# Overcoming Impossibility Results in Composable Security using Interval-Wise Guarantees⋆

Daniel Jost 🆔 and Ueli Maurer

Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland.
{dajost, maurer}@inf.ethz.ch

**Abstract.** Composable security definitions, at times called simulation-based definitions, provide strong security guarantees that hold in any context. However, they are also met with some skepticism due to many impossibility results; goals such as commitments and zero-knowledge that are achievable in a stand-alone sense were shown to be unachievable composably (without a setup) since provably no efficient simulator exists. In particular, in the context of adaptive security, the so-called "simulator commitment problem" arises: once a party gets corrupted, an efficient simulator is unable to be consistent with its pre-corruption outputs. A natural question is whether such impossibility results are unavoidable or only artifacts of frameworks being too restrictive.

In this work, we propose a novel type of composable security statement that evades the commitment problem. Our new type is able to express the composable guarantees of schemes that previously did not have a clear composable understanding. To this end, we leverage the concept of system specifications in the Constructive Cryptography framework, capturing the conjunction of several interval-wise guarantees, each specifying the guarantees between two events. We develop the required theory and present the corresponding new composition theorem.

We present three applications of our theory. First, we show in the context of symmetric encryption with adaptive corruption how our notion naturally captures the expected confidentiality guarantee—the messages remain confidential until either party gets corrupted—and that it can be achieved by any standard semantically secure scheme (negating the need for non-committing encryption). Second, we present a composable formalization of (so far only known to be standalone secure) commitment protocols, which is instantiable without a trusted setup like a CRS. We show it to be sufficient for being used in coin tossing over the telephone, one of the early intuitive applications of commitments. Third, we reexamine a result by Hofheinz, Matt, and Maurer [Asiacrypt'15] implying that IND-ID-CPA security is not the right notion for identity-based encryption, unmasking this claim as an unnecessary framework artifact.

---

# 1 Introduction

## 1.1 A Plea for Composable Security

Common security definitions found in the literature are game-based, i.e., they require that an adversary cannot win a game that exports certain oracles to the adversary. The goal of such a security game is to capture the adversary's potential attacks in a minimal manner. However, the mapping of the game's interface to potential attacks in the real-world use of the cryptographic protocol is commonly not straight-forward. Thus, it is often a-priori unclear which game-based security notion is required in order for the protocol to be secure in a specific application. Rather, that aspect is often informally considered and passed down outside the security definitions, becoming "folklore" over the years.

Composable frameworks, such as [5,22,17,12], on the other hand, provide operational security definitions instead. The way they formalize security is based around comparing the execution of the protocol in the real world to an idealized world that intrinsically has the desired security properties. Importantly, this definition is with respect to any environment, thereby ensuring that the security guarantees not only do not exclude any attacks but also hold irrespective of other protocols (or multiple instances of the same one) being executed. For instance, the composable security definition of a symmetric encryption scheme *is* the construction of a secure communication channel from an authentic one and a key, with different assumed and constructed channels leading to different notions (e.g., whether replaying is possible). Hence, for a given application it is now trivial to decide whether a certain scheme suffices.

Finally, composable frameworks facilitate modularity. First, they are based on defining components with clean abstraction boundaries (e.g., a secure channel) that abstract away the details of how that module has been constructed (or otherwise obtained). This idealized module can then be used by a higher-level protocol with the security of the combined overall protocol following directly from the composition theorem. Thus, the security of complex protocols can be neatly proven by composing it from smaller sub-protocols.

## 1.2 Obstacles for Composable Security

While the clear semantics, modularity, and high security guarantees suggest that all protocols should be proven secure in a composable framework rather than in an ad-hoc game-based manner, composable definitions are still not prevalent with the majority of new research still carried out using game-based definitions.

One of the main reasons hindering adoption might be that many primitives are known to be impossible to achieve in the plain UC model, such as zero knowledge [5] and commitments [6]. Furthermore, Lindell has shown [13] that impossibility results are not specific to the UC model but inherent to any kind of similar model based around the existence of an efficient simulator. As a consequence, respective protocols have to rely on additional setup assumptions, such as a common reference string, and are also generally less efficient.

One particular obstacle composable definitions often face is the so-called "simulator commitment problem", which mainly arises when considering adaptive security. In a nutshell, it describes the simulator's inability to explain some of its previous choices the moment a party gets corrupted. More concretely, consider the example of two parties securing their communication using symmetric encryption. The intuition is that the adversary does not learn the messages until either of the parties gets corrupted, thereby revealing the key. Before, the adversary should learn at most the length. As a result, the simulator, in the first phase, has to output a fake ciphertexts independent of the real messages. For any semantically secure encryption scheme he can actually do so. This, however, commits him on those fake ciphertexts. At the moment a party gets corrupted, the simulator then needs to be able to explain those ciphertexts by outputting a matching encryption key. Even if he learns all the previous messages, he will not be able to do so for regular encryption schemes. Note, however, that the commitment problem is not restricted to adaptive corruptions only. Similar issues also arise, for instance, in the context of password-based security [9] or identity-based encryption [10], where it has been shown that due to this commitment problem the standard game-based notions do not induce the expected corresponding composable statements.

On a general level, this raises the fundamental question whether such impossibility results actually indicate a security issue, and hence protocols not satisfying the stronger composable definitions should not be used, or whether they present an artifact of the framework. Especially for the commitment problem, the common understanding is that the latter is true. Furthermore, the obstacles are often dealt with by either reverting to composable security with static corruptions only, or by simply retracting to game-based definitions. As a result, there is a clear need for a better composable security notion that lets us settle this question and remedy the issue of the spurious impossibilities.

### 1.3   Existing Attempts to Overcome the Obstacles

A number of approaches have been proposed in order to circumvent the aforementioned issues of composable security.

First, Canetti and Krawczyk proposed the notion of non-information oracles [8] within the UC-framework. A non-information oracle is essentially a game-based definition embedded into an ideal functionality. For instance, rather than saying that an encryption scheme should realize a secure channel that only leaks the length, the respective functionality leaks the output of the non-information oracle, which is required to satisfy a CPA-like definition. While this circumvents the commitment problem, there are two drawbacks. First, it weakens composition by requiring explicit reductions to the embedded games in the security proof of the higher-level protocols using the functionality. Second, for each ideal functionality a different type of non-information oracle needs to be defined, without providing any generic template. As a consequence, the question of the "right" non-information oracle re-arises, just like when defining a security game.

Second, a line of work considers super-polynomial simulators [21,23,4]. The initial proposal by Pass [21] considered sub-exponential simulators and polynomi-

ally bounded environments. This implies, however, that the simulator cannot be absorbed into the environment, ceding some of the most fundamental composition properties of the UC-framework. The later works by Prabhakaran and Sahai [23] and Broadnax et al. [4] empower the simulator in a more controlled manner, preserving most natural composition properties. Their adoption, however, still suffers from being rather technical, and moreover, still quite limited in the number of issues a more powerful simulator can overcome. For instance, when considering a PRG whose seed might leak, even an all powerful simulator will not be able to explain a truly randomly chosen output with an appropriate seed.

Finally, Backes, Dürmuth, Hofheinz, and Küsters [1] proposed an approach where the real-world resource would just disallow certain activation sequences by the environment that were otherwise impossible to simulate. While this avoids the complications of the other approaches, it scarifies the evident semantics of composable security notions by excluding certain—deemed artificial—attacks. A similar approach has recently been used by Jost, Maurer and Mularczyk in [11].

## 1.4 Contributions

**Interval-wise guarantees.** In this work, we propose an alternative solution to the simulator-commitment problem that is aimed at expressing the guarantees of regular schemes within a composable framework. More concretely, we introduce a novel type of construction notion within the Constructive Cryptography (CC) framework that avoids the commitment problem while providing a number of distinct benefits. First, it provides a clean semantics of how the guarantees should be interpreted. Second, it holds in any environment, just as any statement in the CC framework. Third, it is equipped with a composition theorem.

Since the commitment problem usually occurs at a very specific point of the protocol execution, such as when a party gets corrupted, where the security guarantees of the protocol anyway inherently change, our novel construction notion is centered around the very natural idea of formalizing guarantees that hold in a certain interval (between two events). That is, our notion for instance allows to formalize separate security guarantees before and after the corruption event. In contrast to existing simulation-based notions, we thereby only require the simulation to work within each interval, not forcing the simulation to be consistent between the intervals (which causes the initial commitment issues). We discuss how the security guarantees provided by our notion should be interpreted, when stronger notions might still be desirable, and how our notion fits into the space of static versus adaptive security.

