ holds or it does not, we observe that when $(\mathcal{E},0)$ sends an input (m,M) to the adversary, the environment $(\mathcal{E},0)$ is next activated with no message either in the left-hand execution or the right-hand execution of Equation (4). This follows from the fact that the conditions of either Claim 1 or Claim 2 are satisfied.

A direct consequence of that, because of the indistinguishability of (4), is that in both executions of (4), the following holds except with negligible probability:

(C) Whenever $(\mathcal{E},0)$ sends an input (m,M) to the adversary, where M is neither an adversary or an environment, the next activation of $(\mathcal{E},0)$ will be with no message.

Indeed, if this behavior occurs non-negligibly in one of the two executions but not in the other, it would be straightforward to construct an environment $\mathcal{E}'$, based on $\mathcal{E}$, that can non-negligibly distinguish between the two executions.

Nevertheless, **(C)** is clearly false: we can straightforwardly construct protocols ρ,π,ϕ and PPT environments $\mathcal{E}$ that does not satisfy it.[44] This invalidates the proof of UC theorem with the construction of simulator $\mathcal{S}$ given in [Can20, Figure 10].

We now proceed to prove Claim 1 and Claim 2.

Proof. Proof of Claim 1:
We note that in any execution of $\rho^{\phi\to\pi}$, by definition of $\rho^{\phi\to\pi}$ and of protocol execution, all ITIs M distinct from the environment and adversary that exist in the execution satisfy ζ. In particular, M_{F} does not exist in the execution.

As a result, when $\mathcal{E}$ sends input (m, M_{F}) to the adversary $(\mathcal{D},\Diamond)$:

1. $(\mathcal{D},\Diamond)$ sends the backdoor message m to M_{F}.
2. As established at the beginning of this proof, M_{F} does not exist in the execution. The control function therefore disallows this backdoor, as it was sent with the `forced-write` flag unset.
3. As a consequence of the disallow, $(\mathcal{E},0)$ is activated with no message.

This completes the proof of Claim 1. □

Proof. Proof of Claim 2:
We note that by hypothesis **H1**, in any execution of ρ, all ITIs M distinct from the environment and adversary that exist in the execution do not satisfy ζ. In particular, M_{T} does not exist in the execution.

As a result, when $\mathcal{E}$ sends input (m, M_{T}) to the adversary $(\mathcal{S},\Diamond)$:

1. $(\mathcal{S},\Diamond)$ is activated, and Item 1.(a) of [Can20, Figure 10] is applied: each existing copy of $\mathcal{S}_\pi$ is activated to check whether M_{T} belongs to its database.
2. By hypothesis **H2**, this is not the case, so no copy of $\mathcal{S}_\pi$ responds positively[45].Consequently, the actions described in Item 1.(b) of [Can20, Figure 10] are not executed.
3. Again by hypothesis **H2**, M_{T} is not a main party of a session of π. Thus, Item 1.(c).(i) of [Can20, Figure 10] does not apply. As a consequence, Item 1.(c).(ii) of [Can20, Figure 10] apply and $(\mathcal{S},\Diamond)$ sends the backdoor message m to M_{T}.

[44] Said otherwise, our reasoning shows that any such choice of ρ, π, ϕ, and $\mathcal{E}$ would constitute a concrete counter-example to Equation (4).

[45] A positive response could, in principle, occur with negligible probability if the UC simulation of ϕ by π is not perfect. We omit this detail here, as it is not relevant to the argument.

4. As established at the beginning of this proof, M_{T} does not exist in the execution. The control function therefore disallows this backdoor, as it was sent with the `forced-write` flag unset.
5. As a consequence of the disallow, $(\mathcal{E}, 0)$ is activated with no message.

This completes the proof of Claim 2. □

Mitigation of the problem. The definition of the simulator $\mathcal{S}$ must be adjusted as follows. Upon receiving an input (m, M), the simulator $\mathcal{S}$ should reject if M does not satisfy $\zeta(M)$ (that is, if the code of M does not correspond to a structured protocol sharing the same shell as $\rho^{\phi\to\pi}$). If M satisfies $\zeta(M)$, it represents a structured protocol whose body is an ITI denoted M_{body}. In this case, Items 1.(a), 1.(b), and 1.(c) of [Can20, Figure 10] must be applied to M_{body} rather than to M.

B.6 Many examples of protocols studied in the literature that are not subroutine exposing

In this appendix, we substantiate the claim made on page 8 of Section 2 that most protocols used in the UC literature are not subroutine exposing, *i.e.*, they do not satisfy the subroutine-exposing property without additional modifications.[46]

To this end, we show that the ideal protocols associated with the following ideal functionalities are not subroutine exposing.[47]

- $\mathcal{F}_{\mathsf{AUTH}}$ [Can20, Figure 12], $\mathcal{F}^{l}_{\mathsf{SMT}}$ [Can20, Figure 13], $\mathcal{F}_{\mathsf{SYN}}$ [Can20, Figure 14] $\mathcal{F}_{\mathsf{NC}}$ [Can20, Figure 15].
- $\mathcal{F}_{\mathsf{PAKE}}, \mathcal{F}_{\mathsf{lePAKE}}, \mathcal{F}_{\mathsf{rlePAKE}}$ [HR25, Figure 1] (Eurocrypt 2024).
- $\mathcal{F}_{\mathsf{bPAKE}}$ and $\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$ [LL25, Figure 5] (Eurocrypt 2025).
- $\mathcal{F}^{f}_{\mathsf{SFE}}$, $\mathcal{F}_{\mathsf{OPRF}}$ and $\mathcal{F}^{f}_{\mathsf{2H-OPRF}}$ [BDFH25, Figures 1 and 2] (Eurocrypt 2025).
- $\mathcal{F}_{\mathsf{cbAC}}$ and $\mathcal{F}^{\mathsf{blind}}_{\mathsf{cbAC}}$ [HSS23, Figure 4] (USENIX Security 2023).

We begin by recalling the complete definition of a subroutine exposing protocol as introduced in [Can20, Definition 21]:

> "Protocol π is subroutine exposing if each session s of π, occurring within an execution of any protocol with any environment, provides an interface to the adversary where the adversary can specify an extended identity α and is notified in response whether α is a member of the extended session (π, s). The interface should take the form of a query on the backdoor tape of an ITI whose extended identity should be determined given (π, s) alone."

As stated, this definition requires the existence of an ITI U—called an *exposing ITI*—that can receive queries on its backdoor tape. Each query contains an extended identity α, and upon receiving such a query, U must send a backdoor message to the adversary indicating whether α belongs to the extended session (π, s).

We now explain why, for each ideal functionality $\mathcal{F}$ listed earlier in the short answer, the corresponding ideal protocol $\mathsf{IDEAL}_{\mathcal{F}}$ is not subroutine exposing. Specifically, we show that no ITI appearing in an execution of $\mathsf{IDEAL}_{\mathcal{F}}$ satisfies the role of an exposing ITI.

By definition of the ideal protocol $\mathsf{IDEAL}_{\mathcal{F}}$ [Can20, Section 5.3], the ITIs that may appear in an extended session $(\mathsf{IDEAL}_{\mathcal{F}}, s)$ are of two types:

- The main parties of $(\mathsf{IDEAL}_{\mathcal{F}}, s)$, which act as simple relays between the environment and the ideal functionality $\mathcal{F}$. More precisely, they relay messages between the

[46] We do not claim that such modifications are impossible; for instance, a generic transformation achieving this is described in [Can20, Section 6.1].

[47] We focus on ideal protocols because they play a central role in the UC framework. The selected functionalities come from [Can20] as well as from several works accepted at major conferences. The cited papers do not necessarily represent the earliest appearances of these functionalities.

environment and the ITI $(\mathcal{F}, (s, \bot))$. They cannot serve as exposing ITIs, as [Can20, Section 5.3, Item (3)] explicitly specifies that they reject any backdoor message.
- The ideal functionality ITI $(\mathcal{F}, (s, \bot))$ itself and any subroutines it may invoke. In all the ideal functionalities we listed earlier, $(\mathcal{F}, (s, \bot))$ does not invoke subroutines. Consequently, it suffices to check whether $(\mathcal{F}, (s, \bot))$ could serve as an exposing ITI. A direct inspection of the corresponding code for each functionality shows that $(\mathcal{F}, (s, \bot))$ is not an exposing ITI: $(\mathcal{F}, (s, \bot))$ never implements an interface allowing the adversary to query, for example, whether a ITI of the form $(\mathsf{IDEAL}_{\mathcal{F}}, (s, p))$ exists in the extended session $(\mathsf{IDEAL}_{\mathcal{F}}, s)$.

C Complete Proof of UC Theorem of Updated UC (Theorem 1)

The aim of this appendix is to provide the complete proof of Theorem 1. The initial appendices introduce necessary notations and conventions (C.1), present more formal definitions of time-complexity (C.2), and establish preliminary results regarding relevant execution prefixes (C.3). The final appendices (C.4, C.5, C.6, C.7 and C.8) are fully dedicated to proving the theorem.

As discussed in Section 7, this proof follows the steps of the original UC theorem for plain UC, leveraging the adjustments in Updated UC to address previously identified issues.

Consequently, most steps in this proof will resemble those of [Can20, (Theorem 22)]. However, we emphasize that, besides the changes introduced by our new definitions, we add significant new contributions.

First, our proof is more modular. It's not only independent of a specific choice of UC-operator but also structured in a modular way. The theorem is proved using a central lemma (Appendix C.4), which itself relies on two additional lemmas (C.5), respectively proven in Appendices C.7 and C.8. This breakdown of the proof simplifies its analysis.

Secondly, the proofs of Lemmas 2 and 3 represent significant new contributions. In the proof of [Can20, (Theorem 22)], the equivalents of these lemmas[48,49] are asserted through the inspection of the involved ITMs. However, detailed demonstrations are not provided, which we believe substantially contributes to the counterexamples we identified concerning the plain UC theorem.

Third, we made some necessary changes beyond those required by the new definitions of Updated UC. Notably, we implemented a significant modification of the simulator, justified by the weaknesses (W.5) described in Appendix D.3.

C.1 Additional Notations and Conventions

Additional Notations for ITIs.

Let SetITI denote the set of all ITIs.

For an ITI M, we denote its code by $\mathsf{code}(M)$ and its identifier by $\mathsf{id}(M)$. If the identifier is of the form $(\mathsf{s}, \mathsf{p}) \in \{0,1\}^* \times \{0,1\}^*$, we write $\mathsf{sid}(M)$ for its sid and $\mathsf{pid}(M)$ for its pid.

For any external-write w, we denote its message by $\mathsf{mess}(w)$ and its destination tape by $\mathsf{tape}(w)$.

[48] Lemma 2 corresponds to Equality [Can20, (Equation (7), page 68)].

[49] Lemma 3 corresponds to the equivalence of inequalities [Can20, (Equation (4), page 67)] and [Can20, (Equation (6), page 68)].

Additional Conventions and Instructions for Algorithms.
Local Functions. Each local function has access to the local variables of the main function and may modify them.

Sending Messages. The instructions **apply** and **send** both indicate that the ITI performs an external-write instruction w. Specifically, **apply** appears as "**apply** w", while **send** is used in the form "**send** 'm' to tape t of ITI N with flags (...)". If the context permits, flags may be omitted. When the tape t is known, the shorthand notations **send input** , **send output** , or **send backdoor** are used.

Deactivation, Reject, and Halt. The instruction **deactivate** deactivates the ITI, **reject** causes it to reject, and **halt** places it in a halt state. Upon entering the halt state, the ITI deactivates and rejects any subsequent messages.

Parsing. The instruction "**checkorchalt** m is (...)" checks whether the message m matches the specified pattern. If it does, execution continues; otherwise, the ITI halts. This instruction is typically used to parse m as a tuple and halt if parsing fails. The variant **checkorreject** performs the same check, but rejects instead of halting.

Simulating. The instruction "**initialize** a simulation of U" means that the ITI M creates all necessary internal state to simulate the ITI U. After each step of simulated execution, M saves U's updated state and tape contents so that the simulation may resume on future activations. The simulation may involve:

- Simulating the application of external-write instructions to U via "**simulate** ".
- Simulating the deactivation or rejection of U via "**simulate** ".
- Reacting to external-write instructions or deactivations produced by the simulation of U.

Ending Execution. In environment algorithms, for $b \in \{0, 1\}$, the instruction "**return** b" causes the environment $\mathcal{E}$ to write b to its outgoing message tape and halt. This marks the termination of the execution with result b. We may refer to this result as the "result of $(\mathcal{E}, 0)$".

Default Behavior. If an ITM's algorithm does not specify behavior for certain events (e.g., it defines reactions to inputs but not to outputs or backdoors), the ITM is assumed to reject by default.

Use of Flags. Unless explicitly stated otherwise, the following conventions apply:

- Any message exchanged between protocol ITIs (that is, ITIs other than the adversary or the environment) has both $\mathsf{forced\text{-}write}$ and $\mathsf{reveal\text{-}ext\text{-}id}$ set.
- Any backdoor sent by the adversary has both $\mathsf{forced\text{-}write}$ and $\mathsf{reveal\text{-}ext\text{-}id}$ unset.
- Any input sent by the environment has both $\mathsf{forced\text{-}write}$ and $\mathsf{reveal\text{-}ext\text{-}id}$ set.

Use of Autoreferences. For clarity, we permit the code of an ITI to reference its own code, even though this is formally disallowed.

This convention is especially useful when analyzing scenarios involving two ITIs (ψ, id) and (α, id') that exchange messages in both directions:

- (ψ, id) sends a message to (α, id'),
- and (α, id') responds by sending a message to (ψ, id).

This mutual reference implies that the code of ψ includes that of α, which itself includes ψ, forming an undefined circular dependency.

To resolve this, we use the identity tape feature introduced in [Can20, (Section 3.1.1)]. The protocol ψ is modified as follows:

- All occurrences of ψ in its own code are replaced by a special sequence seq.
- At runtime, the ITI replaces seq with the code read from its identity tape.

C.2 Detailed Definitions of Time Complexity

Definition 15 (PPT environment)**.** Let $\mathcal{E}$ an ITM. For $(k, z) \in \mathbb{N} \times \{0,1\}^*$, $B \in \mathbb{N}^*$ and a finite sequence of external-write operations $W = (w_i)_{i=1}^B$ with recipient $(\mathcal{E}, 0)$, we define the (k, z, W)-execution of $\mathcal{E}$ as follows:

- It begins by executing $(\mathcal{E}, 0)$ with input (k, z).
- If $\mathcal{E}$ halts, the (k, z, W)-execution ends.
- If $\mathcal{E}$ deactivates (due to self-deactivation or to an external-write operation), the next external operation of $(w_i)_{i=1}^B$ is given to $\mathcal{E}$. If there is no next external operation, the (k, z, W)-execution ends.

We say that $\mathcal{E}$ is a PPT environment if there exists a polynomial P such that, for any $(k, z) \in \mathbb{N} \times \{0,1\}^*$, $B \in \mathbb{N}$ and any finite sequence of external-write operations $W = (w_i)_{i=1}^B$ with recipient $(\mathcal{E}, 0)$, the (k, z, W)-execution of $\mathcal{E}$ has an execution time bounded by $P(k + |z|)$.

Definition 16 (PPT adversary)**.** Let $\mathcal{A}$ an ITM. For $k \in \mathbb{N}, B \in \mathbb{N}^*$ and a finite sequence of external-write operations $W = (w_i)_{i=1}^B$ with recipient $(\mathcal{A}, \Diamond)$, we define the (k, W)-execution of $\mathcal{A}$ as follows:

- It begins by executing $(\mathcal{A}, \Diamond)$ with external-write operation w_1 and writing k in the security parameter tape of $\mathcal{A}$.
- When $\mathcal{A}$ deactivates (due to self-deactivation or to an external-write operation), the next external operation of $(w_i)_{i=1}^B$ is given to $\mathcal{E}$. If there is no next external operation, the (k, W)-execution ends.

We say that $\mathcal{A}$ is a PPT adversary if there exists a polynomial P such that, for any $k \in \mathbb{N}$, $B \in \mathbb{N}^*$ and any finite sequence of external-write operations $W = (w_i)_{i=1}^B$ with recipient $(\mathcal{A}, \Diamond)$, the (k, W)-execution of $\mathcal{A}$ has an execution time bounded by $\sum_{i=1}^B Q(k + |w_i|)$.

Definition 17 (PPT protocol)**.** A parametrized system $\boldsymbol{\rho}$ is a PPT protocol if for any PPT environment $\mathcal{E}$ and PPT adversary $\mathcal{A}$, there is a polynomial $R = R(\mathcal{E}, \mathcal{A})$ such that for any $(k, z) \in \mathbb{N} \times \{0,1\}^*$: any execution of $\boldsymbol{\rho}$ with adversary $\mathcal{A}$, environment $\mathcal{E}$ and input (k, z) has an execution time bounded by $R(k + |z|)$.

Definition 18 (Triplet of PPT parametrized systems)**.** $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ is a triplet of PPT parametrized systems if for any PPT environment $\mathcal{E}$ and PPT adversary $\mathcal{A}$, there is a polynomial $R = R(\mathcal{E}, \mathcal{A})$ such that for any $(k, z) \in \mathbb{N} \times \{0,1\}^*$ and $l \in \mathbb{N}$: any $(\boldsymbol{\phi}, l, \boldsymbol{\pi})$-execution of $\boldsymbol{\rho}$ with adversary $\mathcal{A}$, environment $\mathcal{E}$ and input (k, z) has an execution time bounded by $R(k + |z|)$.

C.3 Relevant Prefixes

In the definition of sub-party respecting protocols (Definition 9), we formulate several conditions that must hold for every prefix of any extended execution.

However, when proving the UC Theorem (Theorem 7), only a small subset of these prefixes is actually involved. We refer to them as "relevant", and we define them precisely in Definition 19.

We next highlight three useful properties of relevant prefixes that play a crucial role in the proof of the UC Theorem: Propositions 14, 15, and 16.

More precisely:

- Proposition 14 essentially states that, in a α-relevant prefix θ and under appropriate hypotheses, any awake ITI $M \in \mathsf{Pan}(\alpha, \mathsf{s})$ belongs to $\mathsf{Ext}(\alpha, \mathsf{s}, \theta)$. This result may be seen as a weak form of inclusion of the panoptic session in the extended session.

- Proposition 15 explains that, in a α-relevant prefix θ and under appropriate hypotheses, the external-writes exchanged between ITIs in $\mathsf{Pan}(\alpha, \mathsf{s})$ are not subject to the

additional rules of replacement execution (*i.e.*, replacements of the form $\alpha \to \beta$ or $\beta \to \alpha$). In other words, when we restrict our attention to the ITIs in $\mathsf{Pan}(\alpha, \mathsf{s})$, their behavior appears as in a standard protocol execution, even if the prefix belongs to a replacement execution.

- Proposition 16 states that, in a α-relevant prefix θ and under the right assumptions, we have $\mathsf{Ext}(\alpha, \mathsf{s}, \theta) \subset \mathsf{Pan}(\alpha, \mathsf{s})$.

Definition 19 (Relevant execution prefix)**.** Let α a protocol. An α-relevant execution prefix refer to either:
- A ρ-execution prefix for $\rho \notin \mathsf{Pan}(\alpha)$.
- A $(\rho, \alpha, \ell, \beta)$-execution prefix or $(\rho, \beta, \ell, \alpha)$-execution prefix for a separated ITM β disjoint with α, $\ell \in \mathbb{N}$ and $\rho \notin \mathsf{Pan}(\alpha) \cup \mathsf{Pan}(\beta)$.

Proposition 14 (Awake ITIs of the Panoptic and Extended Sessions)**.** *Let α a separated and sub-party respecting protocol. For any α-relevant execution prefix θ and any sid $\mathsf{s} \in \{0,1\}^*$, there is no ITI $M \in \mathsf{Pan}(\alpha, \mathsf{s})$ which is awake on θ but does not belong to $\mathsf{Ext}(\alpha, \mathsf{s}, \theta)$.*

Proof. Suppose, for contradiction, that there exists an awake ITI in θ that belongs to $\mathsf{Pan}(\alpha, \mathsf{s})$ but not to $\mathsf{Ext}(\alpha, \mathsf{s}, \theta)$. Let M be the first such ITI, and let w be the external-write operation whose sender identity is N and which wakes M.[50]

By definition of panoptic and extended sessions,

$$\mathsf{MainParty}(\alpha, \mathsf{s}) \ \subset \ \mathsf{Pan}(\alpha, \mathsf{s}) \ \cap \ \mathsf{Ext}(\alpha, \mathsf{s}, \theta),$$

so, we have

$$M \in \mathsf{PanSub}(\alpha, \mathsf{s})\,. \tag{5}$$

We now prove three facts about w:

fact.1 The message of w is not a backdoor.

fact.2 If the message of w is an input, then $N \in \mathsf{Pan}(\alpha, \mathsf{s})$.

fact.3 If the message of w is an output, then $N \in \mathsf{PanSub}(\alpha, \mathsf{s})$.

> By Condition spr.1 of sub-party respecting (Definition 9, page 33), M would reject any backdoor, so fact fact.1 holds. Moreover:
> - If w is an input, then by Condition spr.3 and the fact that M did not reject, we have $N \in \mathsf{Pan}(\alpha, \mathsf{s})$, so fact fact.2 holds.
> - If w is an output, then by Condition spr.4 and the fact that M did not reject, we have $N \in \mathsf{Pan}(\alpha, \mathsf{s})$. Moreover, using Condition spr.2, we obtain $N \in \mathsf{PanSub}(\alpha, \mathsf{s})$, so fact fact.3 holds.

Although the sender identity indicated in w is N, it is possible that N was not the actual originator of the external-write. Indeed, w may have been modified by the control function.

We now divide the analysis into separate cases, depending on how w could have been generated.

Case 1: N actually issued the external-write received by M.

Assume that w was genuinely sent by N. Let θ' be the sub-prefix of θ that ends just before N sends the message. By the minimality of M, no ITI $U \in \mathsf{Pan}(\alpha, \mathsf{s})$ is awake in θ' without also belonging to $\mathsf{Ext}(\alpha, \mathsf{s}, \theta')$.

Since N is awake in θ' and that $N \in \mathsf{Pan}(\pi, \mathsf{s})$ (consequence of Facts fact.1, fact.2 and fact.3), we have:

[50] Note that the real ITI that made the external-write operation can be distinct of N : N can be the result of replacement of extended identity made by the control function.

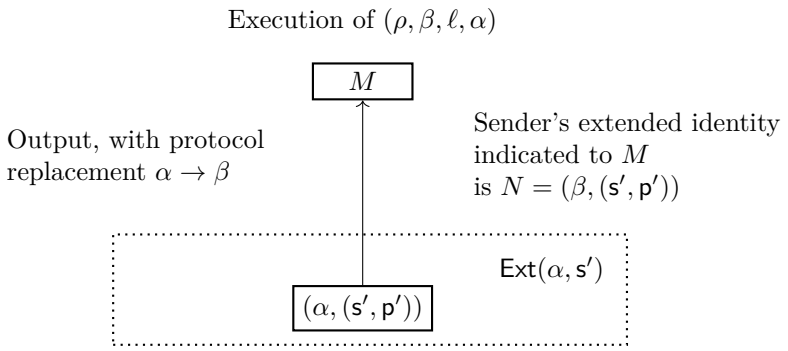


Figure 21: Illustration of cases 3.

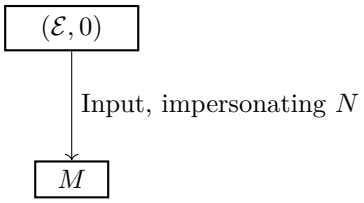
$$N \in \mathsf{Ext}(\alpha, \mathsf{s}, \theta') \subset \mathsf{Ext}(\alpha, \mathsf{s}, \theta).$$

Now, recalling that the message of w is not a backdoor (Fact fact.1), that $N \in \mathsf{Ext}(\alpha, \mathsf{s}, \theta')$ and that M is sleeping when receiving w, the definition of extended session (Definition 3) implies that $M \in \mathsf{Ext}(\alpha, \mathsf{s}, \theta)$[51].

This contradicts the assumption that $M \notin \mathsf{Ext}(\alpha, \mathsf{s}, \theta)$.

Case 2: The environment sent the external-write received by M, impersonating N.

We now consider the possibility that w was issued by the environment while impersonating N. This situation is illustrated below:



By compliance hypothesis on ρ, we have $\rho \notin \mathsf{Pan}(\alpha)$, Moreover, we know that $M \in \mathsf{PanSub}(\alpha, \mathsf{s})$ (Equation (5)). As a result, M cannot be a main party of the execution of ρ[52], and thus the environment cannot send an input to M.

This is a contradiction.

Case 3: The relevant prefix is a $(\rho, \beta, \ell, \alpha)$-execution prefix and the external-write received by M is the result of a replacement $\alpha \to \beta$.

As illustrated in Figure 21, this implies that $N = (\beta, (\mathsf{s}', \mathsf{p}'))$ for some sids and pids $(\mathsf{s}', \mathsf{p}')$.

Moreover, from Fact fact.3, we know that $N \in \mathsf{PanSub}(\alpha, \mathsf{s})$. We thus obtain that $N \in \mathsf{MainParty}(\beta, \mathsf{s}) \cap \mathsf{PanSub}(\alpha, \mathsf{s})$. However, this is impossible because α and β are disjoint, which implies that , $\mathsf{MainParty}(\beta, \mathsf{s}) \cap \mathsf{PanSub}(\alpha, \mathsf{s}) = \emptyset$.

Case 4: The relevant prefix is a $(\rho, \alpha, \ell, \beta)$-execution prefix and the external-write received by M is the result of a replacement $\beta \to \alpha$.

This implies that $N = (\alpha, (\mathsf{s}', \mathsf{p}'))$ for some sids and pids $(\mathsf{s}', \mathsf{p}')$, and that N sent an output to M. By construction of N and M and Fact fact.1, we also know that $M \in \mathsf{PanSub}(\alpha, \mathsf{s})$ is awaken by this output.