**Theory extensions.** On a technical level, we leverage the specification-based approach of the CC framework, where proving a protocol $\pi$ to be secure corresponds to modeling the assumed real-world specification $\mathcal{R}$, and showing that the resulting specification $\pi\mathcal{R}$ is contained in an ideal specification $\mathcal{S}$, i.e, $\pi\mathcal{R} \subseteq \mathcal{S}$.

We formalize interval-wise guarantees as a novel type of specifications within the CC framework. We carefully consider the subtleties arising when defining such specifications and show how they interact with the other aspects of the

framework. Finally, we present the respective composition theorem, that actually supersedes all the existing ones, and in particular allows to syntactically combine multiple such interval-wise construction statements, or an interval-wise one with a regular construction statement.

**Applications.** As a third contribution, we apply our methodology to several examples. First, we consider the encrypt-then-MAC paradigm in a setting where the keys can adaptively leak to the adversary, stylizing adaptive passive corruptions. Using our interval-wise guarantees, we obtain a simple composable security definition thereof without the need for non-committing encryption. More concretely, we consider the following three properties. First, we require the messages to be confidential as long as neither the encryption nor the authentication key leaked. (An IND-CPA secure scheme cannot guarantee confidentiality without authenticity.) In our definition, this is phrased as the construction of a secure channel *up to that point*. Second, between the exposure of the encryption key and the authentication key, we require communication to still be authentic, i.e., an authenticated channel to be constructed. Finally, after the encryption key has been exposed, we still require correctness.

As a second application, we show a composable formalization of information-theoretically binding commitment schemes realizable in the plain model. We then show how, based on such a commitment scheme, Blum's protocol constructs a composable coin-toss notion. Applying composition then directly implies that this formalization can be achieved in the plain model as well. While the resulting specification is obviously too weak to serve as a common reference string, it guarantees unbiasedness. Hence, it is provides a good enough type of randomness resource whenever unbiasedness is sufficient, in particular formalizing and formally validating the intuitive-level argumentation about flipping a coin over the telephone of the corresponding papers of that time.

Finally, we consider the composable guarantees of identity-based encryption. We revisit the result by Hofheinz, Matt, and Maurer [10] that shows the standard ind-id-cpa notion to be too weak when considering a traditional composable statement based on the existence of a single simulator, even when considering *static corruptions*, due to the commitment problem. Furthermore, the authors have shown that the same weaker construction that actually can be achieved, could also be achieved by a weaker game-based notion ind-id1-cpa, modeling so-called lunch-time attacks. We refute their results in the following way: Based on interval-wise guarantees we formalize a composable specification of IBE that corresponds exactly to the standard ind-id-cpa notion.

## 1.5 Outline

In Section 3, we provide a high-level motivation and introduction to our notion, before presenting the technical details in Section 4. As a running example, we consider encrypt-then-MAC with symmetric keys that might get leaked to the adversary. In Section 5 we present a novel composable definition of perfectly

binding commitments and its application to coin tossing. Finally, in Section 6 we revisit composable security of identity-based encryption.

## 2 Preliminaries: Constructive Cryptography

This work builds upon some more recent aspect of the Constructive Cryptography (CC) framework [17,14]. We therefore revisit the key aspects thereof, follows the exposition introduced in [17], with some adaptations from [11].

### 2.1 Resources, Converters, and the Interaction Model

At its heart, the Constructive Cryptography framework views cryptography as a resource theory, in which parties use certain resources (e.g., communication channels and a shared secret key) to construct another resource via a protocol.

**Global events.** In this work, we use the version of Constructive Cryptography introduced in [11] that enriches the interaction model by a notion of globally observable events. Formally, events are a generalization of monotone binary outputs (MBO) introduced by Maurer et al. [16]. Roughly, an MBO is a value that can change from 0 to 1 but not back, which can be interpreted as a single event happening once the MBO changes to 1. An event then just corresponds to a named MBO and the *global event history* $\mathcal{E}$ is a list of event names without duplicates (to model that every event can occur at most once). For an event name $n$, we denote by $\mathcal{E} \xleftarrow{+} \mathcal{E}_n$ the act of appending $n$ to $\mathcal{E}$ (or leaving it unchanged if it is already contained). Moreover, we use $\mathcal{E}_n$ as a short-hand notation to denote that $n$ is in the list $\mathcal{E}$, and say that the event happened. Finally, we denote by $\mathcal{E}_{n_1} \prec \mathcal{E}_{n_2}$ that the event $n_1$ precedes the event $n_2$ in the event history.

**Resources.** A resource[1] R is a reactive system that interacts in the following two ways with the rest of the world: First, it allows interaction at one or several named *communication interfaces*, in the following just called interfaces, at which it can be queried an input $x$, and must answer with an output $y$ at the same interface. Second, during an activation, the resource R can depend on the global event history $\mathcal{E}$, and can furthermore append events from a predefined set of names. We call this set of names the events controlled by R.

Formally, resources are modeled as random systems [15], where the interface address, the actual input $x$, and the current state of the event history are encoded as part of the input. Analogously, the answer $y$ and the new state of the event history are encoded as part of the output, under the constraint that the old state of the event history is a prefix of the new one. For the sake of this paper, a reader unfamiliar with the CC framework might however just think of a resource as the behavior of an oracle machine, where each interface corresponds to an oracle and

---

[1] The analogon to *functionalities* in the UC framework [5].

the event history being similar to the "directory" ITI used in the recent version of UC [5]. Note, however, that a resource only defines the behavior of the system and not its description, i.e., two different (pseudo-code descriptions of) ITMs having the same input-output behavior denote the same resource.

A set of resources can be viewed as a single one, with the interface set of the composed resource being the union. For resources $R_1, \ldots, R_n$ (with disjoint interface sets) we denote by $[R_1, \ldots, R_n]$ the *parallel composition*.

**Converters and protocols.** In the Constructive Cryptography framework, *converters* express the local action executed by one party. A converter expects to be connected to a given number of interfaces at the "inside", and emulates a certain set of interfaces at the "outside". Upon an input at an outside interface, the converter is allowed to make a bounded number of oracle queries to the inside interfaces, before returning a value at the queried interface.

For a converter $\pi$ and a resource $R$, let $\mathcal{I}$ denote a tuple describing an injective mapping from $\pi$'s inside interfaces to interfaces of $R$. We then denote by $R' := \pi^{\mathcal{I}} R$ the resource obtained from connecting the converter accordingly. The resource $R'$ no longer exposes those interfaces but the ones emulated by $\pi$ instead. Converter attachment satisfies the natural property of *composition order independence*, stating that the composition order does not matter—only the final system does.

**Proposition 1.** *Let $\pi_1$ and $\pi_2$ be two converters, let $R$ be a resource and let $\mathcal{I}_1$ and $\mathcal{I}_2$ be such that they assign* disjoint *interfaces. Then, $\pi_1^{\mathcal{I}_1} \pi_2^{\mathcal{I}_2} R = \pi_2^{\mathcal{I}_2} \pi_1^{\mathcal{I}_1} R$. Moreover, if $S$ is another resource such that the interface sets of $R$ and $S$ are disjoint, then we have $\pi_1^{\mathcal{I}_1} [R, S] = [\pi_1^{\mathcal{I}_1} R, S]$.*

We define a *protocol* to be a set of converter-connection pairs, i.e,. $\boldsymbol{\pi} := \{(\pi_1, \mathcal{I}_1), \ldots, (\pi_n, \mathcal{I}_n)\}$ with pairwise disjoint $\mathcal{I}_i$'s. Moreover, we say that $\boldsymbol{\pi}$ is a protocol for a resource $R$, if $R$ has all the required interfaces for the protocol application to be well-defined, and write $\boldsymbol{\pi} R$ to denote its application.

**The environment (distinguisher).** The distinguisher $D$ is a special type of environment that first interacts with a resource $R$ by making queries to the resource's interfaces. Between two such queries it can access the global event history and append events to it, except the ones controlled by $R$. Note that activations are atomic, i.e., at any moment in time either the resource or the distinguisher is activated, but not both. Finally, the distinguisher ends the interaction with the resource by outputting a bit. The advantage of $D$ is then defined as

$$\Delta^D(R, \ S) := \Pr[D^{\mathcal{E}}(S) = 1] - \Pr[D^{\mathcal{E}}(R) = 1],$$

where we use the syntax $D^{\mathcal{E}}(\cdot)$ to make explicit that the distinguisher has oracle access to the global event history $\mathcal{E}$.

## 2.2 Constructions

**Specifications.** It is natural to consider only certain desired (or assumed) properties of a system and deliberately not specify others. Some of those choices

are intrinsic to the mathematical model we use, such as only considering the input-output behavior and ignoring the physical aspects. Other properties can be purposefully ignored by considering specifications of systems that simply leave out those aspects, focusing only on the relevant properties. Following [18], we model specifications as sets of resources $\mathcal{R}$ that all have the same interface set. For each property, such as confidentiality, one has in mind, one can consider the set $\mathcal{R}$ of resources satisfying that property. Vice versa, each set of resources $\mathcal{R}$ can be interpreted as the set of properties common to all elements. For instance, authenticated communication might be modeled as the set of all communication channels that are authentic—not specifying the level of confidentiality by including both confidential as well as non-confidential channels.

**Constructions as subsets.** In provable security one typically considers the execution of a protocol $\pi$ that makes use of some assumed specification $\mathcal{R}$, such as a communication network or a public-key infrastructure. In short, one wants to show that the specification $\pi\mathcal{R} \coloneqq \{\pi\mathsf{R} \mid \mathsf{R} \in \mathcal{R}\}$ satisfies the desired security properties. As explained in the previous section, those properties are formalized as a specification $\mathcal{S}$, and thus proving security means proving $\pi\mathcal{R} \subseteq \mathcal{S}$. Note that obviously the guarantees given by $\mathcal{S}$ are generally weaker than the ones by $\pi\mathcal{R}$. The purpose of such a statement is, however, that the security properties are in $\mathcal{S}$ both more explicit and simpler to analyze. In other words, the goal is to distill out the relevant properties and abstract away the others.

Traditionally, the statement $\pi\mathcal{R} \subseteq \mathcal{S}$ is read as the protocol $\pi$ constructing the specification $\mathcal{S}$ from the specification $\mathcal{R}$, or in UC-jargon the protocol securely realizing the specification $\mathcal{S}$ in the $\mathcal{R}$-hybrid model. Hence, as a shorthand notation we introduce the following construction notion.

**Definition 1.** *Let $\mathcal{R}$ and $\mathcal{S}$ be specifications, and let $\pi$ be a protocol for $\mathcal{R}$. Then, we say that $\pi$ constructs $\mathcal{S}$ from $\mathcal{R}$, denoted $\mathcal{R} \xrightarrow{\pi} \mathcal{S}$, if and only if $\pi\mathcal{R} \subseteq \mathcal{S}$, i.e.,*

$$\mathcal{R} \xrightarrow{\pi} \mathcal{S} \ :\Longleftrightarrow\ \pi\mathcal{R} \subseteq \mathcal{S}.$$

*In slight abuse of notation, we write $\mathsf{R} \xrightarrow{\pi} \mathsf{S}$ in lieu of $\{\mathsf{R}\} \xrightarrow{\pi} \{\mathsf{S}\}$ for singleton specifications.*

This construction notion is associated with the usual composition properties of Constructive Cryptography: sequential and parallel composition—which form the equivalence of the universal composition theorem of the UC-framework.

**Theorem 1.** *Let $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$ be arbitrary specifications, and let $\pi$ and $\pi'$ be arbitrary protocols for $\mathcal{R}$ and $\mathcal{S}$, respectively. Then, we have*

1. $\mathcal{R} \xrightarrow{\pi} \mathcal{S} \wedge \mathcal{S} \xrightarrow{\pi'} \mathcal{T} \implies \mathcal{R} \xrightarrow{\pi' \circ \pi} \mathcal{T}$,
2. $\mathcal{R} \xrightarrow{\pi} \mathcal{S} \implies [\mathcal{R}, \mathcal{T}] \xrightarrow{\pi} [\mathcal{S}, \mathcal{T}]$.

*Proof.* The first property follows directly from the transitivity of the subset relation, $\pi'(\pi\mathcal{R}) \subseteq \pi'\mathcal{S} \subseteq \mathcal{T}$, and the second property follows from Proposition 1: $\pi[\mathcal{R}, \mathcal{T}] = [\pi\mathcal{R}, \mathcal{T}] \subseteq [\mathcal{S}, \mathcal{T}]$.

The specifications $\boldsymbol{\pi}\mathcal{R}$ and $\mathcal{S}$ are often referred to as real- and ideal-world, respectively, according to the so-called real-world/ideal-world paradigm on which most composable frameworks [5,22,17,12] are based. Following that paradigm, security statement affirm that the real word is "just-as-good" as the ideal world, meaning that for all parties, no matter whether honest or adversarial, it does not make a difference whether they live in the real (where an arbitrary element of $\boldsymbol{\pi}\mathcal{R}$ is present), or in the ideal world (where some element of $\mathcal{S}$ is present). Hence, if the honest parties are content with the guarantees they get from the ideal specification, they can safely execute the protocol in the real world instead.

**The (in)existence of a simulator.** Simulation-based security turned out to be one of the most fundamental concepts in cryptography and is closely linked with the real-world / ideal-world paradigm. It not only forms the foundation of semantic security, zero knowledge, and the security of MPC, but also of virtually every composable framework. Whereas the former definitions tend to require an after-the-fact simulation of the transcript, composable frameworks get their stronger guarantees from requiring on-line simulation, where an adaptive environment interacts with the simulator. The common understanding of those security definitions is then that the simulator "translates" the attacks from the real-world adversary to the ideal world such that they achieve the same effect.

While the initial version of the Constructive Cryptography framework also hard-coded the existence of a simulator (with respect to the dummy adversary), starting from [18], the simulator is no longer an integral part of the construction notion. Rather, employing a simulator is just one way of defining an ideal specification, $\boldsymbol{\sigma}\mathcal{S}$ that makes the achieved security properties obvious. For instance, the specification of confidential channels can then be written as the specification $\mathcal{S}$ of channels that only leak the message length, combined with an arbitrary simulator. From this description it is apparent that for any resource in the combined specification $\boldsymbol{\sigma}\mathcal{S}$, only the length is leaked.

If one restricts oneself to specifications of this type, then the following more specific composition theorem can be deduced.

**Proposition 2.** *Let $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$ be specifications, and let $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$ be protocols for $\mathcal{R}$ and $\mathcal{S}$, respectively. For any simulators $\boldsymbol{\sigma}$ (for $\mathcal{S}$) and $\boldsymbol{\sigma}'$ (for $\mathcal{T}$), such that the set of interfaces controlled by the simulators are disjoint from the ones controlled by the protocols, we have*

1. $\mathcal{R} \xrightarrow{\boldsymbol{\pi}} \boldsymbol{\sigma}\mathcal{S} \wedge \mathcal{S} \xrightarrow{\boldsymbol{\pi}'} \boldsymbol{\sigma}'\mathcal{T} \implies \mathcal{R} \xrightarrow{\boldsymbol{\pi}'\circ\boldsymbol{\pi}} \boldsymbol{\sigma}\boldsymbol{\sigma}'\mathcal{T},$
2. $\mathcal{R} \xrightarrow{\boldsymbol{\pi}} \boldsymbol{\sigma}\mathcal{S} \implies [\mathcal{R}, \mathcal{T}] \xrightarrow{\boldsymbol{\pi}} \boldsymbol{\sigma}[\mathcal{S}, \mathcal{T}].$

*Proof.* By composition order invariance we have $\boldsymbol{\pi}'\boldsymbol{\sigma}\mathcal{S} = \boldsymbol{\sigma}\boldsymbol{\pi}'\mathcal{S} \subseteq \boldsymbol{\sigma}\boldsymbol{\sigma}'\mathcal{T}$, implying $\mathcal{S} \xrightarrow{\boldsymbol{\pi}'} \boldsymbol{\sigma}'\mathcal{T} \implies \boldsymbol{\sigma}\mathcal{S} \xrightarrow{\boldsymbol{\pi}'} \boldsymbol{\sigma}\boldsymbol{\sigma}'\mathcal{T}$. The first property then follows directly from combining this with Theorem 1. The second property follows from Theorem 1 and Proposition 1 as well: $\boldsymbol{\pi}[\mathcal{R}, \mathcal{T}] = [\boldsymbol{\pi}\mathcal{R}, \mathcal{T}] \subseteq [\boldsymbol{\sigma}\mathcal{S}, \mathcal{T}] = \boldsymbol{\sigma}[\mathcal{S}, \mathcal{T}]$. $\qquad\square$

## 2.3 Relaxations

The basic construction notion does not take into account statistical errors or computational assumptions. Those aspects are formalized by so-called relaxations, as introduced in [18]. On an abstract level, a relaxation is a mapping from specifications to weaker, so-called relaxed, specifications. For our purpose, where we instantiate specifications by sets of resources, we can define a relaxation as a function mapping a single resource to a set of resources.