This lead to a contradiction. We justify it differently whether $\mathsf{s} = \mathsf{s}'$ or not:

[51] It is important to note that this conclusion also used the fact that if the message is an output, we actually have $N \in \mathsf{ExtSub}\alpha, \mathsf{s}, \theta'$.

[52] *i.e.*, cannot belong to $\mathsf{MainParty}(\rho, \mathsf{s}_{\mathsf{test}})$ for $\mathsf{s}_{\mathsf{test}}$ the test sid of the execution of (ρ, ϕ, ℓ, π).

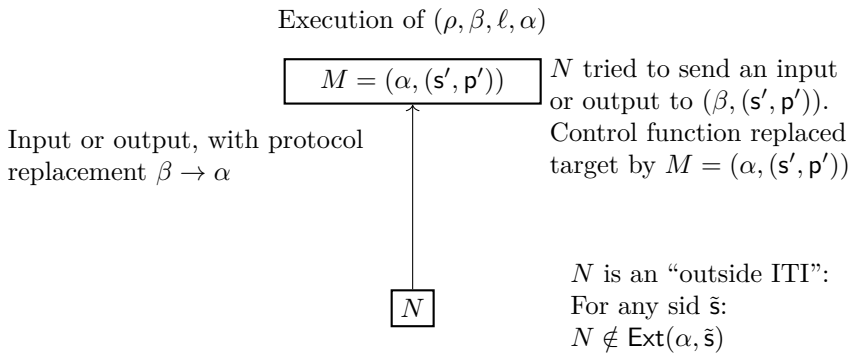


Figure 22: Illustration of cases 5 and 6.

- if $\mathsf{s} = \mathsf{s}'$, we use the fact that α is sub-party respecting. More precisely, condition spr.2 (see definition 9) implies that M cannot be awaken by this output.
- if $\mathsf{s} \neq \mathsf{s}'$, we use the fact that α is sub-party respecting and separated. Because α is separated, we know that $N \notin \mathsf{Pan}(\alpha, \mathsf{s})$. Then, we conclude by using condition spr.4 (see definition 9), as it implies that M cannot be awaken by this output.

Case 5: The relevant prefix is a $(\rho, \beta, \ell, \alpha)$-execution prefix and the external-write received by M is the result of a replacement $\beta \to \alpha$.

As illustrated in Figure 22, this implies that the message is an input and $N \in \mathsf{PanSub}(\alpha, \mathsf{s})$. However, by definition of the replacement $\beta, \to \alpha$, N is an output ITI and thus, $N \notin \mathsf{Pan}(\alpha, \mathsf{s})$. This is a contradiction.

In all possible cases, we reach a contradiction. Therefore, no such ITI M can exist, which completes the proof. □

Proposition 15. *Let π, ϕ be two ITMs. Assume that π and ϕ are disjoint, separated, and sub-party respecting protocols. Let $\ell \in \mathbb{N}$ and ρ a (π, ϕ)-compliant protocol.*

Then, for any sid s, and in any execution of (ρ, ϕ, ℓ, π) with any adversary and environment, none of the following situations can occur:

- *An ITI $V \in \mathsf{Pan}(\phi, \mathsf{s})$ receives an input or output whose indicated sender is an ITI $W \in \mathsf{Pan}(\phi, \mathsf{s})$, but W did not send such an input or output.*
- *An ITI $U \in \mathsf{Pan}(\phi, \mathsf{s})$ intends to send an input or output to an ITI $V \in \mathsf{Pan}(\phi, \mathsf{s})$, but V does not receive it.*
- *An ITI $V \in \mathsf{Pan}(\pi, \mathsf{s})$ receives an input or output whose indicated sender is an ITI $W \in \mathsf{Pan}(\pi, \mathsf{s})$, but W did not send such an input or output.*
- *An ITI $U \in \mathsf{Pan}(\pi, \mathsf{s})$ intends to send an input or output to an ITI $V \in \mathsf{Pan}(\pi, \mathsf{s})$, but V does not receive it.*

Proof. In an execution of (ρ, ϕ, ℓ, π), only one of the following three mechanisms could potentially cause the situations described above:

- The impersonation capacity of the environment or a redirection of an output to the environment.
- A replacement $\phi \to \pi$.
- A replacement $\pi \to \phi$.

Since $\rho \notin \mathsf{Pan}(\phi) \cup \mathsf{Pan}(\pi)$ (one of the conditions of the (π, ϕ)-compliance notion), none of these situations can be caused by impersonation capabilities of the environment or by output redirection to the environment.

For each situation, we justify why it cannot be the result of a replacement $\phi \to \pi$ or $\pi \to \phi$, thereby proving the impossibility of the situation.

First situation Suppose an ITI $V \in \mathsf{Pan}(\phi, \mathsf{s})$ receives an input or output whose indicated sender is an ITI $U \in \mathsf{Pan}(\phi, \mathsf{s})$, but U did not send such an input or output.

This cannot result from a replacement $\phi \to \pi$, as such a replacement does not modify the indicated sender's extended identity in the external-write.

If it results from a replacement $\pi \to \phi$, the only possibility is that (1) the message is an output and (2) an ITI $W = (\pi, (\mathsf{s}, \mathsf{p})) \in \mathsf{MainParty}\pi, \mathsf{s}$ sent the output to V, but the replacement $\phi \to \pi$ implied that V sees $U = (\phi, (\mathsf{s}, \mathsf{p}))$ as the sender.

However, this would directly contradict the item 4 of the (π, ϕ)-compliance notion.

Second situation Suppose an ITI $U \in \mathsf{Pan}(\phi, \mathsf{s})$ intends to send an input or output to an ITI $V \in \mathsf{Pan}(\phi, \mathsf{s})$, but V does not receive it.

This cannot result from a replacement $\pi \to \phi$, as such a replacement does not modify the target of the external-write.

Suppose instead that this is due to a replacement $\phi \to \pi$. Then $U \in \mathsf{Pan}(\phi, \mathsf{s})$ intended to send this input or output to $V = (\phi, (\mathsf{s}, \mathsf{p})) \in \mathsf{MainParty}\phi, \mathsf{s}$, but a replacement $\phi \to \pi$ implies that it is instead sent to $W = (\pi, (\mathsf{s}, \mathsf{p}))$ in (ρ, ϕ, ℓ, π).

Using Proposition 14 on U, which is awake when it intended to send this input, we know that $U \in \mathsf{Ext}(\phi, \mathsf{s}, \theta)$, where θ is the prefix before U sent the input.

According to the terminology used to define the replacement $\phi \to \pi$, this implies that U is not an outside ITI. As a result, the replacement $\phi \to \pi$ cannot not occur: this is a contradiction.

Third situation Suppose an ITI $V \in \mathsf{Pan}(\pi, \mathsf{s})$ receives an input or output whose indicated sender is an ITI $U \in \mathsf{Pan}(\pi, \mathsf{s})$, but U did not send such an input or output.

This cannot result from a replacement $\phi \to \pi$, as such a replacement does not modify the indicated sender's extended identity in the external-write.

This cannot either result from a replacement $\pi \to \phi$, as U would have code $\pi \neq \phi$.

Fourth situation Suppose an ITI $U \in \mathsf{Pan}(\pi, \mathsf{s})$ intends to send an input or output to an ITI $V \in \mathsf{Pan}(\pi, \mathsf{s})$, but V does not receive it.

This cannot result from a replacement $\pi \to \phi$, as such a replacement does not modify the target of the external-write.

This cannot either result from a replacement $\phi \to \pi$, as U would have code $\phi \neq \pi$.

This implies that $V \in \mathsf{MainParty}\phi, \mathsf{s}$. However, because $V \in \mathsf{Pan}(\pi)$ and ϕ and π are disjoint, this is a contradiction. □

Proposition 16 (Extended Session is Included in the Panoptic Session)**.** *Let α be a separated, sid-exposing, and sub-party respecting protocol. Then, for any α-relevant execution prefix θ and any $\mathsf{s} \in \{0,1\}^*$: $\mathsf{Ext}(\alpha, \mathsf{s}, \theta) \subset \mathsf{Pan}(\alpha, \mathsf{s})$.*

Proof. Fix an α-relevant execution prefix θ. Assume that θ is a $(\rho, \alpha, \ell, \beta)$-execution prefix, where β is an ITM that is separated and disjoint from α, $\ell \in \mathbb{N}^*$, and $\rho \notin \mathsf{Pan}(\alpha) \cup \mathsf{Pan}(\beta)$. The proof is analogous in other cases.

The main difficulty of this proof lies in the mismatch between θ, which is a replacement execution prefix of some protocol $\rho \neq \alpha$, and the definition of $\mathsf{Pan}(\alpha, \mathsf{s})$, which considers only α-execution prefixes (i.e., executions of α without replacements).

To overcome this, we proceed in two steps: we first show that θ is "close enough" to an α-execution prefix, then use this to prove the inclusion.

Step 1: Construction and analysis of $\mathcal{E}_{\mathsf{sim}}$.

We define an environment $\mathcal{E}_{\mathsf{sim}}$ (given in Figure 24) whose role is to simulate the entire execution of θ—including $(\mathcal{E}, 0)$ and $(\mathcal{A}, \Diamond)$—except for the ITIs belonging to $\mathsf{Pan}(\alpha, \mathsf{s})$.[53]

Let θ_{sim} be the prefix of the execution of α with environment $\mathcal{E}_{\mathsf{sim}}$, dummy adversary $\mathcal{D}$, and test sid s. We claim that θ_{sim} "simulates" θ.

This setup is illustrated in Figure 23. Each ITI in θ is either a real ITI in $\mathsf{Ext}(\alpha, \mathsf{s}, \theta_{\mathsf{sim}})$ or simulated by $(\mathcal{E}_{\mathsf{sim}}, 0)$. By construction, $\mathcal{E}_{\mathsf{sim}}$ only simulates ITIs that are not in $\mathsf{Pan}(\alpha, \mathsf{s})$.

Ideally, this simulation would be perfect: every message exchanged between ITIs in θ would correspond to a real or simulated message in θ_{sim}. However, this is not quite the case. There are two primary discrepancies:

- The dummy adversary $\mathcal{D}$ in θ_{sim} has no counterpart in θ.
- Not all interactions in θ are simulated exactly in θ_{sim}. For instance, in θ, an ITI $M \notin \mathsf{Pan}(\alpha, \mathsf{s})$ might send an input to some $N \in \mathsf{PanSub}(\alpha, \mathsf{s})$, whereas in θ_{sim}, the simulation of M simply deactivates instead.

Despite these differences, the simulation is "almost perfect". The dummy adversary $\mathcal{D}$ does not pose a problem: it merely relays communications between ITIs of the test session and the simulated adversary $(\mathcal{A}, \Diamond)$, ensuring faithful simulation.

The second issue is more subtle. Differences only arise in five specific scenarios, which we analyze below:

- **Input from outside the panoptic session:** If $M \notin \mathsf{Pan}(\alpha, \mathsf{s})$ sends an input to $N \in \mathsf{PanSub}(\alpha, \mathsf{s})$ in θ, then the simulation in θ_{sim} deactivates M instead of sending the input. This is harmless because N would reject the input anyway due to Condition spr.3 of sub-party respecting.
- **Output to the panoptic session:** If $M \notin \mathsf{Pan}(\alpha, \mathsf{s})$ sends an output to $N \in \mathsf{Pan}(\alpha, \mathsf{s})$ in θ, the simulation again deactivates M. This causes no issue because, by Condition spr.4, N would reject the message.
- **Unspecified simulation of outputs to outside:** The definition of $\mathcal{E}_{\mathsf{sim}}$ does not specify how to simulate outputs from $M \in \mathsf{PanSub}(\alpha, \mathsf{s})$ to $N \notin \mathsf{Pan}(\alpha, \mathsf{s})$. This case, however, cannot occur due to Condition spr.5.
- **Unspecified simulation of inputs to outside:** Likewise, the simulation does not specify how to handle inputs from $M \in \mathsf{Pan}(\alpha, \mathsf{s})$ to $N \notin \mathsf{Pan}(\alpha, \mathsf{s})$, which is also ruled out by Condition spr.5.
- **Control-function-induced sender/receiver changes:** If a message exchanged between M and N with $M, N \in \mathsf{Ext}(\alpha, \mathsf{s}, \theta)$ were modified by the control function due to a protocol replacement $\alpha \to \beta$ or $\beta \to \alpha$, such a modification would not occur in θ_{sim}, as θ_{sim} is an execution without replacement—thus breaking the simulation. However, this situation cannot arise thanks to Proposition 15 and the fact that $\mathsf{Ext}(\alpha, \mathsf{s}, \theta_{\mathsf{sim}}) \subset \mathsf{Pan}(\alpha, \mathsf{s})$.

In summary, the only differences between θ and θ_{sim} are:

bad.1 If $M \notin \mathsf{Pan}(\alpha, \mathsf{s})$ sends an input to $N \in \mathsf{PanSub}(\alpha, \mathsf{s})$, N rejects it in θ, and M is deactivated in θ_{sim}.

bad.2 If $M \notin \mathsf{Pan}(\alpha, \mathsf{s})$ sends an output to $N \in \mathsf{Pan}(\alpha, \mathsf{s})$, N rejects it in θ, and M is deactivated in θ_{sim}.

Finally, since θ_{sim} is an α-execution prefix, we have by definition:

$$\mathsf{Ext}(\alpha, \mathsf{s}, \theta_{\mathsf{sim}}) \subset \mathsf{Pan}(\alpha, \mathsf{s}).$$

Step 2: Proof of the proposition.

[53] This is achieved using Expose^{α}, hence the requirement that α be sid-exposing.

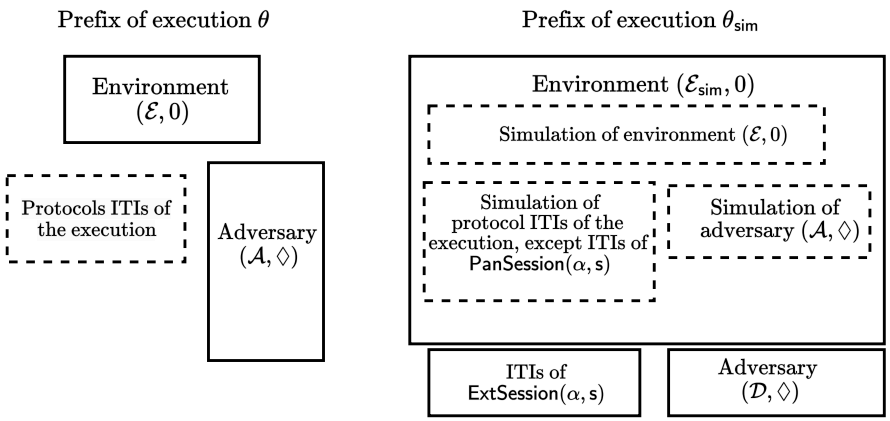


Figure 23: Illustration of θ and θ_{sim}.

Let $u = |\mathsf{ExtSub}(\alpha, \mathsf{s}, \theta)|$. For each $0 \leq n \leq u$, define the property:

$$\mathcal{P}(n): \quad \text{For any sub-prefix } \theta' \text{ of } \theta \text{ such that } |\mathsf{Ext}(\alpha, \mathsf{s}, \theta')| = n : \\ \mathsf{Ext}(\alpha, \mathsf{s}, \theta') \subset \mathsf{Pan}(\alpha, \mathsf{s}) \,.$$

Our goal is to prove $\mathcal{P}(u)$ by induction on n.

Base case: $\mathcal{P}(0)$.
If $|\mathsf{ExtSub}(\alpha, \mathsf{s}, \theta')| = 0$, then by definition,

$$\mathsf{Ext}(\alpha, \mathsf{s}, \theta') = \mathsf{MainParty}(\alpha, \mathsf{s}) \subset \mathsf{Pan}(\alpha, \mathsf{s}).$$

Inductive step: Assume $\mathcal{P}(n-1)$ holds for $0 < n \leq u$, and let us prove $\mathcal{P}(n)$.
Suppose $|\mathsf{ExtSub}(\alpha, \mathsf{s}, \theta)| = n$. Let N be the last ITI added to $\mathsf{ExtSub}(\alpha, \mathsf{s}, \theta)$. Let $\hat{\theta} \subset \theta$ be a sub-prefix such that

$$\mathsf{ExtSub}(\alpha, \mathsf{s}, \hat{\theta}) = \mathsf{ExtSub}(\alpha, \mathsf{s}, \theta) \setminus \{N\}.$$

By definition of Ext, in θ, N was sleeping and became active upon receiving a message (input or output) from some ITI $M \in \mathsf{Ext}(\alpha, \mathsf{s}, \theta)$.
By the inductive hypothesis:

- If $M \in \mathsf{MainParty}(\alpha, \mathsf{s})$, then clearly $M \in \mathsf{Pan}(\alpha, \mathsf{s})$.
- If $M \in \mathsf{ExtSub}(\alpha, \mathsf{s}, \theta)$, then $M \in \mathsf{ExtSub}(\alpha, \mathsf{s}, \hat{\theta})$, and since $\mathcal{P}(n-1)$ holds, we have $M \in \mathsf{Pan}(\alpha, \mathsf{s})$.

Hence, the message that activated N was sent by an ITI $M \in \mathsf{Pan}(\alpha, \mathsf{s})$. Therefore, it cannot fall under the exceptional cases $\mathsf{bad}.1$ or $\mathsf{bad}.2$, since these require $M \notin \mathsf{Pan}(\alpha, \mathsf{s})$. This means the message corresponds to a real message in θ_{sim}, and thus both M and N lie in $\mathsf{Ext}(\alpha, \mathsf{s}, \theta_{\mathsf{sim}})$.
Since $\mathsf{Ext}(\alpha, \mathsf{s}, \theta_{\mathsf{sim}}) \subset \mathsf{Pan}(\alpha, \mathsf{s})$, we conclude that $N \in \mathsf{Pan}(\alpha, \mathsf{s})$.
This completes the inductive step, and hence the proof. □

C.4 Proof of the Theorem Relying on the Use of a Central Lemma

This section and the following sections (C.5, C.6, C.7, and C.8) are entirely dedicated to proving the UC Theorem of Updated UC.
More precisely:

$(\mathcal{E}_{\mathsf{sim}}, 0)$ on entry z

// Note that $\ell \in \mathbb{N}^*$ and $\mathsf{s} \in \{0,1\}^*$ are fixed, known values
// Note also that the auxiliary algorithm Simulation is defined in Figure 25
// Note also that the result of $(\mathcal{E}_{\mathsf{sim}}, 0)$ is described in Line 15 of Simulation

1 : At first activation :
2 : Simulation (z)
3 : At reception of a backdoor "(m, M)" from the adversary :
Simulation $((\mathsf{m}, M))$
4 : At reception of an output m from the protocol with identifier (s, p) and intended receiver an ITI M :
// by definition of protocol execution, note that s is the test sid
5 : Simulation $((\mathsf{m}, \mathsf{s}, \mathsf{p}, M))$
6 : At activation – not the first – with no message
// This happens in exactly three situations:
// - If an ITI deactivates (including rejects).
// - If the adversary sends a backdoor to a protocol ITI that does not exist.
// - The control function disallows an external write operation.
7 : Simulation $()$

Figure 24: Main algorithm used in the definition of $\mathcal{E}_{\mathsf{sim}}$, to prove Proposition 16. It relies on the algorithm Simulation defined in Figure 25.

Simulation(input)

1 : // Note that $\ell \in \mathbb{N}^*$ and $\mathsf{s} \in \{0,1\}^*$ are fixed, known values
if input can be parsed as $z \in \{0,1\}^*$ **then**
2 : **initialize** the simulation of the (α, ℓ, β)-execution of ρ
with environment $\mathcal{E}$ and adversary $\mathcal{A}$ on input z
3 : Continue the simulation until one of the exceptions after line 14 occurs
4 : **else if** input can be parsed as a pair $(\mathsf{m}, M) \in \{0,1\}^* \times$ SetITI **then**
5 : **simulate** the sending of backdoor m to the simulation of
$(\mathcal{A}, \Diamond)$ as if it came from M.
6 : Continue the simulation until one of the exceptions after line 14 occurs
7 : **else if** input can be parsed as $(\mathsf{m}, \mathsf{s}, \mathsf{p}, M) \in \{0,1\}^* \times \{0,1\}^* \times$ SetITI **then**
8 : **simulate** the sending of output m to the simulation of M as if
it came from $(\alpha, (\mathsf{s}, \mathsf{p}))$
9 : Continue the simulation until one of the exceptions after line 14 occurs
10 : **else if** Simulation has been called with no input **then**
11 : **simulate** the activation of $(\mathcal{E}, 0)$ with no message and continue
12 : the simulation until one of the exceptions after line 14 occurs
13 : **else halt**
14 : **List of exceptions:**
15 : **exception** : simulation of $(\mathcal{E}, 0)$ returns a bit $b \in \{0,1\}$ **then return** b
16 : **exception** : an input or output "m" is sent by a simulated protocol ITI M
to a protocol ITI N that is a main party of (α, s) **then**
if it is an input **then**
17 : **send input** "m" to the protocol with same sid and pid than N
18 : **if** it is an output **then**
19 : **simulate** the deactivation of M and continue the simulation
20 : until one of the exceptions after line 14 occurs
21 : **exception** : an input or output "m" is sent by a simulated protocol ITI M
to a protocol ITI N with code $\neq \alpha$
such that $\mathsf{Expose}^{\alpha}(N) = \mathsf{s}$ **then**
22 : **simulate** the deactivation of M and continue the simulation
23 : until one of the exceptions after line 14 occurs
24 : **exception** : the simulation of $(\mathcal{A}, \Diamond)$ intends to send backdoor m
25 : to a protocol ITI N such that $\mathsf{Expose}^{\alpha}(N) = \mathsf{s}$ **then**
26 : **send input** "(m, N)" to the adversary

Figure 25: Auxiliary algorithm used in the definition of $\mathcal{E}_{\mathsf{sim}}$, to prove Proposition 16.

- This section restates and proves the theorem with the help of a central lemma (Lemma 1).
- Section C.5 proves the central lemma using two auxiliary lemmas (Lemmas 2 and 3).
- Section C.6 defines an environment $\mathcal{E}_\phi$ and a polynomial Q utilized to prove the two auxiliary lemmas.
- Section C.7 proves the first auxiliary lemma.
- Section C.8 proves the second auxiliary lemma.

We now restate the theorem and begin the proof.

Restatement of Theorem 1 (Universal Composition Theorem)**.**
Let $\boldsymbol{\pi}$, $\boldsymbol{\phi}$ and $\boldsymbol{\rho}$ three PPT parametrized systems and ξ a predicate on extended identities. Assume that $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ is a PPT triplet of parametrized systems, and that $\boldsymbol{\phi}, \boldsymbol{\pi}$ are sub-party respecting, separated, disjoint and sid-exposing. Furthermore, suppose that $\boldsymbol{\rho}$ is $(\boldsymbol{\pi}, \boldsymbol{\phi}, \xi)$-compliant and that $\boldsymbol{\pi}$ ξ-UC-emulates $\boldsymbol{\phi}$. Then, $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ UC-emulates $\boldsymbol{\rho}$.

Using UC-emulation with respect to the dropping dummy adversary $\mathcal{D}^{\mathsf{DROP}}$, we are looking for a PPT adversary $\mathcal{S}_\rho$ such that, for any PPT environment $\mathcal{E}$:

$$(\boldsymbol{\phi}, \infty, \boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho}, \mathcal{D}^{\mathsf{DROP}}, \mathcal{E}} \approx \mathsf{EXEC}_{\boldsymbol{\rho}, \mathcal{S}_\rho, \mathcal{E}} \,. \tag{6}$$

The fact that $\boldsymbol{\pi}$ ξ-UC-emulates $\boldsymbol{\phi}$ guarantees that there exists a PPT adversary $\mathcal{S}_\phi$ of $\boldsymbol{\phi}$, such that for any ξ-identity-bounded, PPT environment $\mathcal{E}$:

$$\mathsf{EXEC}_{\boldsymbol{\pi}, \mathcal{D}, \mathcal{E}} \approx \mathsf{EXEC}_{\boldsymbol{\phi}, \mathcal{S}_\phi, \mathcal{E}} \,. \tag{7}$$

We construct a PPT adversary $\mathcal{S}_\rho$ out of $\mathcal{S}_\phi$ and then show it satisfies Equation 6.

C.4.1 Definition of the Adversary $\mathcal{S}_\rho$.

The complete code of the adversary $\mathcal{S}_\rho$ is provided in Figure 26, and its behavior is described in this paragraph.

The adversary $\mathcal{S}_\rho$ simulates multiple variants of $\mathcal{S}_\phi$, each slightly modified. For each $\mathsf{s} \in \{0,1\}^*$, the adversary $\mathcal{S}_{\phi,\mathsf{s}}$ behaves identically to $\mathcal{S}_\phi$, with one exception: when $\mathcal{S}_\phi$ attempts to send a backdoor to an ITI $M \notin \mathsf{Pan}(\phi, \mathsf{s})$, $\mathcal{S}_{\phi,\mathsf{s}}$ deactivates instead of sending the message. This modification is introduced to avoid undesirable side effects.[54] The full code of $\mathcal{S}_{\phi,\mathsf{s}}$ is given in Figure 27.

The behavior of $\mathcal{S}_\rho$, represented in Figure 28, can be summarized as follows:

- Upon receiving an input (m, M) from the environment, $\mathcal{S}_\rho$ uses the sid-exposing property of both $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ in order to compute $\mathsf{positionM}_\pi = \mathsf{PSExpose}^{\boldsymbol{\pi}}(k, M)$, and $\mathsf{positionM}_\phi = \mathsf{PSExpose}^{\boldsymbol{\phi}}(k, M)$. It then proceeds as follows:
  - If $\mathsf{positionM}_\pi \in \{0,1\}^*$, then $\mathcal{S}_\rho$ uses the simulation of $\mathcal{S}_{\phi,\mathsf{positionM}}$ to process the input (m, M).
  - Else, if $\mathsf{positionM}_\phi \in \{0,1\}^*$, then $\mathcal{S}_\rho$ rejects.[55]
  - Otherwise, if $\mathsf{positionM}_\phi = \mathsf{positionM}_\pi = \bot$, then $\mathcal{S}_\rho$ acts as the dummy adversary and sends backdoor m to M.
- Upon receiving a backdoor m from some ITI M, $\mathcal{S}_\rho$ uses the sid-exposing property of $\boldsymbol{\phi}$ to compute $\mathsf{positionM}_\phi = \mathsf{PSExpose}^{\boldsymbol{\phi}}(k, M)$. It then proceeds as follows:
  - If $\mathsf{positionM}_\phi \in \{0,1\}^*$, then $\mathcal{S}_\rho$ uses the simulation of $\mathcal{S}_{\phi,\mathsf{positionM}}$ to process the backdoor m received from M.