**Definition 2.** *Let $\Theta$ denote the set of all resources. A* relaxation $\phi$ *is a function* $\phi\colon \Theta \to 2^\Theta$ *(where $2^\Theta$ denotes the power set of $\Theta$) such that* $\mathsf{R} \in \phi(\mathsf{R})$ *for all* $\mathsf{R} \in \Theta$. *In addition, for a specification $\mathcal{R}$, we define $\mathcal{R}^\phi := \bigcup_{\mathsf{R} \in \mathcal{R}} \phi(\mathsf{R})$ as a shorthand notation.*

A concrete relaxation thereby formalizes some notion of resources being "almost-as-good" in some context. That is, if we were happy with constructing a resource specification $\mathcal{S}$, then we should also be happy with $\mathcal{S}^\phi$, if we believe the weakening $\phi$ to be justifiable in the given context. For instance, one could consider the relaxation that maps the resource $\mathsf{R}$ to the set of all computationally indistinguishable resources from $\mathsf{R}$ under some computational assumption. Hence, if we believe the computational assumption to be valid, we should be as content with the relaxed specification as with the original one.

Abstracting away irrelevant properties is a core paradigm of any modular analysis. Applied to Constructive Cryptography, this means that ideally we should be able to "forget" relaxations. That is, if one shows that one protocol constructs $\mathcal{S}^\phi$ (from some assumed resources), one should be able to compose it with another statement that assumes $\mathcal{S}$ instead. On the most abstract level, it is easy to see that the following rules apply to any relaxation.

**Proposition 3.** *For any specifications $\mathcal{R}$ and $\mathcal{S}$, and any relaxation $\phi$, we have*

1. $\mathcal{R} \subseteq \mathcal{R}^\phi$,
2. $\mathcal{R} \subseteq \mathcal{S} \implies \mathcal{R}^\phi \subseteq \mathcal{S}^\phi$,
3. $(\mathcal{R} \cap \mathcal{S})^\phi \subseteq \mathcal{R}^\phi \cap \mathcal{S}^\phi$,
4. $(\mathcal{R} \cup \mathcal{S})^\phi = \mathcal{R}^\phi \cup \mathcal{S}^\phi$.

*Proof.* All properties trivially follow from $\mathsf{R} \in \phi(\mathsf{R})$.

**The reduction relaxation.** We now introduce the most fundamental relaxation, which captures computational security based on explicit reductions. This is defined as a function $\epsilon$ that maps distinguishers to their respective performance in $[0, 1]$, where $\epsilon(\mathsf{D})$ typically refers to the winning probability of a modified distinguisher $\mathsf{D}'$ (the reduction) on the underlying computational problem.

**Definition 3.** *Let $\epsilon$ be a function that maps distinguishers to a value in $[0, 1]$. Then, the induced relaxation on a resource $\mathsf{R}$, denoted $\mathsf{R}^\epsilon$, is defined as*

$$\mathsf{R}^\epsilon := \big\{ \mathsf{S} \mid \forall \mathsf{D} : |\Delta^\mathsf{D}(\mathsf{R}, \mathsf{S})| \leq \epsilon(\mathsf{D}) \big\}.$$

*We call such a relaxation generally an $\epsilon$-relaxation or reduction relaxation.*

We now discuss several properties that $\epsilon$-relaxations have. First, the errors just add up, as expressed by the following theorem.

**Theorem 2.** *Let $\mathcal{R}$ be an arbitrary specification, and let $\epsilon_1$ and $\epsilon_2$ be arbitrary $\epsilon$-relaxations. Then we have $\left(\mathcal{R}^{\epsilon_1}\right)^{\epsilon_2} \subseteq \mathcal{R}^{\epsilon_1 + \epsilon_2}$.*

*Proof.* This follows directly from the triangle inequality of the distinguishing advantage. A more detailed proof is given in Appendix A.1. □

Second, they naturally commute with protocol application and parallel composition of additional resources, i.e., the relaxation can be "pulled out". In such a step, however, the additional resource or converter has to be explicitly accounted for in the reduction.

**Theorem 3.** *The $\epsilon$-relaxation is* compatible *with protocol application in the following sense that $\boldsymbol{\pi}\left(\mathcal{R}^{\epsilon}\right) \subseteq \left(\boldsymbol{\pi}\mathcal{R}\right)^{\epsilon_{\boldsymbol{\pi}}}$, for $\epsilon_{\boldsymbol{\pi}}(\mathsf{D}) \coloneqq \epsilon(\mathsf{D}\boldsymbol{\pi}(\cdot))$, where $\mathsf{D}\boldsymbol{\pi}(\cdot)$ denotes the distinguisher that first attaches $\boldsymbol{\pi}$ to the given resource and then executes $\mathsf{D}$. Moreover, it is compatible with parallel composition, i.e., $\left[\mathcal{R}^{\epsilon}, \mathcal{S}\right] \subseteq \left[\mathcal{R}, \mathcal{S}\right]^{\epsilon_{\mathcal{S}}}$, for $\epsilon_{\mathcal{S}}(\mathsf{D}) \coloneqq \sup_{\mathsf{S} \in \mathcal{S}} \epsilon(\mathsf{D}[\,\cdot\,, \mathsf{S}])$, where $\mathsf{D}[\,\cdot\,, \mathsf{S}]$ denotes the distinguisher that emulates $\mathsf{S}$ in parallel to the given resource and then lets $\mathsf{D}$ interact with them.*

*Proof.* The proof is given in Appendix A.2. □

The composition theorem with $\epsilon$-relaxations then follows directly from these compatibility results. The following corollary phrases the corresponding result—which in older version of Constructive Cryptography used to be called *the* composition theorem, thereby hard-coding computational security.

**Corollary 1.** *For any specifications $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$, any protocols $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$, and any $\epsilon$-relaxation $\epsilon$ and $\epsilon'$, we have*

*1. $\mathcal{R} \xrightarrow{\boldsymbol{\pi}} \mathcal{S}^{\epsilon} \wedge \mathcal{S} \xrightarrow{\boldsymbol{\pi}'} \mathcal{T}^{\epsilon'} \implies \mathcal{R} \xrightarrow{\boldsymbol{\pi}' \circ \boldsymbol{\pi}} \mathcal{T}^{\epsilon_{\boldsymbol{\pi}'} + \epsilon'}$,*
*2. $\mathcal{R} \xrightarrow{\boldsymbol{\pi}} \mathcal{S}^{\epsilon} \implies [\mathcal{R}, \mathcal{T}] \xrightarrow{\boldsymbol{\pi}} [\mathcal{S}, \mathcal{T}]^{\epsilon_{\mathcal{T}}}$,*

*where $\epsilon_{\boldsymbol{\pi}}$ and $\epsilon_{\mathcal{S}}$ are defined as in Theorem 3, respectively.*

*Proof.* The proof can be found in Appendix A.3. □

## 3 Interval-Wise Guarantees: Motivation and Intuition

In this section, we outline the general approach, and its motivation, proposed in this work, before we deep dive into the technicalities in Section 4. In particular, we believe that the conceptual contributions are of interest independent from the exact mathematical formalization.

### 3.1 A Motivating Example

Consider two parties, Alice and Bob, who want to communicate securely over the Internet. If they have a pre-shared secret key available, e.g. from running a key agreement protocol, then it is well known that the encrypt-then-MAC paradigm achieves the desired goal. Assuming independent keys for the encryption and MAC scheme, this construction is secure if the underlying encryption scheme is IND-CPA secure and the MAC scheme is weakly unforgeable.

What, however, if we assume that in reality the keys to not be one hundred percent secure? Intuitively one should expect the scheme to remain secure until either of the keys leak to an adversary, and the security properties then to gracefully downgrade accordingly. More concretely, there is little reason to doubt the following security guarantees should be provided by the scheme:

1. until either of the keys leak, the scheme should provide both confidentiality and authenticity;
2. if only the encryption key leaked so far, then the scheme should still provide authenticity;
3. once the MAC key leaked, the scheme should at least still provide correctness, i.e., allow the parties to communicate in the absence of an active network attack.

(Note that if first the MAC key gets exposed, then a scheme that is only IND-CPA secure might not provide full confidentiality.)

### 3.2 A Naive Attempt

While the encrypt-then-MAC paradigm has composably proven to be sound in a context where both parties are honest and the keys are secure (e.g. [8,19]), extending those results to deal with key exposures has turned out to be surprisingly strenuous.

Intuitively, one might model the achieved security guarantees as an *secure channel with downgradable security*, which waives confidentiality and authenticity once the respective keys leaked. The protocol should then construct such a channel from an insecure channel and two leakable keys[2], for authentication and encryption, respectively. See Figure 1 for a formal definition of the respective resources InsecCh, AuthKey, and EncKey (for the assumed resources), and SecCh for the secure channel with downgradable security. Following the paradigm of modularity, one might try to formalize and prove this in two steps and first consider authentication only, as modeled by a downgradable authenticated channel AuthCh (c.f. Figure 1 as well). Indeed, one can show the following construction.

---

[2] In CC, the adversary by definition only has access to interfaces statically assigned to him. Hence, adaptive corruptions are modeled by introducing explicit memory and computation resources with an adversarial interface, granting the adversary access once the party is corrupted. For simplicity, we here consider directly leaking key resources instead. Assuming secure erasure, this is equivalent to passive corruptions.

## Resource InsecCh

**Initialization**
$M_A[\cdot] \leftarrow \bot$
$m_B \leftarrow \bot$
$n \leftarrow 0$

**Interface A**
**Input:** $(\texttt{send}, m)$
  $n \leftarrow n + 1$
  $M_A[n] \leftarrow m$
  **output** ok

**Interface B**
**Input:** receive
  **output** $m_B$

**Interface E**
**Input:** $(\texttt{leak}, i)$
  **output** $M_A[i]$

**Input:** $(\texttt{inject}, m)$
  $m_B \leftarrow m$
  **output** ok

**Interface W**
**Input:** $(\texttt{deliver}, i)$
  $m_B \leftarrow M_A[i]$
  **output** ok

## Resource AuthCh

**Initialization**
$M_A[\cdot] \leftarrow \bot$
$m_B \leftarrow \bot$
$n \leftarrow 0$

**Interface A**
**Input:** $(\texttt{send}, m)$
  $n \leftarrow n + 1$
  $M_A[n] \leftarrow m$
  **output** ok

**Interface B**
**Input:** receive
  **output** $m_B$

**Interface E**
**Input:** $(\texttt{leak}, j)$
  **output** $M_A[j]$

**Input:** $(\texttt{inject}, m)$
  **if** $\mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}$ **then**
  | $m_B \leftarrow m$
  **else**
  | $m_B \leftarrow \bot$
  **output** ok

**Interface** $i \in \{\texttt{E}, \texttt{W}\}$
**Input:** $(\texttt{deliver}, j)$
  $m_B \leftarrow M_A[j]$
  **output** ok

## Resource SecCh

**Initialization**
$M_A[\cdot] \leftarrow \bot$
$m_B \leftarrow \bot$
$n \leftarrow 0$

**Interface A**
**Input:** $(\texttt{send}, m)$
  $n \leftarrow n + 1$
  $M_A[s] \leftarrow m$
  **output** ok

**Interface B**
**Input:** receive
  **output** $m_B$

**Interface E**
**Input:** $(\texttt{leak}, j)$
  **if** $M_A[j] = \bot$ **then**
  | **output** $\bot$
  **else if** $\mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}} \vee \mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}$ **then**
  | **output** $M_A[j]$
  **else**
  | **output** $|M_A[j]|$

**Input:** $(\texttt{inject}, m)$
  **if** $\mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}$ **then** $m_B \leftarrow m$
  **else** $m_B \leftarrow \bot$
  **output** ok

**Interface** $i \in \{\texttt{E}, \texttt{W}\}$
**Input:** $(\texttt{deliver}, j)$
  $m_B \leftarrow M_A[j]$
  **output** ok

## Resource ResKey

for $\mathsf{Res} \in \{\mathsf{Auth}, \mathsf{Enc}\}$

**Initialization**
$k \leftarrow \mathcal{K}$

**Interface** $i \in \{\texttt{A}, \texttt{B}\}$
**Input:** fetch
  **output** $k$

**Interface E**
**Input:** read
  **if** $\mathcal{E}_{\mathsf{ResKey}}^{\mathsf{leaked}}$ **then**
  | **output** $k$
  **else**
  | **output** $\bot$

Fig. 1: The resources involved in the encrypt-then-MAC example. Observe how the authenticated and the secure channel degrade their guarantees once the respective keys have been leaked.

**Proposition 4.** *Let* AuthCh *denote the authenticated channel that degrades its security once the respective key is leaked, as formally defined in Figure 1, and let* $\boldsymbol{\pi}_{\mathsf{MAC}}$ *denote the simple protocol that applies a MAC scheme to the messages. Then, there exists a simulator* $\boldsymbol{\sigma}_{\mathsf{MAC}}$ *such that*

$$\big[\mathsf{AuthKey}, \mathsf{InsecCh}\big] \xrightarrow{\ \boldsymbol{\pi}_{\mathsf{MAC}}\ } \big(\boldsymbol{\sigma}_{\mathsf{MAC}}\mathsf{AuthCh}\big)^{\epsilon_{\mathsf{MAC}}},$$

*where* $\epsilon_{\mathsf{MAC}}$ *denotes a simple reduction to the MAC-forgery game.*

*Proof.* This is a well-known result, which has for instance been sketched in [14].

13

However, once we turn our attention towards the second construction step—using encryption to achieve confidentiality—we run into the so-called simulator commitment problem of composable security, as expressed by the following proposition.

**Proposition 5.** *Let* SecCh *denote a secure channel that degrades the respective guarantees once the keys have been exposed, as depicted in [Figure 1](), and let* $\pi_{\mathsf{ENC}}$ *be the protocol that applies a symmetric encryption scheme. For any (efficient) simulator* $\sigma_{\mathsf{ENC}}$, *and (an efficiency preserving) reduction* $\epsilon_{\mathsf{CPA}}$ *to the IND-CPA game, we have*

$$\left[\mathsf{EncKey}, \mathsf{AuthCh}\right] \xrightarrow{\ \pi_{\mathsf{ENC}}\ } \left(\sigma_{\mathsf{ENC}}\mathsf{SecCh}\right)^{\epsilon_{\mathsf{CPA}}},$$

*i.e., IND-CPA security does not suffice.*

*Proof (Sketch).* In the first phase, the simulator has to produce, without knowing the message, fake ciphertexts $c_1, c_2, \ldots, c_n$ that look indistinguishable from the real one. For an IND-CPA secure scheme, he can easily do so by encrypting an arbitrary message of the correct length. The moment the encryption key leaks in the real world, he however has to output a uniformly looking key that makes his ciphertexts decrypt to the correct messages. Even knowing the messages by now, this is infeasible unless we assume a non-committing encryption scheme. Furthermore, as long as the requested key is shorter than $n$, Nielsen [20] showed that NCE cannot be achieved in the standard model by non-interactive protocols.

Of course one could avoid this impossibility by utilizing stronger primitives and/or assumptions, such as non-committing encryption. In some contexts, such as when considering deniability, their stronger guarantees might even be inherently necessary. In this work, we however pose the following question: How can we express the aforementioned security guarantees, that the encrypt-then-MAC paradigm using regular encryption intuitively does provide, in a composable framework? That is, rather than establishing a stronger security notion, we aim at expressing the exact guarantees provided by existing game-based notions.

### 3.3 Our Solution

So how to express the natural properties that are achieved? First, let us have another look at the reason for the impossibility: traditional simulation-based security notions require the simulator to commit to a ciphertext, emulating the encryption, based on the length only. Even if the simulator later gets to learn the entire message, it cannot come up with an encryption key that decrypts the previously output ciphertext to this message. Observe, however, that outputting the length only is just a technical way of expressing confidentiality until either one of the keys leak. In principle, there is no inherent requirement for a consistent simulation strategy across the different phases of the experiments. This is exactly what our proposal of interval-wise guarantees builds on: allowing disjoint simulation strategies for different phases of a protocol run. In other words, we simply make three disjoint security statements, one guaranteeing confidentiality and

authenticity until either key is leaked, one only guaranteeing authenticity between the exposure of the encryption key and the MAC key, and one guaranteeing correct delivery of messages afterwards. Given the specification centric approach of Constructive Cryptography, this can be phrased as

$$\pi_{\mathsf{ENC}}\pi_{\mathsf{MAC}}[\mathsf{AuthKey}, \mathsf{EncKey}, \mathsf{InsecCh}] \subseteq \mathcal{S}_1 \cap \mathcal{S}_2 \cap \mathcal{S}_3,$$

where $\mathcal{S}_1$ to $\mathcal{S}_3$ are specifications formalizing the respective guarantees.