[54] In the proof of the UC Theorem of Plain UC [Can20, (Theorem 22)], the simulator directly simulates copies of $\mathcal{S}_\phi$. However, our proof requires this adjustment. Although we do not describe a specific vulnerability in plain UC, we believe this change is also necessary in that context.

[55] This case does not appear in the simulator used in the proof of the UC Theorem of Plain UC [Can20, (Theorem 22)]. Its omission is the source of Weakness (W.5), as described in Section D.3, for readers familiar with the proof of the plain UC Theorem.

$(\mathcal{S}_\rho, \Diamond)$

// $(\mathcal{S}_\rho, \Diamond)$ will simulate multiple modified adversaries $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$
// whose code is given in Figure 27

1 : At first activation, $(\mathcal{S}_\rho, \Diamond)$ reads the security parameter k indicated in its dedicated tape, then acts as indicated
2 : At reception of an external-write w that is a backdoor m from an ITI M :
3 : $\mathsf{positionM}_\phi \in \{\bot\} \cup \{0,1\}^* \leftarrow \mathsf{PSExpose}^{\phi}(k, M)$
4 : **if** $\mathsf{positionM}_\phi \in \{0,1\}^*$ **then** $\mathsf{Simulate}\left(w, \mathsf{positionM}_\phi\right)$
5 : **if** $\mathsf{positionM}_\phi = \bot$ **then send backdoor** "(m, M)" to the environment
6 : At reception of an external-write w that is an input (from the environment) :
7 : **checkorchalt** the message can be parsed as a pair (m, M) with m a message and M an ITI
8 : $\mathsf{positionM}_\pi \in \{\bot\} \cup \{0,1\}^* \leftarrow \mathsf{PSExpose}^{\pi}(k, M)$
9 : $\mathsf{positionM}_\phi \in \{\bot\} \cup \{0,1\}^* \leftarrow \mathsf{PSExpose}^{\phi}(k, M)$
10 : **if** $\mathsf{positionM}_\pi \in \{0,1\}^*$ **then** $\mathsf{Simulate}\,(w, \mathsf{positionM})$
11 : **else if** $\mathsf{positionM}_\phi \in \{0,1\}^*$ **then reject**
12 : **else then send backdoor** "m" to M

$\mathsf{Simulate}\,(w, \mathsf{s} \in \{0,1\}^*)$

1 : **checkorreject** w is an external-write
2 : **if** $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ has never been simulated **then**
3 : **initialize** its simulation (with security parameter k)
4 : $\widetilde{w} \leftarrow w$ with recipient replaced by $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$
5 : **simulate** applying $\widetilde{w}$ to the simulation of $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$
6 : Then simulation of $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ continues until one of the following events occurs :
if the simulation of $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ rejects **then reject**
7 : **if** the simulation of $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ deactivates (but does not reject) **then deactivate**
8 : **if** the simulation of $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ intends to send a backdoor **then**
9 : **send** this backdoor

Figure 26: Code of adversary $\mathcal{S}_\rho$. It uses the adversary $\mathcal{S}_{\phi,\mathsf{s}}$ defined in Figure 27.

$(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$

// $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ will simulate the adversary $(\mathcal{S}_{\phi}, \Diamond)$

1 : At first activation, with reception of an external-write w
2 : $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ reads the security parameter k indicated in its dedicated tape.
3 : **initialize** a simulation of the adversary $(\mathcal{S}_{\phi}, \Diamond)$ with security parameter k
4 : SimulateOperation (w)
5 : At reception of an external-write w :
6 : SimulateOperation (w)

SimulateOperation (w)

1 : **checkorchalt** w is an external-write
2 : $\widetilde{w} \leftarrow w$ with recipient replaced by $(\mathcal{S}_{\phi}, \Diamond)$
3 : **simulate** applying $\widetilde{w}$ to the simulation of $(\mathcal{S}_{\phi}, \Diamond)$
4 : Then simulation of $(\mathcal{S}_{\phi}, \Diamond)$ continues until one of the following events occurs :
5 : **if** the simulation of $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ rejects **then reject**
6 : **if** the simulation of $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ deactivates (but does not reject) **then deactivate**
7 : **if** the simulation of $(\mathcal{S}_{\phi}, \Diamond)$ intends to send a backdoor to the environment **then**
8 : **send** this backdoor to the environment
9 : **if** the simulation of $(\mathcal{S}_{\phi}, \Diamond)$ intends to send a backdoor to an ITI N **then**
10 : positionN $\in \{\perp\} \cup \{0,1\}^* \leftarrow \mathsf{PSExpose}^{\phi}(k, N)$
11 : **if** positionN $\neq \mathsf{s}$ **then deactivate**
12 : **else send** this backdoor to N

Figure 27: Code of the adversary $\mathcal{S}_{\phi,\mathsf{s}}$ for $\mathsf{s} \in \{0,1\}^*$.

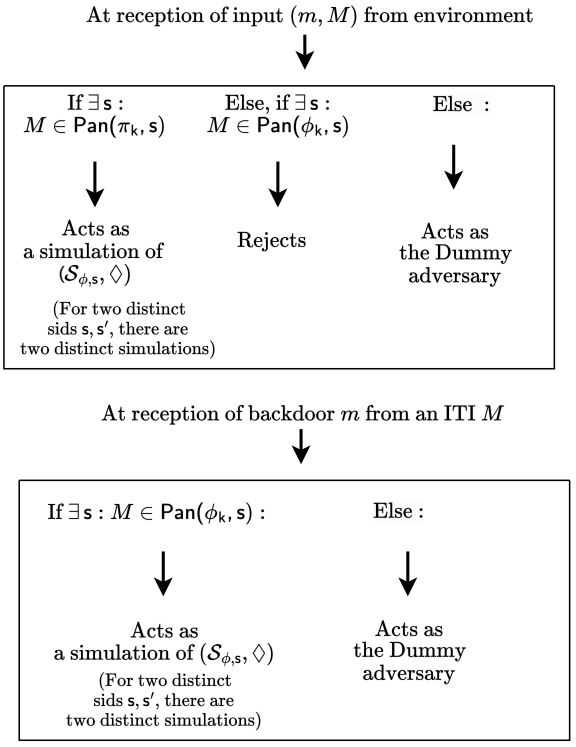
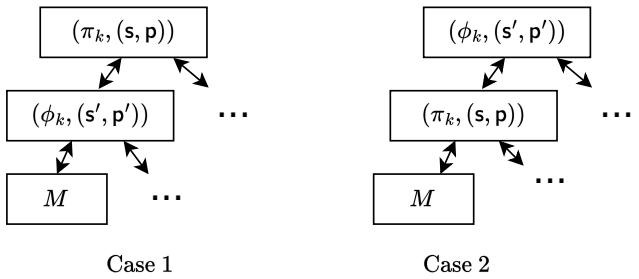


Figure 28: Simplified representation of the adversary $(\mathcal{S}_\rho, \Diamond)$ for security parameter k.

– Otherwise, $\mathcal{S}_\rho$ acts as the dummy adversary and sends the backdoor (m, M) to the environment.

The adversary $\mathcal{S}_\rho$ is PPT because $\mathcal{S}_\phi$ is PPT, and both $\mathsf{Expose}^{\boldsymbol{\pi}}$ and $\mathsf{Expose}^{\boldsymbol{\phi}}$ are PPT Turing machines.

We end this description by noting that disjointness property is essential to avoid ambiguous situations. For instance, if an ITI M lies in $\mathsf{Pan}(\pi_k, \mathsf{s}) \cap \mathsf{Pan}(\phi_k, \mathsf{s}')$ – as may be detected via the sid-exposing property of $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ – then it would be impossible to decide whether we are in the first or second case below, each requiring distinct treatments.



C.4.2 Proof of Equation (6) by Contraposition, Using a Central Lemma

To show that $\mathcal{S}_\rho$ satisfies Equation (6), we proceed by contraposition.

Suppose that there exists a PPT environment $\mathcal{E}_\rho$ such that Equation (6) is false. By definition, this means: $\exists c, d \in \mathbb{N}$ such that $\forall k_0 \in \mathbb{N}, \exists k > k_0, \exists z \in \bigcup_{0<a\leq k^d}\{0,1\}^a$:

$$\left|\Pr\left[(\boldsymbol{\phi}, \infty, \boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{D}^{\mathsf{DROP}},\mathcal{E}_\rho}(k,z)=1\right] - \Pr\left[\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{S}_\rho,\mathcal{E}_\rho}(k,z)=1\right]\right| \geq k^{-c}\,. \tag{8}$$

We will use this environment $\mathcal{E}_\rho$ to construct a ξ-identity-bounded PPT environment $\mathcal{E}_\phi$ for which Equation (7) also fails.

More precisely, we will show that $\mathcal{E}_\phi$ satisfies the following central lemma:

Lemma 1 (Central Lemma for UC Theorem)**.** *There is a polynomial Q increasing on $[0,\infty[$, with values in $]0,\infty[$, such that, for any $(k,z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d}\{0,1\}^a$:*

$$\begin{aligned}&\left|\Pr\left[\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi}(k,z)=1\right] - \Pr\left[\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi}(k,z)=1\right]\right| \\ &= \frac{1}{Q(k)}\left|\Pr\left[(\boldsymbol{\phi},\infty,\boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{D}^{\mathsf{DROP}},\mathcal{E}_\rho}(k,z)=1\right] - \Pr\left[\mathsf{EXEC}_{\rho,\mathcal{S}_\rho,\mathcal{E}_\rho}(k,z)=1\right]\right|.\end{aligned}$$

Using Lemma 1 and Equation (8), we can conclude that Equation (7) is false. Indeed, they imply that $\forall k_0, \exists k > k_0, \exists z \in \bigcup_{0<a\leq k^d}\{0,1\}^a$ such that:

$$\left|\Pr\left[\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi}(k,z)=1\right] - \Pr\left[\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi}(k,z)=1\right]\right| \geq \frac{k^{-c}}{Q(k)}\,.$$

Which completes the proof of UC Theorem.

As a consequence, to complete the proof of the theorem, it remains to construct the environment $\mathcal{E}_\phi$ and prove Lemma 1.

This will be carried out in the following steps:

- Section C.5: Proof of Lemma 1, using two auxiliary lemmas.
- Section C.6: Definition of the environment $\mathcal{E}_\phi$ and the polynomial Q.
- Section C.7: Proof of the first auxiliary lemma (Lemma 2).
- Section C.8: Proof of the second auxiliary lemma (Lemma 3).

C.5 Proof of the Central Lemma (Lemma 1) Using two Auxiliary Lemmas

In this section, we assume the existence of a PPT ξ-identity environment $\mathcal{E}_\phi$, along with a polynomial Q that is increasing on $[0,\infty[$ and takes values in $[2,\infty[$. We also assume that for each $\ell \in \mathbb{N}$, there exists an environment $\mathcal{E}_\phi^\ell$ such that the first operation of $\mathcal{E}_\phi$ on any input (k,z) is to uniformly sample an integer $\ell \in \{0,\ldots,Q(k)-1\}$ and then to act exactly as the environment $\mathcal{E}_\phi^\ell$ with input (k,z) until the end of the execution.

Furthermore, we assume that $\mathcal{E}_\phi$ and Q satisfy the following two auxiliary lemmas, which will be proven later.

Lemma 2 (First Auxiliary Lemma)**.** *Let d the constant described in Section C.4, whose existence follows from the assumption that Equation (6) is false. For $(k,z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d}\{0,1\}^a$ and $\ell \in \{0,\cdots,Q(k)-2\}$:*

$$\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi^\ell}(k,z) = \mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi^{\ell+1}}(k,z)\,.$$

Proof. See Appendix C.7. □

Lemma 3 (Second Auxiliary Lemma)**.** *Let d the constant described in Section C.4, whose existence follows from the assumption that Equation (6) is false. For $(k,z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d}\{0,1\}^a$:*

$$\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi^{Q(k)-1}}(k,z) = (\boldsymbol{\phi},\infty,\boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{D}^{\mathsf{DROP}},\mathcal{E}_\rho}(k,z)\,,$$
$$\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi^0}(k,z) = \mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{S}_\rho,\mathcal{E}_\rho}(k,z)\,.$$

Proof. See Appendix C.8. □

With these assumptions, we can prove the central Lemma.

Restatement of Lemma 1 (Central Lemma for UC Theorem)**.** *There is a polynomial Q increasing on $[0,\infty[$, with values in $[2,\infty[$, such that, for any $(k,z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d}\{0,1\}^a$:*

$$\left|\Pr\left[\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi}(k,z)=1\right] - \Pr\left[\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi}(k,z)=1\right]\right|$$
$$= \frac{1}{Q(k)}\left|\Pr\left[(\boldsymbol{\phi},\infty,\boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{D}^{\mathsf{DROP}},\mathcal{E}_\rho}(k,z)=1\right] - \Pr\left[\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{S}_\rho,\mathcal{E}_\rho}(k,z)=1\right]\right|.$$

Proof. We set $(k,z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d}\{0,1\}^a$. For $0 \leq \ell \leq Q(k)-1$, we denote by E^ℓ the event "$\mathcal{E}_\phi$ sampled ℓ".

We can see that for any $0 \leq \ell \leq Q(k)-1$, $\Pr\left[E^\ell\right] = 1/Q(k)$. Moreover, we have:

$$\Pr\left[\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi}(k,z)=1 \;\middle|\; E^\ell\right] = \Pr\left[\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi^\ell}(k,z)=1\right],$$
$$\Pr\left[\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi}(k,z)=1 \;\middle|\; E^\ell\right] = \Pr\left[\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi^\ell}(k,z)=1\right]. \tag{9}$$

We can thus compute:

$$\begin{aligned}
&\left|\Pr\left[\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi}(k,z)=1\right] - \Pr\left[\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi}(k,z)=1\right]\right| \\
&= \frac{1}{Q(k)}\left|\sum_{\ell=0}^{Q(k)-1} \Pr\left[\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi}(k,z)=1 \;\middle|\; E^\ell\right] - \Pr\left[\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi}(k,z)=1 \;\middle|\; E^\ell\right]\right| \\
&\overset{\text{Equation (9)}}{=} \frac{1}{Q(k)}\left|\sum_{\ell=0}^{Q(k)-1} \Pr\left[\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi^\ell}(k,z)=1\right] - \Pr\left[\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi^\ell}(k,z)=1\right]\right| \\
&= \frac{1}{Q(k)}\left|\Pr\left[\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi^{Q(k)-1}}(k,z)=1\right] - \Pr\left[\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi^0}(k,z)=1\right]\right. \\
&\qquad \left. + \sum_{\ell=0}^{Q(k)-2} \Pr\left[\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi^\ell}(k,z)=1\right] - \Pr\left[\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi^{\ell+1}}(k,z)=1\right]\right| \\
&\overset{\text{Lemma 2}}{=} \frac{1}{Q(k)}\left|\Pr\left[\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi^{Q(k)-1}}(k,z)=1\right] - \Pr\left[\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi^0}(k,z)=1\right]\right| \\
&\overset{\text{Lemma 3}}{=} \frac{1}{Q(k)}\left|\Pr\left[(\boldsymbol{\phi},\infty,\boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{D}^{\mathsf{DROP}},\mathcal{E}_\rho}(k,z)=1\right] - \Pr\left[\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{S}_\rho,\mathcal{E}_\rho}(k,z)=1\right]\right|.
\end{aligned}$$

□

C.6 Definition of Environment $\mathcal{E}_\phi$ and Polynomial Q

In this section, we define the environment $\mathcal{E}_\phi$, which corresponds to the environment $\mathcal{E}_\pi$ used in the proof of [Can20, (Theorem 22)].

This definition requires introducing an auxiliary adversary, denoted $\widetilde{\mathcal{S}}_\rho$, which is a variation of $\mathcal{S}_\rho$ designed specifically for $(\boldsymbol{\pi}, \ell, \boldsymbol{\phi})$-executions of ρ.

Additionally, we introduce a polynomial Q, defined separately after presenting $\widetilde{\mathcal{S}}_\rho$. It will be used in the construction of $\mathcal{E}_\phi$ to ensure it is a PPT ITM.

The final subsection of this section then formally defines and explains the construction of $\mathcal{E}_\phi$.

Throughout this section, we will sometimes refer to the ordering of extended sessions of ϕ_k according to their position in the execution. By definition of a $(\boldsymbol{\pi}, \ell, \boldsymbol{\phi})$-execution, the first ℓ extended sessions of ϕ_k are replaced by extended sessions of π_k. We use the same ordering for panoptic sessions: for $u \geq 1$, the u-th panoptic session of ϕ_k corresponds to the u-th extended session of ϕ_k. Therefore, the first ℓ panoptic sessions of ϕ_k are replaced by the corresponding panoptic sessions of π_k.

C.6.1 Definition of Adversary $\widetilde{\mathcal{S}}_\rho$.

The algorithm for $\widetilde{\mathcal{S}}_\rho$ is presented in Figure 29.

Its behavior is illustrated in Figure 30. To understand this diagram, recall that in any $(\boldsymbol{\pi}, \ell, \boldsymbol{\phi})$-execution of ρ with security parameter $k \in \mathbb{N}$, each message received by the adversary is a pair $(\mathsf{ReplacedSession}, m)$, where $\mathsf{ReplacedSession}$ denotes the set of sids corresponding to extended sessions of ϕ_k that have been replaced by sessions of π_k.

We can see that $\widetilde{\mathcal{S}}_\rho$ is PPT, since $\mathcal{S}_\phi$, $\mathsf{PSExpose}^{\boldsymbol{\pi}}$, and $\mathsf{PSExpose}^{\boldsymbol{\phi}}$ are PPT.

C.6.2 Definition of Polynomial Q.

Since $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ is a PPT triplet, there exists a polynomial R such that for any $\ell \in \mathbb{N}$ and input $(k, z) \in \mathbb{N} \times \{0,1\}^*$, every $(\boldsymbol{\phi}, \ell, \boldsymbol{\pi})$-execution of $\boldsymbol{\rho}$ with adversary $\widetilde{\mathcal{S}}_\rho$ and environment $\mathcal{E}_\rho$ on input (k, z) terminates in at most $R(k + |z|)$ steps. Without loss of generality, we may assume R is increasing on $[0, \infty[$ and takes values in $[2, \infty[$.

We define $Q(k) = R(k + k^d)$, where d is the constant from Equation (8) page 73. Consequently, for any $\ell \in \mathbb{N}$ and any $(k, z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d} \{0,1\}^a$, the total number of computational steps (and therefore of extended sessions of ϕ_k and π_k) in a $(\boldsymbol{\phi}, \ell, \boldsymbol{\pi})$-execution of $\boldsymbol{\rho}$ with adversary $\widetilde{\mathcal{S}}_\rho$, environment $\mathcal{E}_\rho$ and input (k, z) is bounded by $Q(k)$.

C.6.3 Definition of Environment $\mathcal{E}_\phi$

The definition of $\mathcal{E}_\phi$ is given via two figures:

- Figure 32 presents the main function of $\mathcal{E}_\phi$, which relies on a subroutine $\mathsf{Simulation}$.
- Figure 33 provides the definition of $\mathsf{Simulation}$.

We now give an overview of the structure of $\mathcal{E}_\phi$.

Upon its first activation with input (k, z), environment $(\mathcal{E}_\phi, 0)$ samples an integer $\ell \in \{0, \ldots, Q(k) - 1\}$ and proceeds as $(\mathcal{E}_\phi^\ell, 0)$ on input (k, z).

Roughly speaking, the goal of $(\mathcal{E}_\phi^\ell, 0)$ on input (k, z) is to simulate a (ϕ_k, ℓ, π_k)-execution of ρ_k with adversary $\widetilde{\mathcal{S}}_\rho$ and environment $\mathcal{E}_\rho$, with the exception that it omits the ITIs from the $(\ell + 1)$-th panoptic session of ϕ_k and the corresponding simulation of $\mathcal{S}_{\phi,\mathsf{s}_\phi}$ (where s_ϕ denotes the sid of the $(\ell+1)$-th panoptic session). This is is depicted in Figure 31. Note that, by definition, s_ϕ is the first extended session of ϕ_k not replaced in the $(\boldsymbol{\phi}, \ell, \boldsymbol{\pi})$-execution.

More formally, on input (k, z), $(\mathcal{E}_\phi^\ell, 0)$ simulates the execution described above, with the following exceptions:

$(\widetilde{\mathcal{S}}_\rho, \Diamond)$

// $(\widetilde{\mathcal{S}}_\rho, \Diamond)$ will simulate multiple modified adversaries $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$
// whose code is given in Figure 27

1 : At first activation, $(\mathcal{S}_\rho, \Diamond)$ reads the security parameter k
indicated in its dedicated tape, then acts as indicated
2 : At reception of an external-write w that is a backdoor from an ITI M :
3 : **checkorreject** the message is a pair $(\mathsf{ReplacedSession}, m)$ with
$\mathsf{ReplacedSession}$ a subset of $\{0,1\}^*$
4 : $w^{\mathsf{WithoutSet}} \leftarrow w$ with message replaced by m
5 : $\mathsf{positionM}_\phi \in \{\bot\} \cup \{0,1\}^* \leftarrow \mathsf{PSExpose}^\phi(k, M)$
6 : **if** $\mathsf{positionM}_\phi \in \{0,1\}^*$ **then** $\mathsf{Simulate}\left(w^{\mathsf{WithoutSet}}, \mathsf{positionM}_\phi\right)$
7 : **if** $\mathsf{positionM}_\phi = \bot$ **then send backdoor** "(m, M)" to the environment
8 : At reception of an external-write w that is an input (from the environment) :
9 : **checkorreject** the message is a pair $(\mathsf{ReplacedSession}, (m, M))$ with
$\mathsf{ReplacedSession}$ a subset of $\{0,1\}^*$
10 : m a message and M an ITI
11 : $w^{\mathsf{WithoutSet}} \leftarrow w$ with message replaced by m
12 : $\mathsf{positionM}_\pi \in \{\bot\} \cup \{0,1\}^* \leftarrow \mathsf{PSExpose}^\pi(k, M)$
13 : $\mathsf{positionM}_\phi \in \{\bot\} \cup \{0,1\}^* \leftarrow \mathsf{PSExpose}^\phi(k, M)$
14 : **if** $\mathsf{positionM}_\pi \in \{0,1\}^*$ **then**
15 : **if** $\mathsf{positionM}_\pi \notin \mathsf{ReplacedSession}$ **then** $\mathsf{Simulate}\left(w^{\mathsf{WithoutSet}}, \mathsf{positionM}\right)$
16 : **else send backdoor** "m" to M
17 : **else if** $\mathsf{positionM}_\phi \in \{0,1\}^*$ **then reject**
18 : **else then send backdoor** "m" to M

$\mathsf{Simulate}\,(w, \mathsf{s} \in \{0,1\}^*)$

1 : **checkorreject** w is an external-write
2 : **if** $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ has never been simulated **then**
3 : **initialize** its simulation (with security parameter k)
4 : $\widetilde{w} \leftarrow w$ with recipient replaced by $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$
5 : **simulate** applying $\widetilde{w}$ to the simulation of $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$
6 : Then simulation of $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ continues until one of the following events occurs :
if the simulation of $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ rejects **then reject**
7 : **if** the simulation of $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ deactivates (but does not reject) **then deactivate**
8 : **if** the simulation of $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ intends to send a backdoor **then**
9 : **send** this backdoor

Figure 29: Code of adversary $\widetilde{\mathcal{S}}_\rho$. The highlighted line shows the precise difference between $\widetilde{\mathcal{S}}_\rho$ and $\mathcal{S}_\rho$ (see Figure 26).

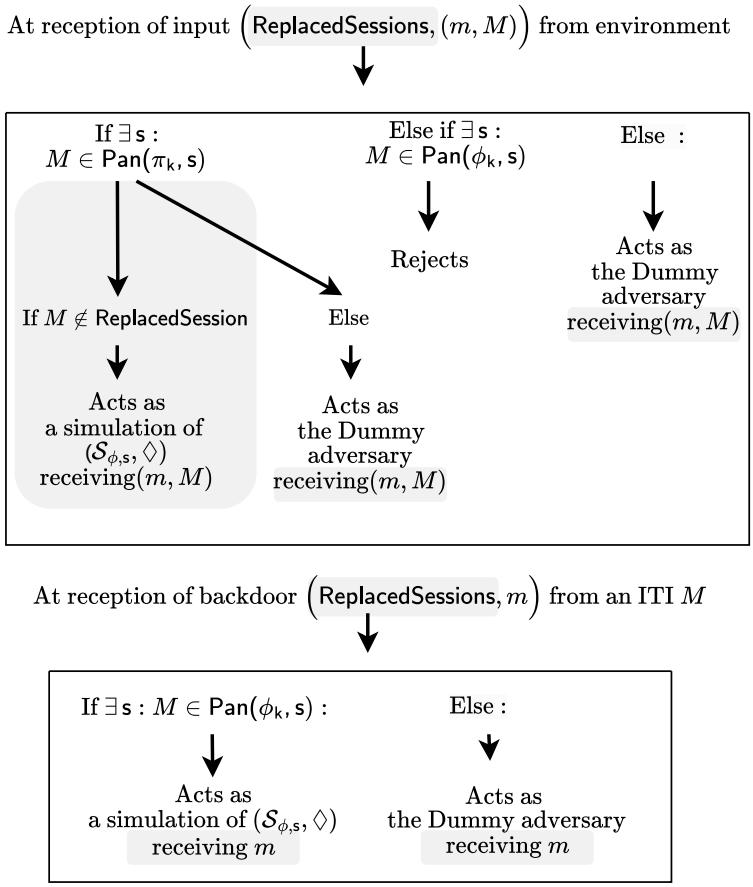


Figure 30: Simplified representation of the adversary $(\widetilde{\mathcal{S}}_\rho, \Diamond)$ for security parameter k. The gray area highlights the difference from $\mathcal{S}_\rho$.