Phrasing separate statements can trivially be done in any framework, but also comes with a number of drawbacks. First, having to specify three constructions of unconnected, potentially differently described, specifications incurs a certain cognitive overhead, making the overall achieved security more demanding to understand. Second, and more severely, one loses some compositional properties. In particular, the analysis of another protocol building on top of those guarantees would require to make the exact same case distinction.

To overcome those drawbacks, we phrase each guarantee as an appropriate *interval-wise relaxation* of the same underlying resource: the downgradable secure channel. That is, we phrase security as

$$\pi_{\mathsf{ENC}}\pi_{\mathsf{MAC}}[\mathsf{AuthKey}, \mathsf{EncKey}, \mathsf{InsecCh}] \subseteq \mathsf{SecCh}^{\phi_1} \cap \mathsf{SecCh}^{\phi_2} \cap \mathsf{SecCh}^{\phi_3},$$

where $\phi_1$ to $\phi_3$ formalize the interval-wise relaxations. Another protocol can then simply assume the overly idealized downgradable secure channel $\mathsf{SecCh}$, with our novel composition theorem taking care of devising the appropriate overall security statement. We formalize this type of relaxation and the corresponding composition theorem in the next section, i.e., Section 4.

Translating the approach to another composable framework, such as UC, might be feasible but non-trivial. First, one might try to formalize a single interval-wise guarantee as a different corruption model, where for instance the adversary simply does not get the encryption key to securely realize a functionality analogon to $\mathsf{SecCh}$. To then compose this step with a $\mathsf{SecCh}$-hybrid statement, one would probably require some compiler translating the statement. We, thus, believe that formalizing our results in CC that allows for arbitrary specifications is both simpler and more natural.

**A remark on adaptive versus static security.** Our security statement makes a static case separation on the intervals considered. This might raise the question as to how this differs from simply considering static corruptions only. We would like to stress that our statement is about a real-world system, where the environment gets to adaptively (depending on all the outputs it sees) choose when the appropriate keys are leaked. Hence, our notion lies somewhere in between the traditional notions of static and adaptive security.

To which extent our notion suffices in practice, and when a stronger traditional adaptive statement is required, is in our opinion an interesting open research problem. On the one hand, fully adaptively secure notions, without doubt, play a crucial role as a technical tool in many cryptographic constructions.

15

On the other hand, very few cases are known where the overall security of an application actually seems to be meaningfully impacted by adaptiveness. For instance, consider the folklore example of an MPC protocol where an adversary knows which party she has to corrupt based on some observed value during the execution. Nevertheless, for a polynomially sized adversary structure (i.e., choices which parties to corrupt), the adversary could still guess upfront, implying that even traditional static security would suffice. This is for instance the case if there are only logarithmically (or constant) many parties overall.

Moreover, even if there super-polynomially many choices, it could still be that our interpretation of the static result is wrong: if we distinguish $n$ static cases, and in each one of them a certain property is violated with probability $\epsilon$, then all we can say is that by the union bound the probability of a property being violated is bounded by $n\epsilon$. Hence, concluding from $\epsilon$ being negligible that the protocol is overall secure, might simply not be sound in the first place.

**A remark on stronger security guarantees.** The primary goal of this work is to express the security guarantee of certain schemes in a composable framework, for which so far this has not been possible. This does not contradict stronger security notions, such as non-committing encryption, being of use as well. For instance, insisting that the simulator can explain the ciphertexts (in the traditional notion) formalizes that the ciphertexts are never of any value—in a broader sense than confidentiality. This might play an important role in advanced properties such as deniability, or e.g. in a scenario where an adversary wants to prove to another party that he managed to wiretap the channel before the transmitted message and the corresponding encryption key are publicly announced. Phrasing that no adversary can succeed requires the simulator to work beyond the public announcement, and achieving it requires non-committing encryption. Otherwise, committing to the ciphertext ahead of the public announcement should convince the other party.

## 4 Interval-Wise Guarantees: Definitions

In this section, we formalize interval-wise guarantees as a type of relaxation and provide the corresponding composition theorem. In the spirit of modularity, we proceed in several steps. First, we introduce one relaxation that waives all guarantees after a certain point, and second, the complementary one that waives all guarantees before a certain event. Third, we combine those relaxations and show that it fits well into the existing theory. Finally, we present the resulting construction notion and phrase the motivating example therein.

### 4.1 Guarantees up to Some Point

As we have seen in the motivational example, the confidentiality of the messages should be guaranteed *until* the key is leaked. To phrase this, we, on a high level, only require that the simulator works up to this event. We formalize this as a

**Resource** $\mathrm{until}_{\mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}} \vee \mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}}}(\mathsf{SecCh})$

**Initialization**
$M_{\mathtt{A}}[\cdot] \leftarrow \perp$
$m_{\mathtt{B}} \leftarrow \perp$
$n \leftarrow 0$

**Interface A**
**Input:** $(\mathtt{send}, m)$
  **require** $\neg\left(\mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}} \vee \mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}\right)$
  $n \leftarrow n + 1$
  $M_{\mathtt{A}}[n] \leftarrow m$
  **output** $\mathtt{ok}$

**Interface B**
**Input:** $\mathtt{receive}$
  **require** $\neg\left(\mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}} \vee \mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}\right)$
  **output** $m_{\mathtt{B}}$

**Interface E**
**Input:** $(\mathtt{leak}, j)$
  **require** $\neg\left(\mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}} \vee \mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}\right) \wedge M_{\mathtt{A}}[j] \neq \perp$
  **output** $|M_{\mathtt{A}}[j]|$

**Input:** $(\mathtt{inject}, m)$
  **output** $\perp$

**Interface** $i \in \{\mathtt{E}, \mathtt{W}\}$
**Input:** $(\mathtt{deliver}, j)$
  **require** $\neg\left(\mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}} \vee \mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}\right)$
  $m_{\mathtt{B}} \leftarrow M_{\mathtt{A}}[j]$
  **output** $\mathtt{ok}$

where **require** *cond* is a short-hand notation for **if** $\neg cond$ **output** $\perp$

Fig. 2: The secure channel from Figure 1 when halted once either key leaks. In contrast to the original one, this resource never leaks the actual messages.

novel type of relaxation consisting of all systems behaving equally up to this point. To this end, for a resource $\mathsf{R}$, we consider the modified resource that halts once a certain predicate on the global event history is satisfied.

**Definition 4.** *Let* $\mathsf{R}$ *denote a resource, and let* $P(\mathcal{E})$ *denote a monotone predicate on the global event history. That is, if* $\mathcal{E}$ *is a prefix of* $\mathcal{E}'$ *then* $P(\mathcal{E}) \to P(\mathcal{E}')$. *Then, we denote by* $\mathrm{until}_P(\mathsf{R})$ *the resource that behaves like* $\mathsf{R}$ *but* halts *the moment* $P(\mathcal{E})$ *becomes true. That is, it no longer triggers any further events and all subsequent (including the one for the query that triggered the condition) answers are the special symbol* $\perp$ .

Getting back to our example, consider the resource $\mathrm{until}_P(\mathsf{SecCh})$ for $P(\mathcal{E}) \coloneqq \mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}} \vee \mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}}$, depicted in Figure 2. Since this resource no longer produces any output once either event occurred, it clearly never leaks the messages to Eve and removes Eve's capability of injecting messages. Hence, the resulting resource closely matches the expected secure channel when ignoring key exposures.

We now define the according relaxation, which maps a system to the set of all systems that behave equivalently up to some event.

**Definition 5.** *Let* $P$ *be a monotone predicate on the global event history, indicating until when the behavior must be the same as the one of the resource* $\mathsf{R}$. *Then, the induced relaxation on a resource* $\mathsf{R}$, *denoted* $\mathsf{R}^{P]}$, *is defined as*

$$\mathsf{R}^{P]} \coloneqq \left\{\mathsf{S} \mid \mathrm{until}_P(\mathsf{R}) = \mathrm{until}_P(\mathsf{S})\right\}$$

*We call such a relaxation generally an* until-relaxation.

As with the $\epsilon$-relaxation, the statements only become reusable and thus truly composable if we understand how the until-relaxation interacts with the other elements of the framework. For this, first observe that equality up to some point is monotone, i.e., if two resources are equivalent up to some point, they are also equivalent up to every earlier point. This furthermore implies that two until-relaxations add up in the natural manner, as follows.

**Theorem 4.** *Let* $\mathsf{R}$ *and* $\mathsf{S}$ *be two resources, and let* $P_1$ *and* $P_2$ *be two monotone predicates. Then, we have*

$$\mathrm{until}_{P_1}(\mathsf{R}) = \mathrm{until}_{P_1}(\mathsf{S}) \implies \mathrm{until}_{P_1 \vee P_2}(\mathsf{R}) = \mathrm{until}_{P_1 \vee P_2}(\mathsf{S}).$$

*In particular, for every specification* $\mathcal{R}$*, we have* $\mathcal{R}^{P_1]} \subseteq \left(\mathcal{R}^{P_1]}\right)^{P_2]} \subseteq \mathcal{R}^{P_1 \vee P_2]}$*.*

*Proof.* The first property follows directly from the definition of the until() projection. In order to prove the second property, let $\mathsf{S} \in \left(\mathcal{R}^{P_1]}\right)^{P_2]}$. Then, there exists $\mathsf{T} \in \mathcal{R}^{P_1]}$ such that $\mathrm{until}_{P_2}(\mathsf{S}) = \mathrm{until}_{P_2}(\mathsf{T})$. Moreover, there exists $\mathsf{R} \in \mathcal{R}$ such that $\mathrm{until}_{P_1}(\mathsf{T}) = \mathrm{until}_{P1}(\mathsf{R})$. By the first property we, thus, obtain $\mathrm{until}_{P_1 \vee P_2}(\mathsf{S}) = \mathrm{until}_{P_1 \vee P_2}(\mathsf{T}) = \mathrm{until}_{P_1 \vee P_2}(\mathsf{R})$, concluding the proof. $\qquad\square$

Furthermore, on a positive note, the relaxation is compatible with both protocol application and parallel composition, as expressed by the following theorem. Those compatibility properties—analogously to Corollary 1—also directly imply sequential and parallel composition properties. For the lack of use, we however omit explicitly stating them.

**Theorem 5.** *The until-relaxation is compatible with protocol attachment, i.e.,* $\pi\left(\mathcal{R}^{P]}\right) \subseteq \left(\pi\mathcal{R}\right)^{P]}$ *and with parallel composition, i.e.,* $\left[\mathcal{R}^{P]}, \mathcal{S}\right] \subseteq \left[\mathcal{R}, \mathcal{S}\right]^{P]}$*.*

*Proof.* A proof is presented in Appendix B.2.

Unfortunately, however, the until-relaxation does not commute directly with the $\epsilon$-relaxation, as expressed by the following theorem.

**Theorem 6.** *There exist specifications* $\mathcal{R}$ *and* $\mathcal{S}$*, a monotone predicate* $P$*, and a function* $\epsilon$ *mapping distinguishers to values in* $[0, 1]$ *such that*

$$\left(\mathcal{R}^{P]}\right)^{\epsilon} \not\subseteq \left(\mathcal{R}^{\epsilon}\right)^{P]} \qquad and \qquad \left(\mathcal{S}^{\epsilon}\right)^{P]} \not\subseteq \left(\mathcal{S}^{P]}\right)^{\epsilon}.$$

*Proof.* The proof can be found in Appendix B.3.

This not only raises the question which order actually corresponds to the intuitive interpretation of such a combination—the set of all systems which behave equally until the condition is triggered assuming the assumption of $\epsilon$ is valid—but also restricts reuse of such statement. That is, if one construction assumes $\mathcal{S}^{P]}$ to obtain $\mathcal{T}$, and another one constructs $\mathcal{S}^{\epsilon}$ instead, then adjusting the former construction to assume $\mathcal{S}^{\epsilon}$ instead is non-trivial. As a consequence, we will introduce a combined relaxation in Section 4.3, resolving both issues.

18

### 4.2 Guarantees From Some Point On

In this section, we now consider the complementing type of guarantees: guarantees that only hold from a certain point on. Formalizing such guarantees in a model where an adaptive environment interacts with the resource is, however, quite delicate. In this work, we thus opt for a rather simple (and restricted) version of it, where we use again a monotone condition on the global event history. We then define the projection that disables access to a system $R$ before that condition is met. Clearly, the condition must rely on "external" events only (the ones not controlled by $R$), i.e., satisfying it must not require accessing the resource itself.

**Definition 6.** *Let $P(\mathcal{E})$ denote a monotone predicate on the global event history. For a resource $R$, let $\mathrm{from}_P(R)$ denote the resource that behaves like $R$, except that it only accepts queries once $P(\mathcal{E})$ is true (and before only returns $\bot$).*

For instance, the resource $\mathrm{from}_{\mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}}}(\mathsf{SecCh})$ only answers queries once the environment triggered the event $\mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}}$. Thus, in contrast to $\mathsf{SecCh}$, this resource always leaks the full message of the adversary, in line with our intuition that it describes the behavior after the key has been exposed. A formal definition of the resource is depicted in Appendix B.1.

Based on this projection, we now introduce the corresponding relaxation.

**Definition 7.** *Let $P(\mathcal{E})$ be a monotone predicate, indicating from which point on the behavior must be the same as the one of the resource $R$. Then, the induced relaxation on a resource $R$, denoted $R^{[P}$, is defined as*

$$R^{[P} := \left\{ S \mid \mathrm{from}_P(R) = \mathrm{from}_P(S) \right\}$$

*We call such a relaxation generally a from-relaxation.*

The way the from-relaxation interacts with the other elements of the theory is analogous to the until-relaxation. First, two from-relaxations add up naturally: if we relax the guarantees offered by a specification to only hold from the moment $P_1$ is satisfied, and then further relax them to only hold once $P_2$ is satisfied, then the guarantees only hold once $P_1 \wedge P_2$ is satisfied.

**Theorem 7.** *Let $R$ and $S$ be two resources, and let $P_1$ and $P_2$ be monotone predicates on the global event history. Then, we have*

$$\mathrm{from}_{P_1}(R) = \mathrm{from}_{P_1}(S) \implies \mathrm{from}_{P_1 \wedge P_2}(R) = \mathrm{from}_{P_1 \wedge P_2}(S).$$

*In particular, for every specification $\mathcal{R}$, we have $\mathcal{R}^{[P_1} \subseteq \left(\mathcal{R}^{[P_1}\right)^{[P_2} \subseteq \mathcal{R}^{[P_1 \wedge P_2}$.*

*Proof.* The proof is analogous to the one of Theorem 4.

Second, the relaxation is compatible with protocol application and parallel composition, which moreover implies that it graciously interacts with the basic construction notion.

**Theorem 8.** *The from-relaxation is compatible with protocol application, i.e.,* $\boldsymbol{\pi}(\mathcal{R}^{[P]}) \subseteq (\boldsymbol{\pi}\mathcal{R})^{[P}$ *and with parallel composition, i.e.,* $[\mathcal{R}^{[P}, \mathcal{S}] \subseteq [\mathcal{R}, \mathcal{S}]^{[P}$.

*Proof.* The proof is analogous to the one of Theorem 5.

Analogously to the until-relaxation, the from-relaxation, however, does not commute with the $\epsilon$-relaxation.

**Theorem 9.** *There exist specifications $\mathcal{R}$ and $\mathcal{S}$, a monotone predicate $P(\mathcal{E})$, and a function $\epsilon$ mapping distinguishers to values in $[0,1]$ such that*

$$\left(\mathcal{R}^{[P]}\right)^\epsilon \not\subseteq \left(\mathcal{R}^\epsilon\right)^{[P} \qquad and \qquad \left(\mathcal{S}^\epsilon\right)^{[P} \not\subseteq \left(\mathcal{S}^{[P]}\right)^\epsilon.$$

*Proof.* The proof is presented in Appendix B.4.