- Inputs intended for main parties of $\mathsf{Pan}(\phi_k, \mathsf{s}_\phi)$ are not simulated, but instead sent directly by $\mathcal{E}_\phi$, impersonating the sender.
- If a simulated ITI N attempts to send an input to an ITI $M \in \mathsf{PanSub}(\phi_k, \mathsf{s}_\phi)$, the message is dropped and N deactivates.
- If a simulated ITI N attempts to send an output to an ITI $M \in \mathsf{Pan}(\phi_k, \mathsf{s}_\phi)$, the message is dropped and N deactivates.
- Messages from the protocol are interpreted as if coming from the $(\ell+1)$-th extended session of ϕ_k.
- Messages addressed to the simulation of $(\mathcal{S}_{\phi,\mathsf{s}_\phi}, \Diamond)$ are rerouted to the adversary.
- Messages from the adversary are interpreted as originating from the simulation of $(\mathcal{S}_{\phi,\mathsf{s}_\phi}, \Diamond)$.

To ensure that $\mathcal{E}_\phi$ runs in polynomial time, it maintains a counter that tracks the number of operations performed by the simulated ITIs. This counter does not account for the operations of $\mathcal{E}_\phi$ itself, but only for those performed by the ITIs being simulated, excluding any simulation overhead. If the counter exceeds $Q(k)$, the simulation halts and $\mathcal{E}_\phi$ returns 0.

Because, by construction, any (ϕ_k, ℓ, π_k)-execution of ρ_k with adversary $\widetilde{\mathcal{S}}_\rho$ and input $(k, z) \in \bigcup_{0<a\leq k^d}\{0,1\}^a$ completes within $Q(k)$ steps, the counter does not affect the simulation in such executions.

Lemma 4. *$\mathcal{E}_\phi$ is a ξ-identity bounded and PPT environment.*

Proof. The counter mechanism ensures that $\mathcal{E}_\phi$ is PPT. Moreover, since $\boldsymbol{\rho}$ is $(\boldsymbol{\pi}, \boldsymbol{\phi}, \xi)$-compliant, it satisfies Item 3 of Definition 10, and thus $\mathcal{E}_\phi$ is ξ-identity bounded. □

C.7 Proof of the First Auxiliary Lemma

In the demonstration of the first auxiliary lemma, we will count the extended sessions of ϕ_k by their order of existence[56]. By definition of the $(\boldsymbol{\pi}, \ell, \boldsymbol{\phi})$-execution, the first ℓ extended sessions of ϕ_k will be replaced by extended sessions of π_k. We will also use this ordering for panoptic sessions: for $u \geq 1$, the u-th panoptic session of ϕ_k is defined as the panoptic session corresponding to the u-th extended session of ϕ_k. Consequently, the first ℓ panoptic sessions of ϕ_k will be replaced by panoptic sessions of π_k.

Restatement of Lemma 2 (First Auxiliary Lemma)**.** *Let d be the constant described in Section C.4, whose existence follows from the assumption that Equation (6) is false.*

For $(k, z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d}\{0,1\}^a$ and $\ell \in \{0, \cdots, Q(k)-2\}$:

$$\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi^\ell}(k, z) = \mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi^{\ell+1}}(k, z)\,.$$

Proof. We will show that for $(k, z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d}\{0,1\}^a$ and $\ell \in \{0, \cdots, Q(k)-1\}$:

$$\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi^\ell}(k, z) = (\boldsymbol{\phi}, \ell, \boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\widetilde{\mathcal{S}}_\rho,\mathcal{E}_\rho}(k, z)\,, \tag{10}$$

$$\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi^\ell}(k, z) = (\boldsymbol{\phi}, \ell+1, \boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\widetilde{\mathcal{S}}_\rho,\mathcal{E}_\rho}(k, z)\,. \tag{11}$$

These two equalities together imply the lemma.

We proceed to prove the first equality; the second can be shown analogously.

Fix $(k, z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d}\{0,1\}^a$ and $\ell \in \{0, \ldots, Q(k)-1\}$.

For clarity, we temporarily disregard the counter mechanism defined in Line 1 of Figure 32, and defer its discussion to page 87.

To simplify terminology throughout this proof, we introduce the following notation:

[56] Note that an extended session becomes none-empty at the moment one of its main party is invoked by an input or output.

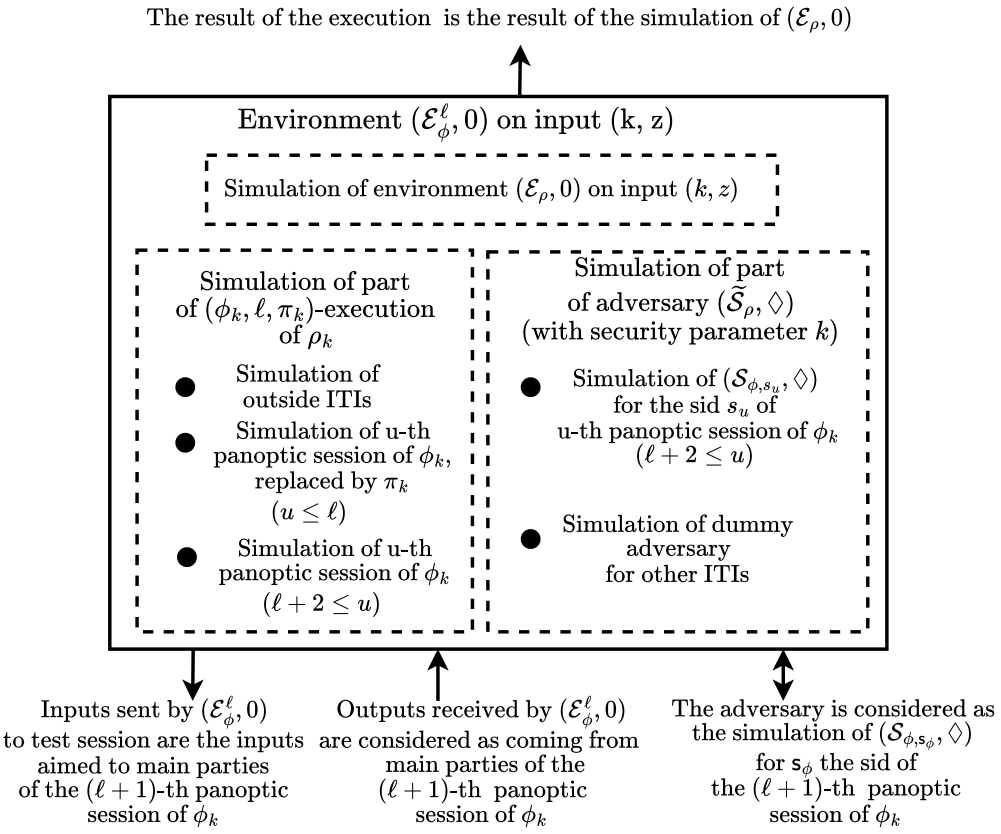


Figure 31: Simplified representation of the environment $(\mathcal{E}_\phi^\ell, 0)$. We recall that s_ϕ is the sid of the $(\ell + 1)$-th panoptic session of ϕ_k.

$(\mathcal{E}_\phi, 0)$ on entry (k, z)

Local variables: s_ϕ initialized to a symbol $\square \notin \{\bot\} \cup \{0,1\}^*$

// Note that the auxiliary function Simulation is defined in Figure 33

1 : Mechanism used to ensure $\mathcal{E}_\phi$ is a PPT adversary :

2 : During the execution of $\mathcal{E}_\phi$ on any input (k, z), a counter cpt tracks the total number of operations performed by all ITIs simulated by $\mathcal{E}_\phi$ (note that it does not count the operations of $\mathcal{E}_\phi$ itself, only those of the simulated ITIs)
If, at any point, $\mathsf{cpt} > Q(k)$, $\mathcal{E}_\phi$ stops and returns 0

3 : // Note that the result of $(\mathcal{E}_\phi, 0)$ is described in Line 15 of Simulation

4 : At first activation :

5 : $(\mathcal{E}_\phi, 0)$ reads the security parameter k

6 : $\ell \leftarrow\!\!\$\ \{0, \cdots, Q(k) - 1\}$

7 : Simulation $((\ell, k, z))$

8 : At reception of a backdoor "m" from the adversary :

// At this moment, s_ϕ will have been set by Simulation

// As shown by Line 28 of Figure 33

9 : Simulation (m)

10 : At reception of an output m from the protocol with identifier (s, p) and intended receiver an ITI M :

// by definition of protocol execution, note that s is the test sid

11 : Simulation $(\mathsf{m}, \mathsf{s}, \mathsf{p}, M)$

12 : At activation – not the first – with no message

// This happens in exactly three situations:

// - If a (real) ITI deactivates (including rejects).

// - If the (real) adversary sends a backdoor to a protocol ITI that does not exist.

// - The (real) control function disallows an external write operation.

13 : Simulation $()$

Figure 32: Main function of $\mathcal{E}_\phi$. For $\ell \in \{0, \ldots, Q(k)-1\}$, we denote by $\mathcal{E}_\phi^\ell$ the environment obtained by fixing the sample at Line 6 to ℓ.

Simulation(input)

1 : **if** input can be parsed as $(\ell, k, z) \in \mathbb{N} \times \mathbb{N} \times \{0,1\}^*$ **then**
2 : **initialize** the simulation of the (ϕ_k, ℓ, π_k)-execution of ρ_k
with environment $\mathcal{E}$ and adversary $\widetilde{\mathcal{S}}_\rho$ on input z
3 : Continue the simulation until one of the exceptions after line 14 occurs
4 : **else if** input can be parsed as a message $\mathsf{m} \in \{0,1\}^*$ **then**
5 : **simulate** the sending of backdoor m as if it came from
the simulation of $(\mathcal{S}_{\pi,\mathsf{s}_\phi}, \Diamond)$ by $\widetilde{\mathcal{S}}_\rho$
6 : Continue the simulation until one of the exceptions after line 14 occurs
7 : **else if** input can be parsed as $(\mathsf{m}, \mathsf{s}, \mathsf{p}, M) \in \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \times$ SetITI **then**
8 : **simulate** the sending of output m to the simulation of M as if it came from $(\phi, (\mathsf{s}, \mathsf{p}))$
9 : Continue the simulation until one of the exceptions after line 14 occurs
10 : **else if** Simulation has been called with no input **then**
11 : **simulate** the activation of $(\mathcal{E}, 0)$ with no message and continue the simulation
12 : until one of the exceptions after line 14 occurs
13 : **else halt**
14 : **List of exceptions:**
15 : **exception** : simulation of $(\mathcal{E}, 0)$ returns a bit $b \in \{0,1\}$ **then return** b
16 : **exception** : an input or output "m" is sent by a simulated protocol ITI M
to a protocol ITI N that is a main party of
the $(\ell+1)$-th extended session of ϕ_k **then**
// An extended session is counted at the moment a main-party receives an input or an output
17 : **if** $\mathsf{s}_\phi = \square$ **then** $\mathsf{s}_\phi \leftarrow \mathsf{sid}(N)$
18 : **if** it is an input **then**
19 : **send input** "m" to the protocol with same sid and pid than N
20 : **if** it is an output **then**
21 : **simulate** the deactivation of M and continue the simulation
22 : until one of the exceptions after line 14 occurs
23 : **exception** : an input or output "m" is sent by a simulated protocol ITI M
to a protocol ITI N with code $\neq \phi_k$
such that $\mathsf{PSExpose}^{\phi}(k, N) = \mathsf{s}_\phi$ **then**
24 : **simulate** the deactivation of M and continue the simulation
25 : until one of the exceptions after line 14 occurs
26 : **exception** : the simulation of $\widetilde{\mathcal{S}}_\rho$ intends to simulate an input
27 : "m" to its simulation of $\mathcal{S}_{\phi,\mathsf{s}_\phi}$ **then**
28 : // Note that, this can only happen if $\mathsf{s}_\phi \neq \square$
29 : // As a consequence, s_ϕ is set before first activation of the adversary
30 : **send input** "m" to the adversary

Figure 33: Definition of the function Simulation used by $\mathcal{E}_\phi$.

- The *R-execution* ("R" for Real) refers to the (ϕ_k, ℓ, π_k)-execution of ρ_k with environment $\mathcal{E}_\rho$ and adversary $\widetilde{\mathcal{S}}_\rho$, on input (k, z).
- The *S-execution* ("S" for Simulated) refers to the execution of ϕ_k with environment $\mathcal{E}_\phi^\ell$ and adversary $\mathcal{S}_\phi$, on input (k, z).

Let s_ρ be the test session identifier in the R-execution, and s_ϕ the test sid in the S-execution.

Our goal is to establish a tight correspondence between the two executions such that the outputs of the real environment $(\mathcal{E}_\rho, 0)$ in the R-execution and the simulated version of $(\mathcal{E}_\rho, 0)$ in the S-execution are identically distributed.

Both executions consist of a combination of real and simulated ITIs, grouped into eight classes ① through ⑧ for the R-execution and corresponding classes Ⓘ through Ⓥ𝖨𝖨𝖨 for the S-execution. These groupings are illustrated in Figure 34.

We adopt the following convention for the remainder of this proof: the group ④ (respectively, Ⓘ𝖵) consists precisely of the ITIs forming a panoptic session—specifically, the session $\mathsf{Pan}(\phi_k, \mathsf{s}_\phi)$. Accordingly, when referring to the "extended session", "main parties", "sub-parties", "panoptic session", or "panoptic sub-parties" of ④ (respectively, Ⓘ𝖵), we mean the corresponding elements of the panoptic session represented by ④ (respectively, Ⓘ𝖵).

In an ideal setting, every message exchanged in the R-execution would have a direct counterpart in the S-execution. However, this is not the case. For example, in the R-execution, an ITI $M \in$ ② may send an input to a panoptic sub-party $N \in$ ④. In the S-execution, however, the corresponding ITI $M \in$ Ⓘ𝖨 is simulated by $(\mathcal{E}_\phi^\ell, 0)$, and N is a panoptic sub-party of Ⓘ𝖵. Due to the protocol execution rules, $(\mathcal{E}_\phi^\ell, 0)$ is not permitted to send inputs to ITIs in $\mathsf{PanSub}(\phi_k, \mathsf{s}_\phi)$ – in particular, not to N.

To guarantee that such mismatches do not impact the outcome of the execution, we must verify that, in each of these cases, the R- and S-executions produce behavior that is indistinguishable from the point of view of $(\mathcal{E}_\rho, 0)$. In particular, we will frequently rely on the observation that, from the point of view of $(\mathcal{E}_\rho, 0)$, there is no difference between an ITI M deactivating and M sending a message to an ITI N that rejects it.

Accordingly, we must carefully examine all situations in which the correspondence between the executions is imperfect, and ensure that none of these discrepancies affect the outcome as observed by $(\mathcal{E}_\rho, 0)$. To this end, we analyze how the correspondence is established for each communication of the form $\textcircled{i} \to \textcircled{j}$ in the R-execution, where $i, j \in \{1, \ldots, 8\}$ and $i \neq j$.

We begin by noting that communications between ITIs belonging to any of the groups ①, ②, ③, ⑤, ⑦, and ⑧ can be simulated perfectly by the S-execution, since the corresponding ITIs in the S-execution (Ⓘ, Ⓘ𝖨, Ⓘ𝖨𝖨, Ⓥ, Ⓥ𝖨𝖨, Ⓥ𝖨𝖨𝖨) are all simulated by $(\mathcal{E}_\phi^\ell, 0)$.

This leaves us with the following four cases, which require careful analysis:

case 1: Messages *from* ④ to other groups versus messages from Ⓘ𝖵.

case 2: Messages *to* ④ versus messages to Ⓘ𝖵.

case 3: Messages *from* the simulated adversary ITI ⑥ to other groups versus those from Ⓥ𝖨.

case 4: Messages *to* ⑥ from other ITIs versus those to Ⓥ𝖨.

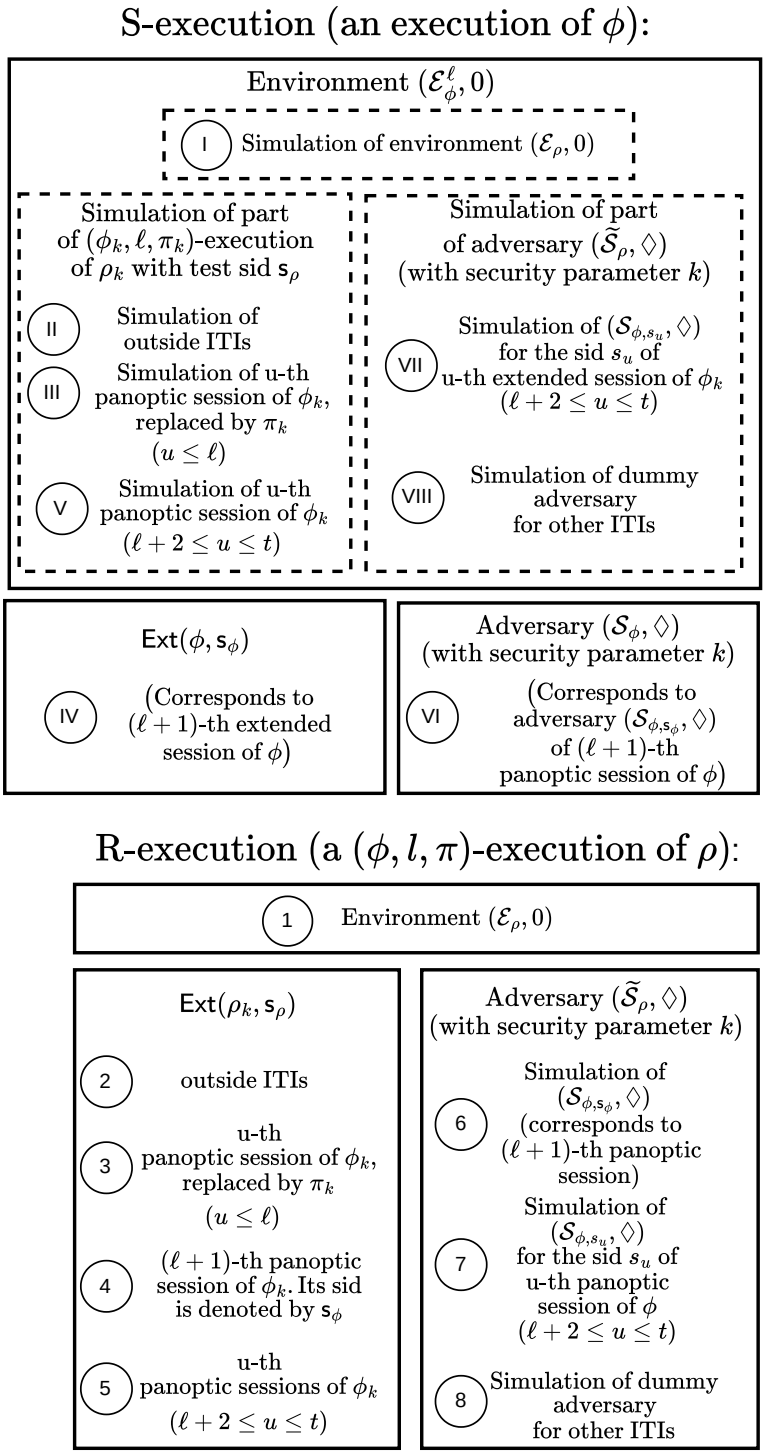


Figure 34: The upper figure is a representation of a S-execution, with the groups $\overline{(I)}$ to $\overline{(VIII)}$. The lower figure is a representation of a R-execution, with the groups (1) to (8).

Case 1: Correspondence for Messages Sent by ④ and (IV)

We now examine the messages sent by ITIs in ④ (in the R-execution) and the corresponding behavior of ITIs in (IV) (in the S-execution). The only types of such communications are:

case1.i Input to an ITI in ②, ③, or ⑤.

case1.ii Output to an ITI in ②, ③, or ⑤.

case1.iii Backdoor to the adversary (an ITI in ⑥, ⑦, or ⑧).

case1.iv An ITI of ④ deactivates, causing the environment to be activated with no message.

case1.v An ITI of ④ performs an external write that is disallowed by the control function, causing the environment to be activated with no message.

Argument for case1.i:

This type of communication cannot occur, due to the fact that ϕ_k is sub-party respecting. More specifically, this follows from Item spr.6 of Definition 9.

Argument for case1.ii:

In the R-execution, the sub-party respecting property of ϕ_k ensures that only main parties of $\mathsf{Ext}(\phi_k, \mathsf{s}_\phi)$ are allowed to send such outputs (Item spr.5 of Definition 9).

In the S-execution, the corresponding outputs are redirected to the environment $(\mathcal{E}^\ell_\phi, 0)$, which then simulates their delivery to the intended ITIs in (II), (III), or (V). This reproduces the same outcome from the perspective of the receiving ITI.

Argument for case1.iii:

We will also use this analysis to address Item case2.iii. Specifically, we examine the following two types of backdoor communications:

- Backdoors sent by ④ to the adversary (that is, to ⑥, ⑦, or ⑧),
- Backdoors sent by the adversary (⑥, ⑦, or ⑧) to ④.

In the R-execution, $\widetilde{\mathcal{S}}_\rho$ internally simulates a collection of adversaries $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ – one for each panoptic session $\mathsf{Pan}(\phi_k, \mathsf{s})$ that is replaced by a panoptic session $\mathsf{Pan}(\pi_k, \mathsf{s})$.

Communications between $\widetilde{\mathcal{S}}_\rho$ and these simulated adversaries are handled in such a way that each $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ "believes" it is operating as a real adversary. This is achieved through the simulation of backdoors and inputs.

In the S-execution, the environment $(\mathcal{E}^\ell_\phi, \Diamond)$ simulates the full behavior of $\widetilde{\mathcal{S}}_\rho$, with the sole exception of the specific adversary $(\mathcal{S}_{\phi,\mathsf{s}_\phi}, \Diamond)$, which is instead realized as the actual adversary $(\mathcal{S}_\phi, \Diamond)$ of the S-execution.

There are four potential flaws in this simulation, which we now examine in detail:

- *Flaw 1:* In the S-execution, the adversary $(\mathcal{S}_\phi, \Diamond)$ is used in place of $(\mathcal{S}_{\phi,\mathsf{s}_\phi}, \Diamond)$. $(\mathcal{S}_\phi, \Diamond)$ is permitted to send backdoors to any ITI that exists in the execution, whereas $(\mathcal{S}_{\phi,\mathsf{s}_\phi}, \Diamond)$ restricts backdoor transmissions to ITIs within $\mathsf{Pan}(\phi_k, \mathsf{s}_\phi)$. Using Propositions 14 and 16, we can see that the behaviors of the two executions coincide.
- *Flaw 2:* In the S-execution, there is no mechanism to simulate backdoors from (IV) to (VII) or from (IV) to (VIII). However, by the definition of $\widetilde{\mathcal{S}}_\rho$, any backdoor sent by an ITI $M \in$ ④ is directed to ⑥, the simulation of $\mathcal{S}_{\phi,\mathsf{s}_\phi}$. As a result, there are no backdoors from ④ to ⑦ or from ④ to ⑧.
- *Flaw 3:* In the S-execution, there is no mechanism for simulating backdoors from (VII) to (IV). However, backdoors from ⑦ to ④ do not occur in the R-execution either. Indeed, for any $\mathsf{s} \neq \mathsf{s}_\phi$, the simulated adversary $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ only sends backdoors to ITIs in $\mathsf{Pan}(\phi_k, \mathsf{s})$. Since ϕ_k is separated, it follows that $\mathsf{Pan}(\phi_k, \mathsf{s}) \cap$ ④ $= \varnothing$ (recall that ④ $= \mathsf{Pan}(\phi_k, \mathsf{s}_\phi)$). Hence, no backdoor from ⑦ to ④ arises.

- *Flaw 4:* In the S-execution, there is no mechanism for simulating backdoors from (VIII) to (IV). However, backdoors from ⑧ to ④ do not occur in the R-execution either. By the definition of $\widetilde{\mathcal{S}}_\rho$, such a backdoor would correspond to an input of the form $(\mathsf{ReplacedSession}, (m, M))$ received by $\widetilde{\mathcal{S}}_\rho$, where $M \in \mathsf{Pan}(\phi_k, \mathsf{s}_\phi)$. But $\widetilde{\mathcal{S}}_\rho$ explicitly rejects all such inputs when $M \in \mathsf{Pan}(\phi_k, \tilde{\mathsf{s}})$ for some sid $\tilde{\mathsf{s}}$. Therefore, no backdoor from ⑧ to ④ can occur.

Argument for case1.iv:

This is perfectly simulated in the S-execution. Indeed, when an ITI of (IV) deactivates, the environment $(\mathcal{E}^l_\phi, 0)$ simulates the activation of $(\mathcal{E}_\rho, 0)$ with no message.

Argument for case1.v:

This is perfectly simulated in the S-execution. Indeed, when the control function disallows an external operation, the environment $(\mathcal{E}^l_\phi, 0)$ simulates the activation of (I) (*i.e.* $(\mathcal{E}_\rho, 0)$) with no message.

Case 2: Correspondence for Messages Sent to ④ and (IV)

We now consider communications from an ITI outside ④ to an ITI within ④. These interactions fall into the following categories:

case2.i Inputs from an ITI in ②, ③, or ⑤.

case2.ii Outputs from an ITI in ②, ③, or ⑤.

case2.iii Backdoors from the adversary (originating from ⑥, ⑦, or ⑧).

Argument for case2.i:

In the R-execution, ITIs belonging to ②, ③, or ⑤ may send inputs to any ITI within the panoptic session ④. However, since ϕ_k is both sub-party respecting and separated, only inputs addressed to main parties of ④ are accepted. All others are rejected in accordance with Item spr.3 of Definition 9. Furthermore, these rejections cannot be exploited by the adversary to inject backdoors, as guaranteed by Item spr.1.

In the S-execution, the protocol execution rules ensure that simulated ITIs in (II), (III), or (V) may only send inputs to main parties of (IV). Any attempt to send such an input to a panoptic sub-party of (IV) results in a deactivation.

The manner this deactivation is made depends on whether the environment $(\mathcal{E}^\ell_\phi, \Diamond)$ has already determined the value of the test sid s_ϕ. This identification occurs only when a simulated ITI first attempts to interact with a main party of (IV).

Let N be a simulated ITI in (II), (III), or (V), and suppose N tries to send an input to a panoptic sub-party $M \in \mathsf{PanSub}(\phi_k, \mathsf{s}_\phi) \subset$ (IV). Two cases must be considered:

- If s_ϕ is already known as the test sid, then $(\mathcal{E}^\ell_\phi, \Diamond)$ directly simulates the deactivation of N.
- If s_ϕ has not yet been determined as the test sid, the simulation proceeds and send the input to a simulation of M. Since $\boldsymbol{\phi}$ is sub-party respecting and separated, the message is rejected, and $(\mathcal{E}^\ell_\phi, 0)$ then simulates the deactivation of M.

Thus, in both the R-execution and the S-execution, the result is the same from the point of view of $(\mathcal{E}_\rho, 0)$.

Argument for case2.ii:

In the R-execution, ITIs belonging to ②, ③, or ⑤ may send outputs to any ITI within the panoptic session ④. However, since ϕ_k is sub-party respecting and separated, each of these outputs are rejected. This follows from Item spr.4 of Definition 9. Moreover, as guaranteed by Item spr.1, these rejections cannot be exploited by the adversary to transmit backdoors.

In the S-execution, the execution rules ensure that simulated ITIs from (II), (III), or (V) cannot send outputs to ITIs of (IV). Any attempt to send such an output results in a deactivation.

The manner this deactivation is made depends on whether the environment $(\mathcal{E}_\phi^\ell, \Diamond)$ has already determined the value of the test sid s_ϕ.

Let N be a simulated ITI in (II), (III), or (V), and suppose N tries to send an output to an ITI $M \in$ (4). Three cases arise:

- If $M \in \mathsf{MainParty}(\phi_k, \mathsf{s}_\phi)$, then the environment necessarily knows that s_ϕ is the test sid, and it simulates the deactivation of M.
- If $M \in \mathsf{ExtSub}(\phi_k, \mathsf{s}_\phi)$ and it is already known that s_ϕ is the test sid, the environment simulates the deactivation of M.
- If $M \in \mathsf{ExtSub}(\phi_k, \mathsf{s}_\phi)$ and s_ϕ is not yet known as the test sid, the simulation proceeds and send the output to a simulation of M. Since $\boldsymbol{\phi}$ is sub-party respecting and separated, the message is rejected, and $(\mathcal{E}_\phi^\ell, 0)$ then simulates the deactivation of M.

Thus, in both the R-execution and the S-execution, the result is the same from the point of view of $(\mathcal{E}_\rho, 0)$.

Argument for case2.iii:

This case concerns backdoors from the adversary to (4). It is already treated by the argument for case1.iii (page 84).

Argument for case2.iii:

This case concerns backdoors sent by the adversary (*i.e.*, from (6), (7), or (8)) to ITIs of (4). This is addressed in the argument for case1.iii (page 84).

Case 3: Correspondence for Messages Sent by (6) and (VI)

We now consider communications from (6) to an ITI distinct of (6). These interactions fall into the following categories:

case3.i Backdoors to the environment (1).

case3.ii Backdoors to protocol ITIs (i.e., ITIs in (2), (3), (4), or (5)).

case3.iii If (6) sends a backdoor to an ITI that does not exist, this results in the activation of the environment with no message.

case3.iv If (6) attempts an external write that is disallowed by the control function, this results in the activation of the environment with no message.

Argument for case3.i:

As explained in the argument for case1.iii on page 84, all backdoor communications from (6) to the environment (1) in the R-execution are perfectly simulated in the S-execution by communications from (VI) to (I).

Argument for case3.ii:

In the R-execution, any adversary $(\mathcal{S}_{\phi,\mathsf{s}}, \Diamond)$ can only send backdoors to ITIs in the panoptic session $\mathsf{Pan}(\phi_k, \mathsf{s})$. In particular, $(\mathcal{S}_{\phi,\mathsf{s}_\phi}, \Diamond)$ can only send backdoors to ITIs within $\mathsf{Pan}(\phi_k, \mathsf{s}_\phi)$.

In the S-execution, the adversary $(\mathcal{S}_\phi, \Diamond)$ is not similarly restricted, but the protocol execution rules guarantee that any backdoor not targeting the environment or an ITI in $\mathsf{Ext}(\phi_k, \mathsf{s}_\phi)$ is disallowed.

The only potential discrepancy arises when $(\mathcal{S}_{\phi,\mathsf{s}_\phi}, \Diamond)$ sends a backdoor to an ITI $M \in \mathsf{Pan}(\phi_k, \mathsf{s}_\phi) \setminus \mathsf{Ext}(\phi_k, \mathsf{s}_\phi)$, which is permitted in the R-execution. However, Proposition 14 ensures that such an M must be sleeping, and Item spr.1 of the sub-party respecting definition guarantees that the message is rejected. Thus, the behaviors are indistinguishable from the point of view of $(\mathcal{E}_\rho, 0)$.

Argument for case3.iii:

Assume that (6) sends a backdoor to a nonexistent ITI M. By definition of protocol execution, and since the adversary always unsets forced-write, the environment (1) $= (\mathcal{E}_\rho, 0)$ is activated with no message in the R-execution.

In the S-execution, the simulation mirrors this behavior.

If $M \notin \mathsf{Pan}(\phi, \mathsf{s}_\phi)$, by definition, $(\mathcal{S}_{\phi,\mathsf{s}_\phi}, \Diamond)$ will not send the backdoor and reject.

We now suppose that $M \in \mathsf{Pan}(\phi, \mathsf{s}_\phi)$.

From the argument in case2.i (page 85), the set of awake ITIs in ④ and $\overline{IV}$ coincide, so M is also sleeping in the S-execution.

Two cases arise:

- If M exists but is sleeping, the message is rejected by Item spr.1 of the sub-party respecting property. This leads to the activation of $(\mathcal{E}_\phi^\ell, 0)$ with no message, which then simulates the activation of $\overline{I} = (\mathcal{E}_\rho, 0)$ with no message.
- If M does not exist at all, the adversary again unsets forced-write, and $(\mathcal{E}_\phi^\ell, 0)$ activates with no message, simulating an activation of $\overline{I} = (\mathcal{E}_\rho, 0)$. with no message.

In both cases, the outcome is an activation of $(\mathcal{E}_\rho, 0)$ with no message, matching the R-execution.

Argument for case3.iv:

This situation is perfectly simulated in the S-execution. Whenever the control function disallows an external write in the R-execution, the environment ① $= (\mathcal{E}_\rho, 0)$ is activated with no message. The same happens in the S-execution: $(\mathcal{E}_\phi^\ell, 0)$ simulates this activation of $\overline{I} = (\mathcal{E}_\rho, 0)$ with no message.

Case 4: Correspondence for Messages Sent to ⑥ and $\overline{VI}$

We now examine the communications received by the simulated adversary ⑥ $= (\mathcal{S}_{\phi,\mathsf{s}_\phi}, \Diamond)$ from ITIs outside ⑥. These fall into two categories:

case4.i Inputs from the environment ① (i.e., inputs sent to $(\widetilde{\mathcal{S}}_\rho, \Diamond)$ that are internally routed to the simulation of $(\mathcal{S}_{\phi,\mathsf{s}_\phi}, \Diamond)$).

case4.ii Backdoors from protocol ITIs (i.e., backdoors sent to $(\widetilde{\mathcal{S}}_\rho, \Diamond)$ by ITIs in ②, ③, ④, or ⑤, that are internally routed to the simulation of $(\mathcal{S}_{\phi,\mathsf{s}_\phi}, \Diamond)$).

Argument for case4.i:

As explained in the argument for case1.iii page 84, the communications between ① and ⑥ are perfectly simulated by the communications between $\overline{I}$ and $\overline{VI}$.

Argument for case4.ii:

In the R-execution, $\widetilde{\mathcal{S}}_\rho$ only forwards backdoors to ⑥ $= (\mathcal{S}_{\phi,\mathsf{s}_\phi}, \Diamond)$ if they originate from an ITI $M \in \mathsf{Pan}(\phi_k, \mathsf{s}_\phi)$.

In the S-execution, the only ITIs authorized to send backdoors to $\overline{VI} = (\mathcal{S}_\phi, \Diamond)$ are those belonging to $\mathsf{Ext}(\phi_k, \mathsf{s}_\phi)$.

These two cases are equivalent. Proposition 16 guarantees that $\mathsf{Ext}(\phi_k, \mathsf{s}_\phi) \subset \mathsf{Pan}(\phi_k, \mathsf{s}_\phi)$, and Proposition 14 ensures that any ITI in the set difference $\mathsf{Pan}(\phi_k, \mathsf{s}_\phi) \setminus \mathsf{Ext}(\phi_k, \mathsf{s}_\phi)$ is necessarily sleeping. Since sleeping ITIs cannot send messages, there is no loss in simulation fidelity.

Addition of the counter:

We now argue that adding a counter to $\mathcal{E}_\phi^\ell$—as specified in Line 1 of Figure 32—has no effect on the correspondence between R- and S-executions.

By construction of the polynomial Q and the assumption that $(k, z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d} \{0,1\}^a$ with $0 \leq \ell \leq Q(k)$, we know that every $(\boldsymbol{\phi}, \ell, \boldsymbol{\pi})$-execution of ρ_k with adversary $\widetilde{\mathcal{S}}_\rho$ and environment $\mathcal{E}_\rho$ on input (k, z) terminates within fewer than $Q(k)$ computational steps.

In particular, this implies that all R-executions are bounded by $Q(k)$ steps. Since we have established a faithful correspondence between R-executions and S-executions, the same bound applies to the total number of simulated steps performed by $\mathcal{E}_\phi^\ell$.

Hence, the counter within $\mathcal{E}_\phi^\ell$ is never exceeded in any of the executions under consideration. The presence of the counter has no effect on the behavior of the simulation and does not alter the outcome of the execution.

This completes the proof □

C.8 Proof of the Second Auxiliary Lemma

Restatement of Lemma 3 (Second Auxiliary Lemma)**.** *Let d be the constant described in Section C.4, whose existence follows from the assumption that Equation (6) is false. For $(k,z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d}\{0,1\}^a$:*

$$\mathsf{EXEC}_{\boldsymbol{\phi},\mathcal{S}_\phi,\mathcal{E}_\phi^0}(k,z) = \mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{S}_\rho,\mathcal{E}_\rho}(k,z)\,,$$
$$\mathsf{EXEC}_{\boldsymbol{\pi},\mathcal{D},\mathcal{E}_\phi^{Q(k)-1}}(k,z) = (\boldsymbol{\phi},\infty,\boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{D}^{\mathsf{DROP}},\mathcal{E}_\rho}(k,z)\,.$$

Proof. Let us fix $(k,z) \in \mathbb{N} \times \bigcup_{0<a\leq k^d}\{0,1\}^a$.
Thanks to Equations (10) and (11), established in the proof of Lemma 2, it suffices to prove:

$$(\boldsymbol{\phi},0,\boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\widetilde{\mathcal{S}}_\rho,\mathcal{E}_\rho}(k,z) = \mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{S}_\rho,\mathcal{E}_\rho}(k,z)\,,$$
$$(\boldsymbol{\phi},Q(k),\boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\widetilde{\mathcal{S}}_\rho,\mathcal{E}_\rho}(k,z) = (\boldsymbol{\phi},\infty,\boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{D}^{\mathsf{DROP}},\mathcal{E}_\rho}(k,z)\,.$$

First equation:

In any $(\boldsymbol{\phi},0,\boldsymbol{\pi})$-execution of $\boldsymbol{\rho}$, the set ReplacedSession is empty by definition. Comparing the algorithms of $\mathcal{S}_\rho$ and $\widetilde{\mathcal{S}}_\rho$ (Figures 26 and 29, respectively), we can directly conclude.

Second equation:

By construction of Q, we know that any $(\boldsymbol{\phi},Q(k),\boldsymbol{\pi})$-execution of $\boldsymbol{\rho}$ on input (k,z) halts in fewer than $Q(k)$ computational steps. This implies that all extended sessions of ϕ_k are replaced by corresponding sessions of π_k within this time bound. Hence, such executions are equivalent to the $(\boldsymbol{\phi},\infty,\boldsymbol{\pi})$-executions of $\boldsymbol{\rho}$.

Furthermore, from the definition of $\widetilde{\mathcal{S}}_\rho$, we observe that when all sessions of ϕ_k have been replaced, $\widetilde{\mathcal{S}}_\rho$ behaves exactly like $\mathcal{D}^{\mathsf{DROP}}$. The result follows. □

D Additional Issues, Limitations, Weaknesses and Discussions

This appendix addresses additional issues, limitations, weaknesses, and discussions concerning Plain UC, with all identified problems listed in Table 4[57]. They constitute supplementary material that is not necessary to understand the article but valuable in its own right.

The first two sections outline an issue and a limitation related to the UC operator and the subroutine respecting property, respectively. Despite being connected to previously discussed notions, they offer innovative insights.

Section D.3 highlights a weakness in the Plain UC UC Theorem proof, justifying the simulator modification in the Updated UC proof.

Section D.4 examines the possibility in Plain UC to specify the receiver of an external-write via a predicate rather than its extended identity. While appealing, this approach leads to multiple issues, which is why it was excluded from Updated UC.

[57] Note that the category "other" used in the table refers to a weakness in the proof of the UC Theorem that is not directly related to any specific concept of UC. It addresses a deficiency in the definition of an adversary.

Section D.5 explores the notion of time-complexity in Plain UC and details the flaws that warranted our use of a different concept for Updated UC.

Lastly, Section D.6 identifies a weakness tied to the balanced environment notion, excluded in Updated UC for this reason.

Table 4: All Issues, limitations and weaknesses described in the article, with a highlight on those detailed in this section.

Notions	Related Issues, limitations or weaknesses
Extended Session and Subroutine Respecting	(L.1) (I.1) (I.2) (I.3) (Section 5.1) (L.3) (Appendix D.2.1)
UC Operator	(I.4) (W.1) (Section 5.2) (I.5) (Appendix D.1.1), (I.6) (Appendix D.1.2)
Subroutine Exposing	(L.2) (W.2) (Section 5.3)
Compliance	(W.3) (W.4) (Section 5.4)
Use of Predicates to Designate Receiver	(I.7) (L.4) (Appendix D.4)
Time Complexity	(W.6) (W.7) (Appendix D.5)
Balanced Environment	(W.8) (Appendix D.6)
Other	(W.5) (Appendix D.3)

D.1 Two additional Issues Regarding the UC Operator

D.1.1 (I.5) Structured Protocols' sid and pid Should Match Their Body.

In Plain UC, for structured protocols, such as the UC operator $\rho^{\phi\to\pi}$, it is indicated that their sid and pid to not match those of their body. For example, as described in [Can20, (Section 5.1)]:

> The extended identity of μ represents the extended identity of the shell followed by that of the body.

However, this discrepancy presents a direct issue, which we briefly describe here. For instance, consider $(\rho, (\mathsf{s}, \mathsf{p}))$ which halts for any $(\mathsf{s}, \mathsf{p}) \neq (0, 0)$ and outputs "0" when $(\rho, (0, 0))$ receives an input. An environment sending an input to the protocol with identifier $(0, 0)$ and checking for an output could distinguish between ρ (where output is received) and $\left(\rho^{\phi\to\pi}, (0, 0)\right)$ (where output is not received because it doesn't simulate $(\rho, (0, 0))$).

D.1.2 (I.6) The Shell-messages Lead to a Distinguisher.

We can construct simple protocols ρ such that for any protocols ϕ and π distinct of ρ, $\rho^{\phi\to\pi}$ does not UC-emulate ρ. The problem arises when the environment sends a message $(\mathsf{m}_{\mathsf{body}}, \mathsf{m}_{\mathsf{shell}})$ to the protocol ($\rho^{\phi\to\pi}$ or ρ) with the correct impersonation. The shell of the protocol $\rho^{\phi\to\pi}$ will interpret this as a valid message and forward $\mathsf{m}_{\mathsf{body}}$ to its body ρ. However, ρ does not expect a shell-part $\mathsf{m}_{\mathsf{shell}}$ and will reject the message.

We think this Issue as a large impact as it can occur with any protocols ϕ and π. In this section, we present a comprehensive counterexample for Issue (I.6).

Detailed counterexample:

We consider the protocol ρ defined in Figure 35 and a subroutine exposing subroutine

respecting protocol π. Moreover, we take a subroutine exposing subroutine respecting protocol $\phi \notin \{\pi, \rho, \mathsf{code}(N)\}$ and such that π UC-emulates ϕ.

N is an ITI with code $\neq \rho$

(ρ, id)

1 : **if** $\mathsf{id} \neq (0,0)$ **then halt**
2 : At first activation:
3 : **if** it is an input with message "0" **then**
4 : **send output** "0" to N
5 : **else reject**
6 : For other activations:
7 : **reject**

Figure 35: Protocol ρ used for Issue (I.6).

It is clear that ρ is (π, ϕ)-compliant and that π UC-emulates ϕ. The UC theorem of Plain UC should thus apply and ensure that $\rho^{\phi \to \pi}$ UC-emulates ρ. Nevertheless, this conclusion is false, as shown by the next proposition.

Proposition 17. *$\rho^{\phi \to \pi}$ does not UC-emulate ρ.*

Proof. We recall that the UC operator of Plain UC is defined in [Can20, (Section 6.1)]. We consider the environment $\widetilde{\mathcal{E}}$ defined in Figure 36. We can see that, for any adversary $\mathcal{A}$:

- In the execution of $\rho^{\phi \to \pi} = (\rho, \sigma_{\pi,\phi})$ with $\widetilde{\mathcal{E}}$ and the dummy adversary, using the definition of the replacement operator [Can20, (Section 6.1)], we see that:
  - At reception of input "$(0, (\texttt{Shell:main:}\ \rho, 0\texttt{;}\ \texttt{subr:}\ \bot))$", the conditions 1.(a).i then 1.(b).ii of the definition of UC operator [Can20, (Section 6.1)] are verified and applied. As a result, the message 0 is passed to the body $(\rho, (0,0))$, with sender identity $(\rho, (0,1))$.
  - By definition of ρ, the body wants to send output 0 to N.
  - The condition 2.(c) of the definition of UC operator is verified and applied, leading to $(\rho, \sigma_{\pi,\phi})$ sending output 0 to N.
  - Because N does not exist in the extended session $((\rho, \sigma_{\pi,\phi}), 0)$ and that it is an output, $(\widetilde{\mathcal{E}}, 0)$ receives the message 0 with the information that it was sent to N.
  - As the second activation of $(\widetilde{\mathcal{E}}, 0)$ is not due to the deactivation of an ITI, $(\widetilde{\mathcal{E}}, 0)$ returns 1.
- In the execution of ρ with environment $\widetilde{\mathcal{E}}$ and adversary $\mathcal{A}$, $(\rho, (0,0))$ rejects the input "$(0, (\texttt{Shell:main:}\ \rho, 0\texttt{;}\ \texttt{subr:}\ \bot))$" sent by $(\widetilde{\mathcal{E}}, 0)$ because it is different of "0". As a result, $(\widetilde{\mathcal{E}}, 0)$ returns 0.

$(\widetilde{\mathcal{E}}, 0)$

1 : At first activation:
2 : **send input** "$(0, (\texttt{Shell:main:}\ \rho, 0;\ \texttt{subr:}\ \perp))$"
to the ITI with $\mathsf{id} = (0, 0)$
impersonating the structured ITI $((\rho, \sigma_{\pi,\phi}), (0, 1))$
3 : At second activation:
4 : **if** it is activated with no message **then**
// This condition occurs if:
// - An ITI deactivates itself.
// - The adversary sends a message to a non-existent ITI.
// - The control function disallows an external write.
return 0
5 : **else return** 1

Figure 36: Environment $\widetilde{\mathcal{E}}$ used for Issue (I.6).

□

D.2 One additional Limitation Regarding the Subroutine Respecting property

D.2.1 (L.3) The Case of "Unusual Sub-parties".

Considering θ the prefix of execution of some protocol and (π, s) the extended session of some protocol π in this prefix. We define an "unusual sub-party" as an ITI $M \in \mathsf{Ext}(\pi, \mathsf{s})$ that is neither a main party nor a subsidiary of π (the notion of subsidiary is defined in [Can20, (Section 3.1.2, Paragraph "Subroutines, etc.")] while the notion of sub-party is defined in [Can20, (Section 3.1.2, Paragraph "Protocol sessions.")]). This can happen, for example, when M is invoked with an output by a sub-party.

In Plain UC, some extra-restrictions are given to unusual sub-parties in the subroutine respecting property [Can20, (Definition 19)]:

- Any input sent by an unusual sub-party to a sub-party must be rejected.
- Any output sent by an unusual sub-party to a member of the extended session must be rejected.

This limits the expressiveness of the model.

In the sub-party respecting property of Updated UC (Definition 9), "unusual parties" are not considered differently than other sub-parties.

D.3 A Missing Case in the Simulator of the Proof of UC Theorem of Plain UC (Weakness (W.5))

In this section, we describe a situation where the simulator used in the proof of the UC theorem of Plain UC [Can20, (Theorem 22)] should reject an input from the environment instead of acting as the dummy adversary.

This issue is intended for readers familiar with Plain UC and the proof of the UC Theorem in Plain UC [Can20, Theorem 22)]. For simplicity, we do not consider imports.

Recall that in this proof, an adversary $\mathcal{S}$ for ρ is defined in [Can20, (Figure 10)].

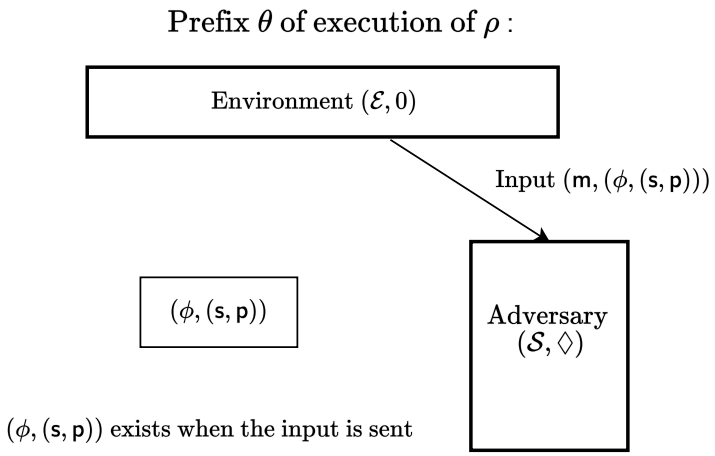


Figure 37: Illustration of a prefix θ that satisfies the hypotheses given in the description of Weakness (W.5).

The goal of the demonstration is to show that for any balanced, PPT environment $\mathcal{E}$, the following holds:

$$\mathsf{EXEC}_{\rho^{\phi\to\pi},\mathcal{D},\mathcal{E}} \approx \mathsf{EXEC}_{\rho,\mathcal{S},\mathcal{E}} \,. \tag{12}$$

By definition, $\mathcal{S}$ simulates multiple, disjoint copies of an adversary $\mathcal{S}_\pi$ for ϕ. Moreover, when $\mathcal{S}$ receives an input (m, M) from the environment:

- By item 1.(a) of [Can20, (Figure 10)], $\mathcal{S}$ will request $(\phi, (\mathsf{s}, \mathsf{p}))$ from the subroutine exposing ITIs of each copy of $\mathcal{S}_\pi$.
- If none of these ITIs respond positively, and additionally, if M is not a main party of ϕ, item 1.(c).ii from [Can20, (Figure 10)] shows that $\mathcal{S}$ will send backdoor m to M.

We consider protocols ρ, π, ϕ, a balanced PPT environment $\mathcal{E}$, sids and pid s, p, and a message m that satisfy the following reasonable hypotheses, illustrated in Figure 37:
- All possible extended sessions of π and ϕ are disjoint. - There is a prefix θ of the execution of ρ with environment $\mathcal{E}$ and adversary $\mathcal{S}$ such that $(\mathcal{E}, 0)$ sends input $(\mathsf{m}, (\phi, \mathsf{s}, \mathsf{p}))$ to $(\mathcal{S}, \Diamond)$. - At the moment $(\mathcal{E}, 0)$ sends this input, $(\phi, \mathsf{s}, \mathsf{p})$ exists in the prefix θ.

Using the description of $\mathcal{S}$ upon the reception of an input and these hypotheses, we see that when $(\mathcal{S}, \Diamond)$ receives the input, it sends backdoor m to $(\phi, \mathsf{s}, \mathsf{p})$, which already exists in the prefix.

We now consider how the same situation is handled in the corresponding execution of $\rho^{\phi\to\pi}$ with the same environment $\mathcal{E}$ and dummy adversary $\mathcal{D}$.

By the definition of $\rho^{\phi\to\pi}$ and because all possible extended sessions of π and ϕ are disjoint, when $\mathcal{E}$ sends input $(\phi, \mathsf{s}, \mathsf{p})$ to $\mathcal{D}$, we know that $(\phi, (\mathsf{s}, \mathsf{p}))$ doesn't exist, as illustrated in Figure 38.

The backdoor m that $(\mathcal{D}, \Diamond)$ will try to send to $(\phi, (\mathsf{s}, \mathsf{p}))$ will thus be rejected by the control function, as it is sent with forced-write unset.

To summarize, we identified a common situation where:

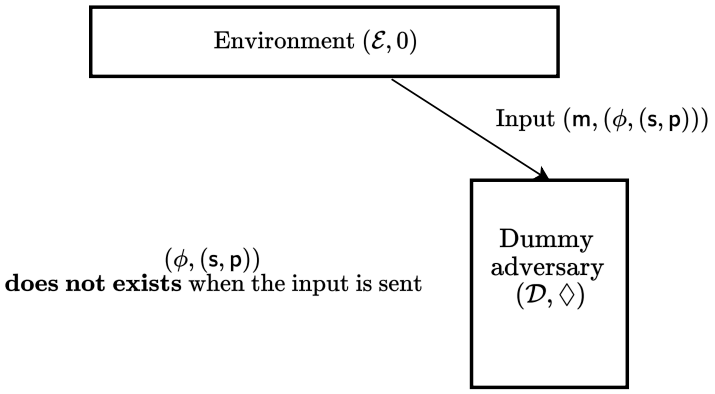


Figure 38: Illustration of an execution of $\rho^{\phi \to \pi}$ with $\mathcal{E}$ and $\mathcal{D}$, corresponding to θ. This figure is used in the description of Weakness (W.5).

- In the execution of ρ with $\mathcal{E}$ and $\mathcal{S}$, $(\phi, (\mathsf{s}, \mathsf{p}))$ receives a backdoor.
- In the execution of $\rho^{\phi \to \pi}$ with $\mathcal{E}$ and $\mathcal{D}$, the control function disallows the sending of the corresponding backdoor.

For a well-chosen ϕ, this contradicts Equation (12).

As a consequence, we see that $\mathcal{S}$ should reject in such a situation. In the proof of the UC Theorem of Updated UC (Appendix C), these situations are taken into account.

D.4 Problems Related to Predicates

In Plain UC, since the revision [Can18], a message sent with flag forced-write unset may designate its recipient using a predicate on ITIs instead of an extended identity [Can20, (Section 3.1.2)].

This predicate is evaluated by the control function to determine the recipient's extended identity. If multiple ITIs in the execution match the predicate, the first one invoked is selected. If no ITI matches, the environment is activated with no message, as though the sender had performed a deactivation.

This feature can be of interest to support cases where the sender has only partial information about the recipient. To our knowledge, both the adversary and protocol ITIs may use this mechanism during protocol execution: the adversary always unsets forced-write, and protocol ITIs are not required to set it [Can20, (Section 4.1)].

However, we show in this section that this mechanism leads to several issues. Specifically:

- The first subsection identifies a problem arising from the UC operator's incompatibility with predicate-based addressing.
- The second subsection presents a pair of protocols π and ϕ such that π UC-emulates ϕ against adversaries that do not use predicates, but fails to UC-emulate ϕ when adversaries are permitted to use them.
- The third subsection assumes a naïve fix to the problem discussed in the first subsection and introduces another counterexample to the UC Theorem, this time involving protocols that use predicates.

To account for these counterexamples, this section departs from our earlier conventions and permits protocol ITIs to unset the flag forced-write for outputs.

D.4.1 (I.7) Non-Adaptation of UC Operator to Predicates.

The definition of the UC operator assumes that the ITIs use extended identities when they want to send a message. This can for example be seen in the item 2 of [Can20, (Section 6.1)]. As a result, the operator cannot be applied to protocols that use predicates.

D.4.2 (L.4) UC-Emulation With Adversary Able to Use Predicates.

To our knowledge, the vast majority of adversaries considered in the proofs of UC-emulation are not allowed to use predicates.

In this section, we show with a counterexample that the UC-emulation for such adversaries does not imply the UC-emulation for adversaries able to use predicates.

The idea of the counterexample is to use predicates that hide the extended identity the adversary will write to.

We define π and ϕ in Figure 39, their subroutines (respectively α and β) are defined on Figure 40.

In a nutshell:

- π is such that if $(\pi, (0,0))$ receives as input a bit b, it activates $(\alpha, (0,0))$ first, then $(\alpha, (0,1))$.
- ϕ is such that if $(\phi, (0,0))$ receives as input a bit b, it activates $(\beta, (0,b))$ first, then $(\beta, (0, 1-b)$.
- For $u \in \{0,1\}$, if the second message received by $(\alpha, (0,u))$ (*resp.* $(\beta, (0,u))$) is a backdoor, it sends output "$\mathsf{0}$" to $(\pi, (0,0))$ (*resp.* $(\phi, (0,0))$).
- For $u \in \{0,1\}$, if the third message received by $(\pi, (0,0))$ (*resp.* $(\phi, (0,0))$) is "$\mathsf{0}$" from $(\alpha, (0,u))$ (*resp.* $(\beta, (0,u))$), then it sends output "u" to an ITI N with code $\notin \{\phi, \psi, \alpha, \beta\}$.

Proposition 18. *π and ϕ are subroutine respecting.*

Proof. The proof is straightforward for both protocols. We briefly discuss π. In the execution of an arbitrary protocol, only $(\pi, (0,0))$ can have subroutine that are $(\alpha, (0,0))$ and $(\alpha, (0,1))$, these subroutines only send outputs to $(\pi, (0,0))$ and accept input or output only from $(\pi, (0,0))$. Moreover, $(\pi, (0,0))$ accepts outputs only from these subroutines. This As a result, all the conditions of the subroutine respecting property are satisfied. □

Proposition 19. *π and ϕ are subroutine exposing.*

Proof. It is direct by construction, the exposing ITI of extended sessions (π, s) (*resp.* (ϕ, s)) being $(\pi, (s, 0))$ (*resp.* $(\phi, (\mathsf{s}, 0))$). □

Proposition 20. *π UC-emulates ϕ.*

Proof. If we consider an adversary $\mathcal{A}$ for the execution of ϕ that always designates the recipient by its extended identity, we can easily create a corresponding adversary $\mathcal{S}$ for the execution of π that ensure UC-emulation.

$\mathcal{S}$ simulates $\mathcal{A}$ and when $\mathcal{A}$ wants to write a backdoor to $(\alpha, (0,u))$ $(u \in \{0,1\})$, $\mathcal{S}$ pass the message to $(\beta, (0,u))$. □

Proposition 21. *π does not UC-emulate ϕ if we allow the adversary to use predicates.*

Proof. We construct an adversary $\mathcal{A}$ and an environment $\widetilde{\mathcal{E}}$.

$\mathcal{A}$ is such that when $(\mathcal{A}, \Diamond)$ receives an input from the environment, it sends a backdoor to an ITI by using the predicate "code is α".

$\widetilde{\mathcal{E}}$ acts as following:

- $(\widetilde{\mathcal{E}}, 0)$ samples a random bit $b \leftarrow\!\!\$\ \{0,1\}$ and sends b to the protocol with identifier $(0,0)$.

	(π, id)	(ϕ, id)
	Local var: integer cpt init to 0	**Local var:** integer cpt init to 0
	integer bit, init to -1	
1 :	**if** id $\neq (0,0) \wedge \neg$IsBackdoor **then**	**if** id $\neq (0,0) \wedge \neg$IsBackdoor **then**
2 :	**reject**	**reject**
3 :	At reception of input or output :	At reception of input or output :
4 :	**if** cpt $= 0$ **then**	**if** cpt $= 0$ **then**
5 :	**if** it is an input "b", b $\in \{0,1\}$ **then**	**if** it is an input "b", b $\in \{0,1\}$ **then**
6 :	cpt $\leftarrow 1$	cpt $\leftarrow 1$
7 :	bit $\leftarrow$ b	
8 :	**send input** "0" to $(\alpha, (0, \mathsf{bit}))$	**send input** "0" to $(\beta, (0,0))$
9 :	**else reject**	**else reject**
10 :	**elseif** cpt $= 1$ **then**	**elseif** cpt $= 1$ **then**
11 :	**if** it is output "0"	**if** it is output "0"
	from $(\alpha, (0, \mathsf{bit}))$ **then**	from $(\beta, (0,0))$ **then**
12 :	cpt $\leftarrow 2$	cpt $\leftarrow 2$
13 :	**send input** "0" to $(\alpha, (0, 1 - \mathsf{bit}))$	**send input** "0" to $(\beta, (0,1))$
14 :	**else reject**	**else reject**
15 :	**elseif** cpt $= 2$ **then**	**elseif** cpt $= 2$ **then**
16 :	**if** it is output "0"	**if** it is output "0"
	from $\alpha, (0,u)), u \in \{0,1\}$ **then**	from $(\beta, (0,u)), u \in \{0,1\}$ **then**
17 :	cpt $\leftarrow 3$	cpt $\leftarrow 3$
18 :	**send output** "u" to N	**send output** "u" to N
19 :	**else reject**	**else reject**
20 :	**elseif** cpt ≥ 3 **then reject**	**elseif** cpt ≥ 3 **then reject**
21 :	At reception of a backdoor :	At reception of a backdoor :
22 :	// subroutine exposing interface:	// subroutine exposing interface:
23 :	**if** p $\neq 0$ **then reject**	**if** p $\neq 0$ **then reject**
24 :	**checkorreject** message can be	**checkorreject** message can be
	parsed as an ITI M	parsed as an ITI M
25 :	**if** $M = (\phi, \mathsf{id})$ **then**	**if** $M = (\phi, \mathsf{id})$ **then**
26 :	**send backdoor** "true" to adversary	**send backdoor** "true" to adversary
27 :	**if** id $= (0,0) \wedge M = (\beta, (0, \mathsf{bit})) \wedge$ cpt ≥ 1 **then**	**if** id $= (0,0) \wedge M = (\beta, (0,0)) \wedge$ cpt ≥ 1 **then**
28 :	**send backdoor** true to adversary	**send backdoor** true to adversary
29 :	**if** id $= (0,0) \wedge M = (\beta, (0, 1 - \mathsf{bit})) \wedge$ cpt ≥ 2 **then**	**if** id $= (0,0) \wedge M = (\beta, (0,1)) \wedge$ cpt ≥ 2 **then**
30 :	**send backdoor** true to adversary	**send backdoor** true to adversary
31 :	**send backdoor** "false" to adversary	**send backdoor** "false" to adversary

Figure 39: Codes of ϕ and π for Limitation (L.4). The relevant differences are highlighted. All messages are sent with reveal-sender-id and forced-write set. N is an ITI with code $\notin \{\phi, \psi, \alpha, \beta\}$.

All messages are sent with
reveal-sender-id and forced-write set
reveal-sender-id and forced-write set

$(\alpha, (\mathsf{s}, \mathsf{p}))$

```
1 : if (s, p) ∉ {(0,0), (0,1)} then reject
2 : At reception of input :
3 : if it is input "0"
        from (π, (0,0)) then
4 :     send output "0" to (π, (0,0))
5 : else reject
6 : At reception of backdoor :
7 :     send output "0" to (π, (0,0))
8 : At reception of output :
9 : reject
```

$(\beta, (\mathsf{s}, \mathsf{p}))$

```
1 : if (s, p) ∉ {(0,0), (0,1)} then halt
2 : At reception of input :
3 : if it is input "0"
        from (φ, (0,0)) then
4 :     send output "0" to (φ, (0,0))
5 : else reject
6 : At reception of backdoor :
7 :     send output "0" to (φ, (0,0))
8 : At reception of output :
9 : reject
```

Figure 40: Codes of α and β for Limitation (L.4).

- At reception of an output, $(\widetilde{\mathcal{E}}, 0)$ sends an input 0 to $(\mathcal{A}, \Diamond)$.
- The next time that $(\widetilde{\mathcal{E}}, 0)$ is activated. It returns 0 if it receives output b, and it returns 1 else.

We note that, in the execution of $(\pi, (0,0))$ with adversary $\mathcal{A}$ and environment $\widetilde{\mathcal{E}}$, $(\widetilde{\mathcal{E}}, 0)$ will always output 0. Indeed, the predicate used by $(\mathcal{A}, \Diamond)$ is both respected by $(\alpha, (0,0))$ and $(\alpha, (0,1))$. As a consequence, the recipient will be the earliest ITI invoked among them, which is $(\alpha, (0, b))$. $\widetilde{\mathcal{E}}$ will thus receive output b and return 0.

Now, we claim that it is impossible to construct an adversary $\mathcal{S}$ such that the execution of ϕ with $\mathcal{S}$ and $\widetilde{\mathcal{E}}$ will always return 0. Indeed, the input b sent by $\widetilde{\mathcal{E}}$ is independent of any information that $\mathcal{S}$ can obtain, even using predicates on the extended identity of ITIs. $\widetilde{\mathcal{E}}$ will thus return 0 with probability 1/2. □

D.4.3 Using Predicates On Protocols.

Issue D.4.1 (page 94) demonstrated that the UC operator must be adapted for protocols using predicates. In this section, we consider a "naive" adaptation of the UC operator and examine some of its concerns.

We will assume in the definition of $\rho^{\phi \to \pi}$, that when the body of a structured protocol of an extended session of $\rho^{\phi \to \pi}$ uses a predicate P, the structured protocol replaces it by a predicate $\widetilde{P}$. The modified predicate is satisfied for structured protocols $((\mu, \sigma_{\pi,phi}), \mathsf{id})$ such that (μ, id) satisfies P.

With this solution, we will construct a counterexample to the UC theorem. We believe this example demonstrates that adding support for predicates to the UC operator must be accompanied by new constraints on the protocols ϕ, π and ρ. These constraints could, for instance, be incorporated into the notion of compliance.

We will create three protocols ϕ, π and ρ such that:

- ϕ and π are subroutine respecting and subroutine exposing.
- ϕ UC-emulates π.

- ρ is (ϕ, π)-compliant.
- The adaptation of UC operator to predicates is such that $\rho^{\phi \to \pi}$ does not UC-emulate π.

The definitions of ϕ and π are given on Figure 41. In a nutshell, when $(\pi, (0,0))$ (*resp.* $(\phi, (0,0))$) receives an input $\mathsf{0}$ from the environment, $(\pi, (0,0))$ (*resp.* $(\phi, (0,0))$) will go back and forth two times with its only subroutine $(\alpha, (0,0))$ (*resp.* $(\beta, (0,0))$) with four distinct messages:

- An input $\mathsf{0}$ from $(\pi, (0,0))$ to $(\alpha, (0,0))$ (*resp.* $(\phi, (0,0))$ to $(\beta, (0,0))$).
- An output $\mathsf{0}$ from $(\alpha, (0,0))$ to $(\pi, (0,0))$ (*resp.* $(\beta, (0,0))$ to $(\pi, (0,0))$).
- An output $\mathsf{1}$ from $(\pi, (0,0))$ to $(\alpha, (0,0))$ (*resp.* $(\phi, (0,0))$ to $(\beta, (0,0))$).
- An output $\mathsf{0}$ from $(\alpha, (0,0))$ to $(\pi, (0,0))$ (*resp.* $(\beta, (0,0))$ to $(\pi, (0,0))$).

Then, $(\pi, (0,0))$ (*resp.* $(\phi, (0,0))$) halts.

Nevertheless, while $(\phi, (0,0))$ sends the output $\mathsf{1}$ to $(\alpha, (0,0))$ by indicating $(\beta, (0,0))$ as recipient, $(\pi, (0,0))$ uses instead a predicate ζ to designate $(\alpha, (0,0))$, where $\zeta(M) = (\mathsf{code}(M) \neq \pi) \wedge ((\mathsf{sid}(M), \mathsf{pid}(M)) = (0,0))$. This will form the basis of the counterexample, as this predicate will only hold for $(\alpha, (0,0))$ in the execution of π with test sid 0, but will apply to other ITIs when executed in $\rho^{\phi \to \pi}$.

Finally, we insist that the third message is an output: this will be necessary to ensure that π is subroutine respecting (Proposition 22).

Proposition 22. *π and ϕ are subroutine respecting.*

Proof. This is direct for ϕ, and for π, we note that the only difficulty is that, in some executions (of an arbitrary protocol), $(\pi, (0,0))$ could send a message to an ITI different of $(\alpha, (0,0))$ as the predicate ζ could be verified by another ITI invoked before. Nevertheless, as this message is an output, the subroutine respecting property is respected. □

Proposition 23. *π and ϕ are subroutine exposing.*

Proof. It is direct by construction, the exposing ITI of session (π, s) (*resp.* (ϕ, s)) being $(\pi, (s, 0))$ (*resp.* $(\phi, (\mathsf{s}, 0))$). □

Proposition 24. *π UC-emulates ϕ.*

Proof. We consider the execution of π with dummy adversary $\mathcal{D}$ and some environment $\mathcal{E}$. The test sid is denoted by $\mathsf{s_{test}}$.

The proof is made in two steps.

Step 1: we can suppose that $\mathcal{D}$ never receives any input or backdoor.

Note that the only communications between protocols ϕ, π and their adversaries are through the requests to the exposing ITI. We create $\mathcal{S}$, adversary of ϕ, as $\mathcal{D}$, with the only difference being that when $\mathcal{D}$ sends or receives a message linked to the subroutine exposing property, $\mathcal{S}$ modifies the message as follows:

- The code of the exposing ITI $(\pi, (\mathsf{s_{test}}, 0))$ is replaced by ϕ, both for messages from the exposing ITI to the adversary and from the adversary to the exposing ITI.
- Requests about the presence of $(\alpha, (0,0))$ in the session, are replaced by requests about the presence of $(\beta, (0,0))$.

We can directly see that $\mathcal{S}$ simulates perfectly the requests made through $\mathcal{D}$ to the exposing ITI $(\pi, (\mathsf{s_{test}}, 0))$ because the subroutine $(\alpha, (0,0))$ is invoked in the execution of ϕ exactly under the same conditions as $(\beta, (0,0))$ in the execution of π.

Moreover, $\mathcal{E}$ does not need to send messages to the dummy adversary in order to know which ITIs are on the extended session $\mathsf{Ext}(\phi, \mathsf{s_{test}})$: the existence of the ITIs depends solely on the inputs $\mathcal{E}$ sent.

We have thus shown that:

$(\pi, (\mathsf{s}, \mathsf{p}))$

```
1 : local variable: boolean alphaSub,
                    initialized to false
                    integer cpt,
                    initialized to 0
2 : if (s, p) ≠ (0, 0) ∧ ¬IsBackdoor then
3 :   reject
4 : At reception of input or output :
5 : if cpt = 0 then
6 :   if it is input "0" then
7 :     alphaSub ← true
8 :     cpt ← 1
9 :     send input "0" to (α, (0, 0))
10 :  else reject
11 : elseif cpt = 1 then
12 :  if it is output "0"
        from (α, (0, 0)) then
13 :    cpt ← 2
14 :    send output "1" with forced-write
15 :     unset using the predicate:
16 :    ζ(M) = code(M) ≠ π ∧
17 :          (sid(M), pid(M)) = (0, 0)
18 :  else reject
19 : elseif cpt ≥ 2 then reject
20 : At reception of a backdoor :
21 :  // subroutine exposing interface:
22 : if p ≠ 0 then reject
23 : checkorreject message can be
                parsed as an ITI M
24 : if M = (π, (s, p)) then
25 :  send backdoor "true" to adversary
26 : if (s, p) = (0, 0) ∧ M = (α, (0, 0)) then
27 :  send backdoor "alphaSub"
                to adversary
28 : send backdoor "false" to adversary
```

$(\phi, (\mathsf{s}, \mathsf{p}))$

```
1 : local variable: boolean betaSub,
                    initialized to false
                    integer cpt,
                    initialized to 0
2 : if (s, p) ≠ (0, 0) ∧ ¬IsBackdoor then
3 :   reject
4 : At reception of input or output :
5 : if cpt = 0 then
6 :   if it is input "0" then
7 :     betaSub ← true
8 :     cpt ← 1
9 :     send input "0" to (β, (0, 0))
10 :  else reject
11 : if cpt = 1 then
12 :  if it is output "0"
        from (β, (0, 0)) then
13 :    cpt ← 2
14 :    send output "1" to (β, (0, 0))
15 :  else reject
16 : elseif cpt ≥ 2 then reject
17 : At reception of a backdoor :
18 :  // subroutine exposing interface:
19 : if p ≠ 0 then reject
20 : checkorreject message can be
                parsed as an ITI M
21 : if M = (π, (s, p)) then
22 :  send backdoor "true" to adversary
23 : if (s, p) = (0, 0) ∧ M = (β, (0, 0)) then
24 :  send backdoor "betaSub"
                to adversary
25 : send backdoor "false" to adversary
```

Figure 41: Codes of π, ϕ for concern of Section D.4.3. α and β are defined in Figure 42. The relevant differences are highlighted. Unless specified otherwise, all messages are sent with reveal-sender-id and forced-write set.

(α, id)	(β, id)
1 : **local variable:** integer cpt, initialized to 0	1 : **local variable:** integer cpt, initialized to 0
2 : **if** id $\neq (0,0)$ **then halt**	2 : **if** id $\neq (0,0)$ **then reject**
3 : At reception of an input :	3 : At reception of an input :
4 : **if** it is input "0" from $(\pi,(0,0))$ **then**	4 : **if** it is input "0" from $(\phi,(0,0))$ **then**
5 : **send output** "0" to $(\pi,(0,0))$	5 : **send output** "0" to $(\phi,(0,0))$
6 : **else reject**	6 : **else reject**
7 : For other activations:	7 : For other activations:
8 : **reject**	8 : **reject**

Figure 42: Codes of subroutines α and β used respectively by π and ϕ for concern of Section D.4.3. The code of π and ϕ is provided in Figure 41. All messages are sent with reveal-sender-id and forced-write set.

(ρ, id)
1 : **if** id $\neq (0,0)$ **then halt**
2 : At first activation :
3 : **send input** "0" to $(\phi,(0,0))$
4 : At second activation :
5 : **if** it is an output "1" **then**
6 : **send output** "1" to N
7 : **else reject**
8 : For other activations :
9 : **reject**

Figure 43: Code of ρ for concern of Section D.4.3. Note that all messages are sent with reveal-sender-id and forced-write set. Moreover, N is an ITI with code $\notin \{\phi, \pi, \alpha, \rho\}$.

- $\mathcal{E}$ cannot exploit a difference of behavior between $\mathcal{D}$ and $\mathcal{S}$ to distinguish the execution of π from ϕ.
- $\mathcal{E}$ does not need to send messages to $\mathcal{D}$, as it already knows the information $\mathcal{D}$ would give.

Step 2: $(\mathcal{E}, 0)$ cannot distinguish the execution of π or ϕ by sending inputs.

Moreover, the only difference of behavior between π and ϕ that could detect the environment is the fact that π uses a predicate on the second input it sends (Line 14). Nevertheless, in an execution of π with any environment and adversary, this predicate will always designate the ITI $(\alpha, (0,0))$. □

The definition of ρ is given in Figure 43.

Proposition 25. *ρ is (π, ϕ)-compliant.*

Proof. It is direct, as ρ never writes to an ITI with code π and that the flag forced-write is always set when sending inputs. □

Proposition 26. *With the proposed adaptation of the UC operator to support predicates, $\rho^{\phi \to \pi}$ does not UC-emulate ρ.*

Proof. Consider an environment $\widetilde{\mathcal{E}}$ that sends any message to the protocol with identifier $(0,0)$. We can observe the following:

- In the execution of ρ, $\widetilde{\mathcal{E}}$ will be activated due to the deactivation of an ITI.
- In the execution of $\rho^{\phi \to \pi}$, $\widetilde{\mathcal{E}}$ will receive the output "1". This is because when $(\pi, (0,0))$ sends output "1" using the predicate $\zeta(M) = (\mathsf{code}(M) \neq \pi) \wedge ((\mathsf{sid}(M), \mathsf{pid}(M)) = (0,0))$, the first invoked ITI that satisfies this condition is $(\rho, (0,0))$, which is invoked before $(\alpha, (0,0))$. As a result, $(\rho, (0,0))$ receives the output "1" and sends output "1" to the environment.

□

D.5 Problems Related to the Notion of T-Bounded Protocols

In this section, we review the definition of a T-bounded protocol and discuss several aspects, including one that introduces a weakness in the proof of the UC Theorem in Plain UC. We note that other works—such as [HUM09, Section 3]—offer valuable discussions of this notion. However, the points we raise here are distinct from those.

D.5.1 Definition of a T-Bounded Protocol.

The notion of T-bounded protocol is defined in [Can20, (Section 3.2)] and discussed in [Can20, (Section 3.3.4)]. We briefly recall it here.

To account for ressource-bounded computation in systems of ITM, each message sent by any ITM is a pair "(i, m)", where m is the actual message and $i \in \mathbb{N}$ a number called its import.

The import can be thought of as a number of "tokens" that an ITI can "consume" in exchange for computation time or transfer to another ITI. In the formal definition provided below, the number of tokens an ITI has at a given moment is called its total budget.

More formally, let's recall the definition of locally T-bounded, T-bounded and PPT bounded protocols, where T is a function $T : \mathbb{N} \to \mathbb{N}$.

Definition 20 (T-bounded [Can20, (Definition 6)])**.** Recall that the import of a message is the value written in the import field of the message. Let $T : \mathbb{N} \to \mathbb{N}$. An ITM μ is locally T-bounded if, at any prefix θ of an execution of a system of ITMs, any ITI M with program μ satisfies the following conditions:

- $\mathsf{Budget}(M, \theta) \geq 0$, where $\mathsf{Budget}(M, \theta) = n_I - n_O$, n_I is the overall imports of the messages written by other machines on the externally writable tapes of M, and n_O is the overall imports of the messages written by M on externally writable tapes of other machines.
- The overall number of computational steps taken by M so far is at most $T(\mathsf{Budget}(M, \theta))$.

T-bounded ITMs are defined inductively: A locally T-bounded ITM that does not make external-writes with forced-write is T-bounded. In addition, a locally T-bounded ITM, all of whose external-writes with forced-write specify a recipient ITM which is T-bounded, is T-bounded as well. ITM μ is PPT if there exists a polynomial P such that μ is P-bounded. A protocol is PPT if it is PPT as an ITM.

D.5.2 Problems with 0-Import Messages.

For any function $T : \mathbb{N} \to \mathbb{N}$ and number u such that $T(u)$ is high enough, we can create a system of ITMs that is T-bounded and whose execution with an initial message of import u contains more than $T(u)$ distinct ITIs.

This gives a counterexample of [Can20, (Proposition 7)], as its proof shows that this number should be bounded by $T(u)$ if T is also super-additive and increasing.

In the next paragraphs, we will present this counterexample, propose a modification of the definition of T-bounded ITI and prove a variant of [Can20, (Proposition 7)] with this new definition.

The Counterexample:

We fix $u \in \mathbb{N}$. We will use two constants $\tau, t_u > 0$ that will be fixed at the end. We consider a protocol $\mu = \mu(u, \tau, t_u)$ that acts as follows. For any identifier id and any activation:

- If its identifier id cannot be interpreted as a number $0 \leq \mathsf{id} < t_u + \tau$, (μ, id) halts.
- If its import is not u, (μ, id) halts.
- If the two first conditions are verified, (μ, id) invokes $(\mu, \mathsf{id}+1)$ with input "0" and import u, then halts.

An execution of $(\mu, 0)$ is shown in Figure 44.

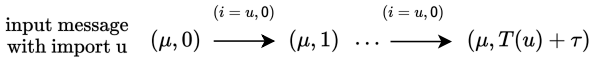


Figure 44: Note that all messages are invocations.

As the reading and writing of t_u, id, τ and u is linear in the size of id and the values $\log_2(t_u), \log_2(u)$, if t_u is high enough, we can find a carefully chosen $\tau > 0$ and $t_u > 0$ such that each ITI with code μ will take less than t_u operations in any execution. As a consequence, μ is T-bounded for any function T with $T(u) = t_u$.

Nevertheless, by construction, the execution of $(\mu, 0)$ contains $T(u) + \tau + 1$ ITIs.

First Try to a Solution:

We could avoid this counterexample by forbidding any ITI with budget u to send messages with forced-write flag set and import u – in particular, it would not be able to invoke an ITI using all its budget.

Nevertheless, more subtle counterexamples could still be created, where (μ, id) invokes (μ, id) with import $u - 1$, and only gives later the last token it kept to (μ, id), as shown in Figure 45.

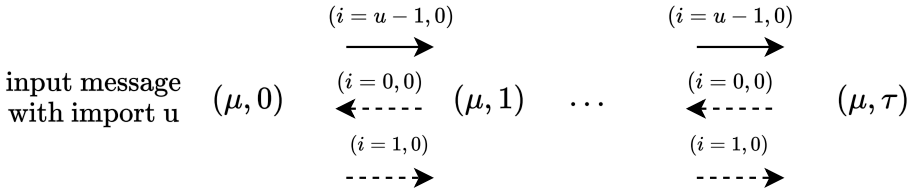


Figure 45: The three messages are sent from up to bottom. The dotted arrows represent messages with forced-write flag unset, solid arrows are invocations.

A Solution:

We propose some additions to the T-bounded notion that aims to forbid the two counterexamples we described.

Definition 21 (T-bounded, revisited)**.** An ITM is locally T-bounded in our new sens if it is locally T-bounded as in Definition 20and if, in addition, in any prefix of execution of a system of ITMs, any ITI M with code μ satisfies the following conditions:

- At any moment, if its budget is strictly positive, M does not send a message with import equal to its budget.
- At any moment, if its budget is 0, M does not send a message with flag forced-write set – in particular, it cannot invoke ITIs.

T-bounded ITMs are defined inductively: A locally T-bounded ITM that does not make external-writes with forced-write is T-bounded. In addition, a locally T-bounded ITM, all of whose external-writes with forced-write specify a recipient ITM which is T-bounded, is T-bounded as well. ITM μ is PPT if there exists a polynomial P such that μ is P-bounded. A protocol is PPT if it is PPT as an ITM.

We now prove a variant of the claims of [Can20, (Proposition 7)] related to the execution time of systems of ITIs and made in its proof.

Recall that a function $T : \mathbb{N} \to \mathbb{N}$ is super additive if, for any $n, n' \in \mathbb{N}$, $T(n + n') \geq T(n) + T(n')$.

Proposition 27 ([Can20, (Proposition 7)], revisited)**.** *Let $T : \mathbb{N} \to \mathbb{N}$ a super additive, increasing function and a system of ITMs (I, C) such that the initial ITM is T-bounded in the sense of Definition 21. We consider any prefix of execution θ of (I, C) where the import of the initial input is $n \in \mathbb{N}$. Then:*

1. *The time of execution of θ, defined as the number of operation made by each ITI in θ, is bounded by $T(n)(1 + T(0))$.*
2. *The total number of ITIs of θ is bounded by $T(n) + 1$.*

Proof. Note that at any moment of the execution, the sum of budgets of each of the active ITIs is equal to n.

Let $\mathcal{M}^t_{\geq 1}$ the set of ITIs that exists in θ, with a budget ≥ 1 and $\mathcal{M}^t_{=0}$ the set of ITIs that exists in θ, with budget 0 (note that budget is always ≥ 0).

For each $N \in \mathcal{M}^t_{\geq 1} \sqcup \mathcal{M}^t_{=0}$, we note $\mathsf{Budget}(N, \theta)$ its budget.

We have:

$$\sum_{N \in \mathcal{M}^t_{\geq 1}} \mathsf{Budget}(N, \theta) = n\,.$$

Moreover, because all theses ITIs are locally T-bounded and that T is super-additive and increasing, we deduce that, in θ, the added execution time of all the ITMs of $\mathcal{M}^t_{\geq 1}$ is bounded by $T(n)$. Indeed:

$$\begin{aligned}\sum_{N \in \mathcal{M}^t_{\geq 1}} T\Big(\mathsf{ExecTime}(N, \theta)\Big) &\leq \sum_{N \in \mathcal{M}^t_{\geq 1}} T\Big(\mathsf{Budget}(N, \theta)\Big) \\ &\leq T\left(\sum_{N \in \mathcal{M}^t_{\geq 1}} \mathsf{Budget}(N, \theta)\right) = T(n)\,.\end{aligned}$$

We now study the addition of the execution time of all ITMs of $\mathcal{M}^t_{=0}$.

First note there are only two possibilities for an ITM N to be in $\mathcal{M}^t_{=0}$:

- $n = 0$ and it is the initial ITI. The results of the proposition are thus direct, as there is only the initial ITI in the system.
- $n > 0$, N has been invoked with import 0 and never received any messages with an import different of 0.

We focus on the second case. Note that an ITI of $\mathcal{M}^t_{=0}$ can only have been invoked by an ITI of $\mathcal{M}^t_{\geq 0}$. As a result, we can roughly upper-bound the number of ITIs of $\mathcal{M}^t_{=0}$ by $T(n)$, as we have shown that the added execution time of all ITIs of $\mathcal{M}^t_{\geq 1}$ is bounded by

$T(n)$. We obtain that the addition of the execution time of the ITIs of $\mathcal{M}^t_{=0}$ is bounded by $T(0)T(n)$.

The sum of time of execution of all the ITIs of the system, at time t, is thus bounded by $T(n)(1+T(0))$. This proves the first point of the proposition.

We now prove the second point. At any time t, except the initial ITI, each ITI of the system has been invoked by an ITI of $\mathcal{M}^t_{\geq 1}$. As we have previously shown that the total time of execution of each ITI of $\mathcal{M}^t_{\geq 1}$ is bounded than $T(n)$, this implies that the total number of ITI of the system is bounded by $T(n)+1$. □

D.5.3 (W.6) The Case of Protocol Execution.

The definition of T-bounded ITI is based on the execution of systems of ITMs, where no impersonation occurs. However, UC-emulation takes place in the context of a protocol execution, which is an extended system where the environment can perform impersonation.

As a result, it is possible to create a protocol π that is T-bounded but for which the execution time can become infinite under certain environment and adversaries.

For example, lets ν an ITM that halts upon its first activation. We construct π such that, for any identity id, (π, id) rejects any message that does not originate from the ITI $(\nu, (0,0))$. However, if the message does come from $(\nu, (0,0))$, π enters an infinite loop.

π is T-bounded because, in any execution of a system of ITMs, it never receives any message from $(\nu, (0,0))$. However, during the protocol execution of π, an environment can send an input to π by impersonating $(\nu, (0,0))$, triggering the infinite loop.

The solution of this problem is to consider the execution of protocols instead of the execution of systems of ITMs. Nevertheless, this improvement is not enough to deal with structured-protocols, as shown in the next section.

D.5.4 (W.7) The Case of Structured Protocols.

In the previous section, we established that the definition of T-bounded protocols should be considered in the context of protocol executions rather than systems of ITMs. From now on, we assume this adjustment is in place.

In this section, we demonstrate that this improvement is still insufficient to handle structured protocols, which are used, for example, by the replacement protocol. Specifically, we cannot conclude that a structured protocol is T-bounded simply because both its body and shell are. This can affect several constructions in Plain UC that rely on structured protocols, such as the universal composition operator.

For instance, consider μ a locally T-bounded ITM and N, an ITI that halts after its first activation. Now, let's construct $\tilde{\mu}$, which is identical to μ, except that if, for any identifier id, $(\tilde{\mu}, \mathsf{id})$ receives an output from N, $(\tilde{\mu}, \mathsf{id})$ enters an infinite loop and never halts.

We claim that $\tilde{\mu}$ is locally T-bounded because for any execution prefix of any protocol, no ITI with code $\tilde{\mu}$ never receives an output from N. Indeed, N never sends such an output, so this could only occur if the environment impersonates N. However, the rules of execution rules of a protocol prohibit an environment from sending outputs to protocols.

Nevertheless, we can easily create a structured protocol with body code $\tilde{\mu}$, that never halts upon receiving any output. It is sufficient for the shell to replace the sender of each output with N.

This counterexample is possible because we cannot infer, from an execution prefix involving a structured protocol with body M and shell S, a corresponding execution prefix that involves both M and S.

This issue leads to several weaknesses in the proof of the UC theorem of Plain UC, as the UC operator relies on structured protocols, and various ITMs constructed in its proof simulates multiples ITIs.

D.5.5 The Variable Value of a Token.

Here, we present a minor remark about the T-bounded notion, with no impact on the correctness of Plain UC.

We will speak of tokens of an ITI M for its total budget, and we will say that an ITI receives a token if it receives a message with import 1.

When a token is given to an ITI M, the additional time it gives to M isn't the same depending on how many tokens M already has. Indeed, if we note i the number of tokens that M already has, M was limited to a total number of computational steps $T(i)$, and will be now limited to a total number of computational steps $T(i+1)$, which is a gain of $T(i+1) - T(i)$ computational steps. If, for example, $T(n) = n^2$, the additional time allowed to M for a new token is thus equal to $T(i+1) - T(i) = 2i + 1$.

We believe it would be more natural for each additional token to provide the same additional number of computational steps. However, modifying the definition to ensure this may not be straightforward.

D.5.6 The Rarity of Tokens.

Here, we present a minor remark about the T-bounded notion, with no impact on the correctness of Plain UC.

We will use our variant of T-bounded definition (Definition 21). More precisely, we will consider that a T-bounded ITM cannot use all its budget to invoke another ITM, as this behavior is used to create a counterexample in Section D.5.2.

Let's $u \in \mathbb{N}$ We will call "u-chain", in the execution of a protocol, a situation where:

- An ITM I_0 is invoked by an input or output with import u.
- I_0 then invokes another ITI I_1 with an input or output.
- I_1 then invokes another ITI I_2 with an input or output.
- This continues until some ITI I_ℓ performs a different action (*e.g.* deactivates, sends a message to the adversary, communicates with an already invoked ITI, etc.).

We define the length of a u-chain as the number ℓ. We call "member of the chain" of the ITM $I_0, \cdots, I_{\ell-1}$. We call chain a u-chain for some $u \in \mathbb{N}$.

Let $u \in \mathbb{N}$ and a protocol μ that is T-bounded (Definition 21) for some function $T : \mathbb{N} \to \mathbb{N}$. We claim that for any execution of μ, any u-chain has a size bounded by u. Indeed, as each member of the chain invokes another ITI, the import it gives is strictly less than the import it has been invoked with.

We believe it would be more natural for the size of these chains to be bounded by $T(u)$ instead of u, although changing the definition to ensure this might be challenging.

D.6 (W.8): A Weakness Related to the Notion of Balanced Environment

In Updated UC, we do not adopt an analogue of the balanced environment notion. In Plain UC, this condition ensures that environments used to test UC-emulation provide sufficient computational resources to the adversary, thereby preventing starvation. More precisely, the notion aims to rule out scenarios where "the computational resources of the protocol, hence the protocol's computational and communication complexity, are arbitrarily large relative to the computational resources available to the adversary" [Can20, (Section 4.2)].

While this property is functionally useful, we argue that it introduces a weakness in the proof of the UC Theorem in Plain UC. For this reason, it is omitted in Updated UC. The nature of this weakness is explained in the next subsection.

To partially offset the absence of this condition, the definition of a PPT adversary in Updated UC guarantees that the adversary is granted at least $n_M \cdot P(k)$ computational

steps, where P is a polynomial, n_M is the number of input or backdoor messages received, and k is the security parameter.

Description of the Weakness. This subsection outlines a technical flaw in the proof of the UC Theorem in Plain UC, intended for readers familiar with its formalism. We use the notion of parametrized systems and time complexity from [Can20, (Section 3.2)]. All notations below refer to the Plain UC framework.

In Plain UC, UC-emulation is defined only with respect to a restricted class of environments known as *balanced environments* [Can20, (Section 4.2)]. A key consequence of this definition is that the first input message sent by the environment with nonzero import must be addressed to the adversary.

Let us consider an execution of ρ with adversary $\mathcal{S}$, a balanced environment $\mathcal{E}$, and input (k, z), where $k > 0$. We will demonstrate with an example that the proof does not guarantee that, for a given session (π, s), the corresponding copy of $\mathcal{S}_\pi$ receives any message with nonzero import before one of the main parties of the session (ϕ, s) is activated.

As a consequence, the behavior of this copy of $\mathcal{S}_\pi$ cannot be meaningfully related to its behavior in executions used to define UC-emulation of ϕ by π. Indeed, in an execution of ϕ with adversary $\mathcal{S}_\pi$ and a balanced environment, $(\mathcal{S}_\pi, (0, \Diamond))$ never receives its first message after $(\phi, (0, 0))$ has already received an input with of import at least k.

Therefore, even if $\mathcal{S}_\pi$ appropriately emulates $\mathcal{D}$ in such settings, this does not establish the correctness of the simulation within the UC Theorem proof. This gap constitutes a structural weakness.

We demonstrate this lack of garantee with the exemple illustrated in Figure 46. The execution proceeds as follows:

- $(\mathcal{E}, 0)$ sends an initial input $((i =)2k, (m_1, M, 0))$ to $(\mathcal{S}, 0)$, where M is an ITI distinct from ρ, ϕ, π, and $\mathcal{E}$, and m_1 is some message. The notation "$(i =)$" indicates the import value.
- By definition, $\mathcal{S}$ initially behaves like the dummy adversary $\mathcal{D}$ and attempts to send a backdoor message $((i =)0, m_1)$ to M. Since M does not exist in the execution, $(\mathcal{E}, 0)$ is activated with no message.
- The environment $(\mathcal{E}, 0)$ then sends an input $((i =)k, m_2)$ to $(\rho, (0, 0))$.
- The ITI $(\rho, (0, 0))$ processes this input and sends an input $((i =)k, m_3)$ to $(\phi, (0, 0))$.

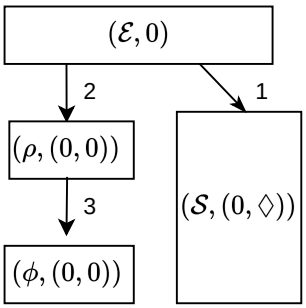


Figure 46: Illustration of Weakness (W.8). Numbers indicate message creation order.

In this example, $\mathcal{E}$ is balanced the initial import to the adversary is $2k$, and the only subsequent import is $k < 2k$. However, $(\phi, (0, 0))$ receives a message with import k before the copy of $\mathcal{S}_\pi$ associated with sid 0 is created.

E Definition of Our Universal Composition Operator

We fix three distinct non-empty binary words: newOperation, simulate, and apply, which will be used throughout the construction.

Our goal is to define a new universal composition (UC) operator that satisfies Definition 13 and resolves the issues outlined in Section 5.2. In addition, we aim for the execution graph of $\rho^{\phi\to\pi}$ to closely match that of the (ϕ,∞,π)-executions of ρ. For instance, we seek a bijective correspondence between ITIs in these two types of executions.

To this end, Sections E.1–E.3 define the construction. Section E.4 states and proves Proposition 28, asserting that the construction indeed defines a UC operator in the sense of Definition 13.

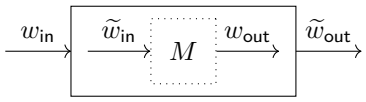
E.1 Main Ideas of the Construction and Definition of Our UC Operator

We define a UC operator on ITMs, written $\rho^{\phi\to\pi}$ for protocols ρ,ϕ,π. The definition naturally extends to parameterized systems $(\boldsymbol{\rho},\boldsymbol{\phi},\boldsymbol{\pi})$ by setting $\boldsymbol{\rho}^{\boldsymbol{\phi}\to\boldsymbol{\pi}} = \left(\rho_k^{\phi_k\to\pi_k}\right)_k$.

We assume that ϕ and π are separated and disjoint, and that $\rho \notin \mathsf{Pan}(\phi)\cup\mathsf{Pan}(\pi)$.

In each extended session $\mathsf{Ext}(\rho^{\phi\to\pi},\mathsf{s}_{\mathsf{test}})$, there exists an ITI $\mathsf{Coord}\,(\mathsf{s}_{\mathsf{test}}) = (\mathsf{repl}\,[\phi\to\pi\,:\,\rho]\,,(\mathsf{s}_{\mathsf{test}},0))$, referred to as the coordinator. The coordinator serves as a middleman, observing and modifying messages exchanged between ITIs of the extended session.

To ensure the ITIs of $\mathsf{Ext}(\rho^{\phi\to\pi},\mathsf{s}_{\mathsf{test}})$ – except for the coordinator itself – route their communication through the coordinator, they are implemented as structured protocols. These simulate a base ITI M – called the body – and may modify both incoming and outgoing external-write operations.



The structured protocols we use are of the form $\mu^{\rightleftarrows\mathsf{Coord}:\rho}$, where μ is the simulated code, $\mathsf{Coord} = \mathsf{repl}\,[\phi\to\pi\,:\,\rho]$ is the coordinator, and ρ,ϕ,π are the protocol to which we want to apply the UC operator.

The two upcoming subsections formally define these structured protocols and the coordinator. These two notions are sufficient to define $\rho^{\phi\to\pi}$.

Proposition 28 later defines a set S for which our operator qualifies as an S-UC Operator, as per Definition 13.

Definition 22 (Universal composition operator)**.** For ρ,ϕ,π such that ϕ,π are separated, disjoint and $\rho\notin\mathsf{Pan}(\phi)\cup\mathsf{Pan}(\pi)$, we define the protocol $\rho^{\phi\to\pi}$ as $\rho^{\rightleftarrows\mathsf{repl}[\phi\to\pi\,:\,\rho]:\rho}$.

For $\boldsymbol{\rho},\boldsymbol{\pi},\boldsymbol{\phi}$ parametrized systems such that $\boldsymbol{\phi},\boldsymbol{\pi}$ are separated, disjoint and $\forall k\in\mathbb{N},\rho_k\notin\mathsf{Pan}(\phi_k)\cup\mathsf{Pan}(\pi_k)$, we define the parametrized system $\boldsymbol{\rho}^{\boldsymbol{\phi}\to\boldsymbol{\pi}}$ as $\left(\rho_k^{\phi_k\to\pi_k}\right)_k$.

E.2 Definition of the Structured Protocol $\mu^{\rightleftarrows\mathsf{Coord}:\rho}$

We define the behavior of $(\mu^{\rightleftarrows\mathsf{Coord}:\rho},\mathsf{id})$, its algorithm being given in Figure 47

We assume that id is structured as a couple (s,p). Else, $(\mu^{\rightleftarrows\mathsf{Coord}:\rho},\mathsf{id})$ halts at first activation.

In a nutshell, $(\mu^{\rightleftarrows\mathsf{Coord}:\rho},(\mathsf{s},\mathsf{p}))$ first determines a sid $\mathsf{s}_{\mathsf{coord}}$ for the coordinator, simulates the ITI $(\mu,(\mathsf{s},\mathsf{p}))$ and performs the following:

- It forwards the backdoors it receives to its body $(\mu,(\mathsf{s},\mathsf{p}))$.
- If $\mu=\rho$, it sends the inputs it receives to its body $(\mu,(\mathsf{s},\mathsf{p}))$. If $\mu\neq\rho$ it rejects.
- It notifies $\mathsf{Coord}\,(\mathsf{s}_{\mathsf{coord}}) = (\mathsf{Coord},(\mathsf{s}_{\mathsf{coord}},0))$ of any external-write the simulation of $(\mu,(\mathsf{s},\mathsf{p}))$ intends to make. Nevertheless, it does not notify of deactivations or rejects.
- If requested by $(\mathsf{Coord},(\mathsf{s}_{\mathsf{coord}},0))$, it simulates the application of any external-write w to the simulation of $(\mu,(\mathsf{s},\mathsf{p}))$, provided the recipient of w is indeed $(\mu,(\mathsf{s},\mathsf{p}))$.

$\left(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p})\right)$

Local variable: a sid $\mathsf{s}_{\mathsf{coord}}$, initialized to $\perp$

1 : At reception of an external-write w that is an input from an ITI M :
2 : **if** $\mu \neq \rho$ **then reject**
3 : **if** $\mathsf{s}_{\mathsf{coord}} = \perp$ **then** $\mathsf{s}_{\mathsf{coord}} \leftarrow \mathsf{s}$
4 : $\widetilde{w} \leftarrow$ the external-write w with recipient replaced by $(\mu, (\mathsf{s}, \mathsf{p}))$
5 : $\mathsf{SimulateOperation}(\widetilde{w})$
6 : At reception of an output "m" from an ITI $(\mathsf{Coord}, (\tilde{\mathsf{s}}, 0))$:
7 : **if** $\mathsf{s}_{\mathsf{coord}} = \perp$ **then** $\mathsf{s}_{\mathsf{coord}} \leftarrow \tilde{\mathsf{s}}$
8 : **if** $\mathsf{s}_{\mathsf{coord}} \neq \perp \wedge \mathsf{s}_{\mathsf{coord}} \neq \tilde{\mathsf{s}}$ **then reject**
9 : **if** $\mathsf{m} = (\mathsf{simulate}, w)$ for an external-write w with recipient $(\mu, (\mathsf{s}, \mathsf{p}))$ **then**
10 : $\quad$ $\mathsf{SimulateOperation}(w)$
11 : **if** $\mathsf{m} = (\mathsf{apply}, w)$ for an external-write w with sender $\left(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p})\right)$ **then**
12 : $\quad$ **apply** w
13 : **reject**
14 : At reception of an external-write w that is a backdoor from an ITI M :
15 : **if** $\mathsf{s}_{\mathsf{coord}} = \perp$ **then reject**
16 : $\widetilde{w} \leftarrow$ the external-write w with recipient replaced by $(\mu, (\mathsf{s}, \mathsf{p}))$
17 : $\quad$ $\mathsf{SimulateOperation}(\widetilde{w})$
18 : In any other case : **reject**

$\mathsf{SimulateOperation}(w)$

1 : **if** the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$ has not been initialized **then**
2 : $\quad$ **initialize** the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$
3 : **simulate** the application of w to the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$
4 : Then simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$ continues until one of the following events occurs :
5 : **if** the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$ rejects **then reject**
6 : **if** the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$ deactivates (but doesn't reject) **then deactivate**
7 : **if** the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$ wants to apply an external-write w **then**
8 : $\quad$ **send input** "$(\mathsf{newOperation}, w)$" to $(\mathsf{Coord}, (\mathsf{s}_{\mathsf{coord}}, 0))$

Figure 47: Code of the ITM $\mu^{\rightleftarrows \mathsf{Coord}:\rho}$.

- If requested by $(\mathsf{Coord}, (\mathsf{s}_{\mathsf{coord}}, 0))$, it applies any external-write w, provided the sender of w is indeed $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$.

The precise behavior of $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ is described in next paragraphs and its code is given in Figure 47. Note that for an ITI M, we will denote $M^{\rightleftarrows \mathsf{Coord}:\rho} = (\mathsf{code}(M)^{\rightleftarrows \mathsf{Coord}:\rho}, \mathsf{id}(M))$.

Determination of the Sid $\mathsf{s}_{\mathsf{coord}}$ for Coord. While $\mathsf{s}_{\mathsf{coord}}$ is unset, the ITI $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ rejects all messages, except in these two situations:

choicesid.1 $\mu = \rho$ and $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ receives an input. $\mathsf{s}_{\mathsf{coord}}$ is then set to s. Next, the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$ is initialized, $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ replaces the recipient of the input with $(\mu, (\mathsf{s}, \mathsf{p}))$, and simulates its sending to the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$.

choicesid.2 $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ receives an output from $(\mathsf{Coord}, (\widetilde{\mathsf{s}}, 0))$, for some sid $\widetilde{\mathsf{s}} \in \{0,1\}^*$. $\mathsf{s}_{\mathsf{coord}}$ is then set to $\widetilde{\mathsf{s}}$. Next, the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$ is initialized, and $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ proceeds as described below in the "Output reception" paragraph.

For next paragraphs, we suppose that the sid $\mathsf{s}_{\mathsf{coord}}$ is set.

Input Reception. When $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ receives an input and $\mu = \rho$, it replaces the recipient by $(\mu, (\mathsf{s}, \mathsf{p}))$ and simulates the sending of this input to the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$. If $\mu \neq \rho$, it rejects.

Backdoor Reception. When $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ receives a backdoor, it replaces the recipient by $(\mu, (\mathsf{s}, \mathsf{p}))$ and simulates the sending of this backdoor to the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$.

Output Reception. $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ rejects any output "m" that does not originate from $(\mathsf{Coord}, (\mathsf{s}_{\mathsf{coord}}, 0))$. When the sender is the ITI $(\mathsf{Coord}, (\mathsf{s}_{\mathsf{coord}}, 0))$, $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ behaves as follows:

- If m is a pair $(\mathsf{simulate}, w)$ where w is an external-write with recipient $(\mu, (\mathsf{s}, \mathsf{p}))$, then $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ simulates the application of w to the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$.
- If m is a pair (apply, w) where w is an external-write with sender $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$, then $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ applies w.
- If neither of these conditions are met, $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ rejects.

Action when the Simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$ Rejects. If the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$ rejects, $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ rejects too.

Action when the Simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$ Deactivates But Does Not Reject. If the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$ deactivates without rejecting, the ITI $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ deactivates too.

Action when the Simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$ Intends to Make an External-Write. If the simulation of $(\mu, (\mathsf{s}, \mathsf{p}))$ intends to make an external-write w, $(\mu^{\rightleftarrows \mathsf{Coord}:\rho}, (\mathsf{s}, \mathsf{p}))$ sends input "$(\mathsf{newOperation}, w)$" to $(\mathsf{Coord}, (\mathsf{s}_{\mathsf{coord}}, 0))$.

E.3 Definition of the Coordinator $\mathsf{Coord} = \mathsf{repl}\,[\phi \to \pi \,:\, \rho]$

We describe $\mathsf{Coord} = \mathsf{repl}\,[\phi \to \pi \,:\, \rho]$, its algorithm being provided in Figures 48 and 49.

Note that $\mathsf{repl}\,[\phi \to \pi \,:\, \rho]$ uses an array $\mathsf{ActualPosition}$. This array stores information about each ITI of the execution by associating each ITI with an element of $\{\mathsf{out}, \bot\} \cup \{0,1\}^*$. Specifically, for an ITI M $\mathsf{ActualPosition}(M)$ is defined as:

- $\bot$ if M does not exist in the current execution prefix. This value can change if M is invoked.

- out if M exists but does not belong to an extended session $\mathsf{Ext}(\pi, \mathsf{s})$ which replaced an extended session $\mathsf{Ext}(\phi, \mathsf{s})$. This value can change because, if M sleeps, it may later belong to such an extended session.
- A sid $\mathsf{s} \in \{0,1\}^*$ if M belongs to an extended session $\mathsf{Ext}(\pi, \mathsf{s})$ that replaced an extended session $\mathsf{Ext}(\phi, \mathsf{s})$. Once set, this value cannot change as an ITI cannot "leave" an extended session.

We will use the notation $\mathsf{Coord}(\mathsf{s}) = (\mathsf{Coord}, (\mathsf{s}, 0))$.

We focus on the behavior of Coord during the execution of $\rho^{\phi \to \pi} = \rho^{\rightleftarrows \mathsf{Coord}:\rho}$ with an adversary $\mathcal{A}$ and environment $\mathcal{E}$, assuming they are fixed. Let $\mathsf{s}_{\mathsf{test}}$ the test sid of this execution. In this context, $\mathsf{Coord}(\mathsf{s}_{\mathsf{test}})$ acts as a middleman, capable of making the same code replacements as in the (ϕ, ∞, π)-execution of ρ because:

- Any ITI U invoked by the environment $(\mathcal{E}, 0)$ will have code $\rho^{\rightleftarrows \mathsf{Coord}:\rho}$ and sid $\mathsf{s}_{\mathsf{test}}$, thus choosing to communicate with $\mathsf{Coord}(\mathsf{s}_{\mathsf{test}})$, as explained in Item choicesid.1 of paragraph "Determination of the sid $\mathsf{s}_{\mathsf{coord}}$ of Coord", page 108.
- Any ITI not invoked by $(\mathcal{E}, 0)$ will be invoked by $\mathsf{Coord}(\mathsf{s}_{\mathsf{test}})$ and will have the form $N^{\rightleftarrows \mathsf{Coord}:\rho}$. This ITI will thus choose to communicate with $\mathsf{Coord}(\mathsf{s}_{\mathsf{test}})$, as explained in Item choicesid.2 of paragraph "Determination of the sid $\mathsf{s}_{\mathsf{coord}}$ of Coord", page 108.

As a consequence, $\mathsf{Coord}(\mathsf{s}_{\mathsf{test}})$ will always know which ITIs are in the extended session

$(\mathsf{repl}[\phi \to \pi : \rho], (\mathsf{s}, \mathsf{p}))$

Local variable: ActualPosition, initialized to out for all ITIs $M \in \mathsf{MainParty}(\rho^{\rightleftarrows \mathsf{repl}[\phi \to \pi : \rho]:\rho}, \mathsf{s})$ and to $\perp$ for all other ITIs

1 : **if** $\mathsf{p} \neq 0$ **then halt**
2 : At reception of a backdoor or output message : **reject**
3 : At reception of an input m from an ITI U :
4 : **checkorreject** there is an ITI M with $U = M^{\rightleftarrows \mathsf{repl}[\phi \to \pi : \rho]:\rho}$
5 : **checkorreject** $\mathsf{m} = (\mathsf{newOperation}, w)$ where w is an external-write
6 : with sender M and recipient some ITI N
7 : we note $V = N^{\rightleftarrows \mathsf{repl}[\phi \to \pi : \rho]:\rho}$, $\mathsf{pos}_U = \mathsf{ActualPosition}[U]$ and $\mathsf{pos}_V = \mathsf{ActualPosition}[V]$
8 : **if** $\mathsf{pos}_U = \perp \wedge \mathsf{code}(M) \neq \rho$ **then reject**
9 : // **Message to adversary:**
10 : **if** w is a backdoor **then**
11 : $\widetilde{w} \leftarrow w$ where sender replaced by U
12 : **send output** "$(\mathsf{apply}, \widetilde{w})$" to U
// **Message relayed to environment:**
13 : **if** w is an output $\wedge\, \mathsf{code}(M) = \rho \wedge \mathsf{sid}(M) = \mathsf{s} \wedge \mathsf{pos}_V = \perp$ **then**
14 : // This is here that the replacement compatible condition (Definition 23)
15 : // is needed. See paragraph "Outputs to Environment", page 111
16 : $\widetilde{w} \leftarrow w$ where sender replaced by U
17 : **send output** "$(\mathsf{apply}, \widetilde{w})$" to U
// End of the algorithm is given in Figure 49

Figure 48: First part of code of $\mathsf{repl}[\phi \to \pi : \rho]$.

$(\mathsf{repl}\,[\phi \to \pi\ :\ \rho]\,, (\mathsf{s}, \mathsf{p}))$

// Begin of the algorithm is given in Figure 48

// **Message with recipient code replacement $\phi \to \pi$:**

if (w is an input or output) $\wedge\ \mathsf{code}(N) = \phi \wedge \mathsf{pos}_U = \mathsf{out}$ **then**

  $\widetilde{N} \leftarrow N$ with code replaced by π

  $\mathsf{ActualPosition}\left[\widetilde{N}^{\rightleftarrows \mathsf{repl}[\phi \to \pi\ :\ \rho]:\rho}\right] \leftarrow \mathsf{sid}(N)$

  $\widetilde{w} \leftarrow w$ with recipient replaced by $\widetilde{N}$

  send output "$(\mathsf{simulate}, \widetilde{w})$" to $\widetilde{N}^{\rightleftarrows \mathsf{repl}[\phi \to \pi\ :\ \rho]:\rho}$

// **Message with sender code replacement $\pi \to \phi$:**

if w is an output $\wedge\ \mathsf{code}(M) = \pi \wedge \mathsf{pos}_U \notin \{\bot, \mathsf{out}, \mathsf{pos}_V\}$ **then**

  $\widetilde{M} \leftarrow M$ with code replaced by ϕ

  $\widetilde{w} \leftarrow w$ with sender replaced by $\widetilde{M}$

  if $\mathsf{pos}_V = \bot$ **then** $\mathsf{ActualPosition}\,[V] \leftarrow \mathsf{out}$

  send output "$(\mathsf{simulate}, \widetilde{w})$" to V

// **Message from two protocols ITI with no replacement:**

// Here, we know that no external-write, deactivation or

// reject has been asked before

if $\mathsf{pos}_V = \bot$ **then** $\mathsf{ActualPosition}\,[V] \leftarrow \mathsf{ActualPosition}\,[U]$

send output "$(\mathsf{simulate}, w)$" to V

Figure 49: Second part of code of $\mathsf{repl}\,[\phi \to \pi\ :\ \rho]$.

$\mathsf{Ext}\left(\rho^{\phi\to\pi}, \mathsf{s}_{\mathsf{test}}\right)$, except for the main-parties of $(\rho^{\phi\to\pi}, \mathsf{s}_{\mathsf{test}})$ that are sleeping. This allows $\mathsf{Coord}\left(\mathsf{s}_{\mathsf{test}}\right)$ to make the replacements of code $\phi \to \pi$, $\pi \to \phi$ that correspond to those in the (π, ∞, ϕ)-execution of ρ. However, we must impose a condition on ρ, ϕ and π to prevent a collision issue, needed in "Messages to Environment" paragraph.

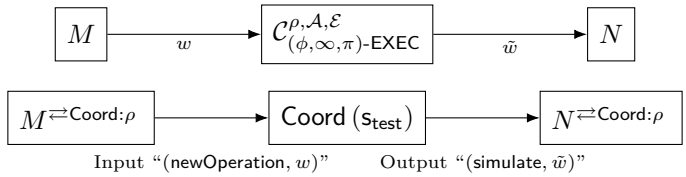
Definition 23 (Replacement compatible)**.** A triplet of protocols (ρ, ϕ, π) is said to be replacement compatible if there is no $(\rho, \phi, \infty, \pi)$-prefix of execution θ, sids s, s' and ITI M such that one of these conditions happens:

- $M^{\rightleftarrows \mathsf{Coord}:\rho} \in \mathsf{Ext}(\rho, \mathsf{s}, \theta)$ (where $\mathsf{Coord} = \mathsf{repl}\left[\phi \to \pi \,:\, \rho\right]$).
- $\mathsf{Coord}\left(\mathsf{s}\right) = \left(\mathsf{repl}\left[\phi \to \pi \,:\, \rho\right], (\mathsf{s}, 0)\right) \in \mathsf{Ext}(\rho, \mathsf{s}, \theta)$.
- A main party of (ρ, s) sends an output to an ITI with code Coord or $M^{\rightleftarrows \mathsf{Coord}:\rho}$ (where $\mathsf{Coord} = \mathsf{repl}\left[\phi \to \pi \,:\, \rho\right]$).

A triplet of parametrized systems $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ is said to be replacement compatible if (ρ_k, ϕ_k, π_k) is replacement compatible for any $k \in \mathbb{N}$.

Next paragraphs go into detail on how $\mathsf{Coord}\left(\mathsf{s}_{\mathsf{test}}\right)$ behaves in each possible situation, and compare the result to the behavior observed on a $(\boldsymbol{\phi}, \infty, \boldsymbol{\pi})$-execution of ρ.

Messages Between Two Protocol ITIs. A message between two ITIs M and N of the (ϕ, ∞, π)-execution corresponds to a message between $M^{\rightleftarrows \mathsf{Coord}:\rho}$ and $N^{\rightleftarrows \mathsf{Coord}:\rho}$ with the coordinator as a middle man. The coordinator acts as the control function of the (ϕ, ∞, π)-execution regarding the replacement of codes.



Note that even if N deactivates – and thus $N^{\rightleftarrows \mathsf{Coord}:\rho}$ too – $\mathsf{Coord}\left(\mathsf{s}_{\mathsf{test}}\right)$ is aware of the existence of $N^{\rightleftarrows \mathsf{Coord}:\rho}$ as it has been a middleman.

Messages to Adversary. Upon receiving input "$(\mathsf{newOperation}, w)$" from ITI $M^{\rightleftarrows \mathsf{Coord}:\rho}$, w being a backdoor with sender M, $\mathsf{Coord}\left(\mathsf{s}_{\mathsf{test}}\right)$ creates $\tilde{w}$ by replacing sender of w with $M^{\rightleftarrows \mathsf{Coord}:\rho}$ and sends output "$(\mathsf{apply}, \tilde{w})$" to $M^{\rightleftarrows \mathsf{Coord}:\rho}$.

As a consequence, when an ITI M sends a backdoor to the adversary in the (ϕ, ∞, π)-execution of ρ, $M^{\rightleftarrows \mathsf{Coord}:\rho}$ will send the same backdoor to the adversary in the execution of $\rho^{\phi\to\pi}$. Moreover, $\mathsf{Coord}\left(\mathsf{s}_{\mathsf{test}}\right)$ is aware of M.

Messages from Adversary. Messages from the adversary are sent directly to their intended recipient. The coordinator thus does not see the messages, but this is not needed as such messages do not require modification.

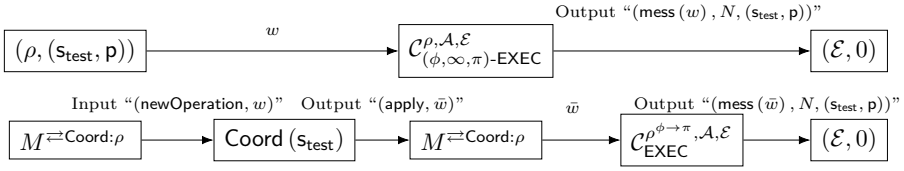
Messages from Environment. Messages from environment are sent directly to their intended recipient. The coordinator thus does not see the messages, but this is not needed as such messages do not require modification (here, we still suppose that $\rho \notin \mathsf{Pan}(\pi) \cup \mathsf{Pan}(\phi)$).

Additionally, if the environment invokes a main party of $(\rho, \mathsf{s}_{\mathsf{test}})$ that deactivates, the coordinator will remain unaware of the existence of the corresponding main party of $(\rho^{\phi\to\pi}, \mathsf{s}_{\mathsf{test}})$, as the corresponding main party will deactivate too. However, this does not impact the knowledge of $\mathsf{Ext}\left(\rho^{\phi\to\pi}, \mathsf{s}_{\mathsf{test}}\right)$ by $\mathsf{Coord}\left(\mathsf{s}_{\mathsf{test}}\right)$ as $\mathsf{MainParty}\left(\rho^{\phi\to\pi}, \mathsf{s}_{\mathsf{test}}\right) \subset \mathsf{Ext}\left(\rho^{\phi\to\pi}, \mathsf{s}_{\mathsf{test}}\right)$.

Outputs to Environment. The environment receives an output when a main party sends an output to a recipient not present in the extended session. In such cases, the control

function sends the message, the sender's identity and the recipient's extended identity to the environment.

This rule was the source of issue (I.4) in the UC operator of Plain UC. The following figure shows how the situation is handled by $\rho^{\phi\to\pi}$. Specifically, it compares a (ϕ,∞,π)-execution of ρ with environment $\mathcal{E}$ and an adversary (upper part) and the corresponding execution of $\rho^{\phi\to\pi}$ (lower part). A detailed description of the messages is provided after the figure.



In the (ϕ,∞,π) execution of ρ:

An external-write w, which is an output, is made to an ITI $N \notin \mathsf{Ext}(\rho, \mathsf{s}_{\mathsf{test}})$ by a main party $M = (\rho, (\mathsf{s}_{\mathsf{test}}, \mathsf{p}))$. This external-write is given to the control function $\mathcal{C}^{\rho,\mathcal{A},\mathcal{E}}_{(\phi,\infty,\pi)\text{-EXEC}}$, which decides to send output "$(\mathsf{mess}\,(w)\,, N, (\mathsf{s}_{\mathsf{test}}, \mathsf{p}))$" to $(\mathcal{E}, 0)$.

In the corresponding execution of Coord:

- The ITI $M = (\rho, (\mathsf{s}_{\mathsf{test}}, \mathsf{p}))$ simulated by the main party $M^{\rightleftarrows \mathsf{Coord}:\rho} = (\rho^{\phi\to\pi}, (\mathsf{s}_{\mathsf{test}}, \mathsf{p}))$ intends to send w to an ITI N, such that $N^{\rightleftarrows \mathsf{Coord}:\rho} \notin \mathsf{Ext}(\rho^{\phi\to\pi}, \mathsf{s}_{\mathsf{test}})$. $M^{\rightleftarrows \mathsf{Coord}:\rho}$ sends the input "$(\mathsf{newOperation}, w)$" to the ITI $\mathsf{Coord}\,(\mathsf{s}_{\mathsf{test}})$.
- $\mathsf{Coord}\,(\mathsf{s}_{\mathsf{test}})$ observes that $N^{\rightleftarrows \mathsf{Coord}:\rho} \notin \mathsf{Ext}(\rho^{\phi\to\pi}, \mathsf{s}_{\mathsf{test}})$ and that the message is an output from a main party of extended session $\mathsf{Ext}(\rho, \mathsf{s}_{\mathsf{test}})$. It creates $\bar{w}$, which is w with sender replaced by $M^{\rightleftarrows \mathsf{Coord}:\rho}$, while keeping the receiver as N. It then sends output "$(\mathsf{apply}, \bar{w})$" to $M^{\rightleftarrows \mathsf{Coord}:\rho}$.
- $M^{\rightleftarrows \mathsf{Coord}:\rho}$ applies $\bar{w}$. Since (ρ, π, ϕ) is replacement compatible, we have $N \notin \mathsf{Ext}(\rho^{\phi\to\pi}, \mathsf{s}_{\mathsf{test}})$, as each ITI of this extended session is of the form $U^{\rightleftarrows \mathsf{Coord}:\rho}$ for some ITI U or is $\mathsf{Coord}\,(\mathsf{s}_{\mathsf{test}})$. Additionally, $\bar{w}$ is an output from a main party of extended session $\mathsf{MainParty}(\rho^{\phi\to\pi}, \mathsf{s}_{\mathsf{test}})$. Therefore, the control function $\mathcal{C}^{\rho^{\phi\to\pi},\mathcal{A},\mathcal{E}}_{\mathsf{EXEC}}$ sends output "$(\mathsf{mess}\,(\bar{w})\,, N, (\mathsf{s}_{\mathsf{test}}, \mathsf{p}))$" to $(\mathcal{E}, 0)$

The output received by $(\mathcal{E}, 0)$ is the same in both cases.

E.4 Proof that this Operator is a UC Operator

Proposition 28. *Let $S \subset \mathsf{PPT\text{-}Param}^3$ be the subset of triplets $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ that are replacement compatible (see Definition 23). Then, the operator $(\boldsymbol{\phi}, \boldsymbol{\pi}, \boldsymbol{\rho}) \to \boldsymbol{\rho}^{\boldsymbol{\phi}\to\boldsymbol{\pi}}$, constructed in Definition 22, is an S-UC Operator in the sense of Definition 13.*

Proof. By construction, we can immediately observe that if $\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi}$ are PPT parametrized systems and that $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$ is a PPT triplet of parametrized systems, then $\boldsymbol{\rho}^{\boldsymbol{\phi}\to\boldsymbol{\pi}}$ is a PPT parametrized system.

We need to show that for any replacement compatible triplets $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$, where $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ are separated, sub-party respecting, sid-exposing, disjoint, and $\boldsymbol{\rho}$ is $(\boldsymbol{\pi}, \boldsymbol{\phi})$-compliant, then $\boldsymbol{\rho}^{\boldsymbol{\phi}\to\boldsymbol{\pi}}$ perfectly UC-emulates $(\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi})$.

For any security parameter k, we consider the injection:

$$\begin{aligned}\mathsf{AddStruct}_k : \mathsf{SetITI} &\to \mathsf{SetITI}\\ M &\mapsto M^{\rightleftarrows \mathsf{repl}[\phi_k\to\pi_k : \rho_k]:\rho_k}\ .\end{aligned}$$

$\mathsf{AddStruct}_k$ can be efficiently computed. Moreover, for any ITI M, it can efficiently be decided if $M \in \mathsf{AddStruct}_k(\mathsf{SetITI})$ and if so, the unique ITI N such that $M = \mathsf{AddStruct}_(N)$ can be efficiently found.

We can thus efficiently compute $\mathsf{InvAddStruct}_k : \mathsf{SetITI} \to \mathsf{SetITI}$ defined by:

$$\mathsf{InvAddStruct}_k(M) = \begin{cases} \perp \text{ if } M \notin \mathsf{AddStruct}_k(\mathsf{SetITI}) \,, \\ \text{The unique } N \text{ such that } \mathsf{AddStruct}_k(N) = M \text{ else.} \end{cases}$$

For $\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\pi}$ three parametrized systems, we define an adversary $\mathcal{D}_{(\boldsymbol{\rho},\boldsymbol{\phi},\boldsymbol{\pi})}$. We describe it for a fixed security parameter k (we recall that the security parameter is provided to the adversary at invocation, on a dedicated tape):

- When activated with an input "$(\mathsf{ReplacedSession}, (\mathsf{m}, M))$" from the environment $(\mathcal{E}, 0)$, where $\mathsf{ReplacedSession}$ is a subset of $\{0,1\}^*$, m is a message, and M an extended identity, $\mathcal{D}_{(\boldsymbol{\rho},\boldsymbol{\phi},\boldsymbol{\pi})}$ computes $N = \mathsf{InvAddStruct}_k(M)$. If $N = \perp$, $\mathcal{D}_{(\boldsymbol{\rho},\boldsymbol{\phi},\boldsymbol{\pi})}$ rejects. Else, $\mathcal{D}_{(\boldsymbol{\rho},\boldsymbol{\phi},\boldsymbol{\pi})}$ sends backdoor "m" to N.
- When activated with a message $(\mathsf{ReplacedSession}, \mathsf{m})$ on its backdoor tape, sent by an ITI M, $\mathcal{D}_{(\boldsymbol{\rho},\boldsymbol{\phi},\boldsymbol{\pi})}$ sends backdoor $(\mathsf{m}, \mathsf{AddStruct}_k(M))$ to the environment.

It is direct by construction that $\mathcal{D}_{(\boldsymbol{\rho},\boldsymbol{\phi},\boldsymbol{\pi})}$ is dropping : it never uses the set $\mathsf{ReplacedSession}$.

Using the explanations provided in Section E.3, we can conclude that for any PPT environment $\mathcal{E}$, any security parameter k and any $z \in \{0,1\}^*$, we have

$$\mathsf{EXEC}_{\boldsymbol{\rho}^{\boldsymbol{\phi}\to\boldsymbol{\pi}},\mathcal{D},\mathcal{E}}(k,z) = (\boldsymbol{\phi}, \infty, \boldsymbol{\pi})\text{-}\mathsf{EXEC}_{\boldsymbol{\rho},\mathcal{D}_{(\boldsymbol{\rho},\boldsymbol{\phi},\boldsymbol{\pi})},\mathcal{E}}(k,z) \,.$$

This completes the proof. □

Contents

1 Introduction **1**
1.1 Our Contributions . . . 3
1.2 Related Work . . . 4

2 Technical Overview **6**
2.1 A Flaw in the Extended Session Definition Making the UC Theorem Unusable for Practical Protocols . . . 6
2.2 Protocols with Dynamic Sub-Parties: A Family of Protocols Unfit for Plain UC 6
2.3 Panoptic Session: A Static Variant of Extended Session . . . 8
2.4 Problems with the Subroutine Exposing Property . . . 8
2.5 Extending Security Definitions Beyond Protocol Executions . . . 10
2.6 Identifying Concerns in the UC Operator and Reimagining It for Enhanced Modularity . . . 11
2.7 Revealing Hidden Scope Constraints in the UC Theorem . . . 12
2.8 Additional Contributions . . . 13

3 Preliminary Notations **13**

4 Overview of Plain UC **14**

5 Problems Within The Universal Composition Theorem **19**
5.1 Problems Related to Extended Sessions and Subroutine Respecting Property 19
5.2 Problems Related to the Universal Composition Operator . . . 24
5.3 Problems Related to the Subroutine Exposing Property . . . 25
5.4 Problem Related to the Compliance Property . . . 27

6 Updated UC: An Update of Plain UC **29**
6.1 A New Notion of Extended Session . . . 29
6.2 Replacement Execution: A (Not Entirely) New Notion of Execution . . . 30
6.3 Panoptic Session . . . 31
6.4 Sid-Exposing . . . 32
6.5 Sub-party Respecting Protocol . . . 33
6.6 Compliance . . . 33
6.7 Universal Composition Operator and Extension of UC-emulation . . . 34

7 Universal Composition Theorem **36**

References **39**

A Quick Guide to Appendices **42**

B Details About Issues, Limitations and Weaknesses Described in the Main Body **42**
B.1 Additional Notations and Conventions . . . 42
B.2 Details Regarding Issue (I.2) . . . 45
B.3 Details Regarding Issue (I.3) . . . 49
B.4 Details Regarding Issue (I.4) . . . 53
B.5 Details Regarding Weakness (W.1) . . . 55
B.6 Many examples of protocols studied in the literature that are not subroutine exposing . . . 57

C Complete Proof of UC Theorem of Updated UC (Theorem 1) 58
C.1 Additional Notations and Conventions 58
C.2 Detailed Definitions of Time Complexity 60
C.3 Relevant Prefixes 60
C.4 Proof of the Theorem Relying on the Use of a Central Lemma 66
C.5 Proof of the Central Lemma (Lemma 1) Using two Auxiliary Lemmas . . 73
C.6 Definition of Environment $\mathcal{E}_\phi$ and Polynomial Q 75
C.7 Proof of the First Auxiliary Lemma 78
C.8 Proof of the Second Auxiliary Lemma 88

D Additional Issues, Limitations, Weaknesses and Discussions 88
D.1 Two additional Issues Regarding the UC Operator 89
D.2 One additional Limitation Regarding the Subroutine Respecting property 91
D.3 A Missing Case in the Simulator of the Proof of UC Theorem of Plain UC (Weakness (W.5)) 91
D.4 Problems Related to Predicates 93
D.5 Problems Related to the Notion of T-Bounded Protocols 100
D.6 (W.8): A Weakness Related to the Notion of Balanced Environment . . . 104

E Definition of Our Universal Composition Operator 105
E.1 Main Ideas of the Construction and Definition of Our UC Operator 106
E.2 Definition of the Structured Protocol $\mu^{\rightleftarrows \mathsf{Coord}:\rho}$ 106
E.3 Definition of the Coordinator $\mathsf{Coord} = \mathsf{repl}\,[\phi \to \pi : \rho]$ 108
E.4 Proof that this Operator is a UC Operator 112