Finally, consider the interaction between the from- and the until-relaxation. While the from-projection and the until-projection commute, i.e.,

$$\mathrm{from}_{P_1}(\mathrm{until}_{P_2}(\mathsf{R})) = \mathrm{until}_{P_2}(\mathrm{from}_{P_1}(\mathsf{R})),$$

it is an interesting open question whether the two respective relaxations actually commute. As a consequence, we introduce a combined from-until relaxation in the next subsection.

### 4.3   The Interval-Wise Relaxation

As we have seen, the $\epsilon$-relaxation commutes neither with the until-relaxation nor the from-relaxation, and its unclear whether the from- and until-relaxations do. This impedes modularity and reusability of the statements. Furthermore, it also deteriorates the intuitive semantics of the statements: if for instance we want to express that a system behaves like a certain ideal up to some point, and under certain computational assumptions, which order of the relaxations is the right one and should be proven? To alleviate those issues, in this section, we introduce two combined relaxations that build on the atomic ones introduced in the previous section. We then show that they both have natural semantics and clean properties.

First, we consider a relaxation that combines the from- and until-relaxation, thereby alleviating the issue that those relaxations might not commute.

**Definition 8.** *Let $P_1(\mathcal{E})$ and $P_2(\mathcal{E})$ be two monotone predicates, indicating from when until when the resource must behave like $\mathsf{R}$. We then define the following relaxation*

$$\mathsf{R}^{[P_1, P_2]} := \left\{\mathsf{S} \mid \mathrm{until}_{P_2}(\mathrm{from}_{P_1}(\mathsf{R})) = \mathrm{until}_{P_2}(\mathrm{from}_{P_1}(\mathsf{S}))\right\}.$$

While this combined relaxation apparently neither corresponds to $\left(\mathsf{R}^{[P_1}\right)^{P_2]}$ nor $\left(\mathsf{R}^{P_2]}\right)^{[P_1}$, it interestingly corresponds to the transitive closure thereof. Taking the transitive closure, moreover, also restores symmetry, i.e., $\mathsf{S} \in \mathsf{R}^{[P_1, P_2]} \Leftrightarrow \mathsf{R} \in \mathsf{S}^{[P_1, P_2]}$, lost by each of the two individual combinations. Overall, this indicates that the combined relaxation best corresponds to the intuition of the "almost-as-good" relation it should intuitively represent.

20

**Theorem 10.** *For any resource* $\mathsf{R}$ *and any monotone predicates* $P_1$ *and* $P_2$*, we have*

$$\mathsf{R}^{[P_1,P_2]} = \bigcup_{n \in \mathbb{N}} \left( \bigcup \left\{ \mathsf{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n} \mid \forall i \le n : \phi_i \in \{P_2\}, [P_1] \right\} \right)$$

$$= \left( \left( \mathsf{R}^{[P_1]} \right)^{P_2} \right)^{[P_1} = \left( \left( \mathsf{R}^{P_2]} \right)^{[P_1]} \right)^{P_2]},$$

*where* $\mathsf{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n}$ *is a shorthand notation for first applying* $\phi_1$*, then* $\phi_2$*, until* $\phi_n$*.*

*Proof.* The proof can be found in Appendix B.5.

We can now leverage this alternative definition to directly derive properties about the combined relaxations based on the proven properties of the two underlying ones. In particular, we can show that two such relaxations add up in the expected manner and are compatible with both protocol application as well as parallel composition.

**Theorem 11.** *For every specification* $\mathcal{R}$*, and all monotone predicates* $P_1$*,* $P_2$*,* $P_1'$*, and* $P_2'$*, we have* $\left( \mathcal{R}^{[P_1,P_2]} \right)^{[P_1',P_2']} \subseteq \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']}$*.*

*Proof.* This follows directly by combining Theorems 4, 7 and 10. □

**Theorem 12.** *The combined relaxation is both compatible with protocol application, i.e.,* $\boldsymbol{\pi}\left( \mathcal{R}^{[P_1,P_2]} \right) \subseteq \left( \boldsymbol{\pi}\mathcal{R} \right)^{[P_1,P_2]}$ *and with parallel composition, i.e.,* $\left[ \mathcal{R}^{[P_1,P_2]}, \mathcal{S} \right] \subseteq \left[ \mathcal{R}, \mathcal{S} \right]^{[P_1,P_2]}$*.*

*Proof.* By Theorem 10 we have that $\mathcal{R}^{[P_1,P_2]} = \left( \left( \mathsf{R}^{[P_1]} \right)^{P_2} \right)^{[P_1}$. Using the compatibility of the from-relaxation and until-relaxation, i.e., Theorems 5 and 8, directly implies the desired properties. □

As we have seen, neither the until- nor the from-relaxation commute with the computational $\epsilon$-relaxation, and the same holds true for the from-until-relaxation as well. As a consequence, neither $(\mathsf{R}^{[P_1,P_2]})^\epsilon$ nor $(\mathsf{R}^\epsilon)^{[P_1,P_2]})$ seems to capture the set of all systems that behave like $\mathcal{R}$ in the interval $[P_1, P_2]$ assuming that the computational problem encoded in $\epsilon$ is hard. In the spirit of the combined from-until relaxation, we solve this issue by introducing a combined relation. Since the $\epsilon$ relaxation is not idempotent, but the epsilons add up, taking the transitive closure, however, does not match the desired relaxation but the following restricted version of transitive closure does.

**Definition 9.** *For two monotone predicates* $P_1$ *and* $P_2$*, and a function* $\epsilon$ *mapping distinguishers to values in* $[0, 1]$*, we define the following relaxation:*

$$\mathsf{R}^{[P_1,P_2]:\epsilon} := \left( \left( \mathsf{R}^{[P_1,P_2]} \right)^\epsilon \right)^{[P_1,P_2]},$$

*and call such a relaxation an* interval-wise relaxation*.*

21

We now prove that the interval-wise relaxation has all the desired properties.

**Theorem 13.** *Let $P_1$ and $P_2$ be two monotone predicates, and let $\epsilon$ be a function mapping distinguishers to values in $[0, 1]$. Then, for any specification $\mathcal{R}$ we have*

$$\left(\mathcal{R}^{[P_1,P_2]:\epsilon}\right)^{[P_1',P_2']:\epsilon'} \subseteq \mathcal{R}^{[P_1\wedge P_1',P_2\vee P_2']:\epsilon_{[P_1\wedge P_1',P_2\vee P_2']}+\epsilon'_{[P_1\wedge P_1',P_2\vee P_2']}},$$

*where $\epsilon_{[P_1\wedge P_1',P_2\vee P_2']}(D) := \epsilon(\mathsf{D}\circ \text{until}_{P_2\vee P_2'}\circ \text{from}_{P_1\wedge P_1'})$, i.e., the performance of the distinguisher interacting with the projected resource, and analogously for $\epsilon'_{[P_1\wedge P_1',P_2\vee P_2']}$.*

*Proof.* The proof can be found in [Appendix B.6](#).

**Theorem 14.** *The interval-wise relaxation is compatible with protocol application, i.e., $\boldsymbol{\pi}\left(\mathcal{R}^{[P_1,P_2]:\epsilon}\right) \subseteq \left(\boldsymbol{\pi}\mathcal{R}\right)^{[P_1,P_2]:\epsilon_{\boldsymbol{\pi}}}$ and with parallel composition, i.e., $\left[\mathcal{R}^{[P_1,P_2]:\epsilon}, \mathcal{S}\right] \subseteq \left[\mathcal{R},\mathcal{S}\right]^{[P_1,P_2]:\epsilon_{\mathcal{S}}}$.*

*Proof.* By definition we have $\mathsf{R}^{[P_1,P_2]:\epsilon} := \left(\left(\mathsf{R}^{[P_1,P_2]}\right)^{\epsilon}\right)^{[P_1,P_2]}$. Using the compatibility of the $\epsilon$-relaxation and the from-until-relaxation, i.e., [Theorems 3](#) and [12](#), directly implies the result.

### 4.4 The Resulting Construction Notion

Based on the interval-wise relaxation, we now introduce our new construction notion. To this end, let $\Omega$ denote a set of tuples $(P_1, P_2, \epsilon, \boldsymbol{\sigma})$, where $P_1$ and $P_2$ are monotone predicates on the global event history, $\epsilon$ is a function mapping distinguishers to values in $[0, 1]$, and $\boldsymbol{\sigma}$ denotes a simulator. We then consider constructions of the following type:

$$\mathcal{R} \xrightarrow{\boldsymbol{\pi}} \bigcap_{(P_1,P_2,\epsilon,\boldsymbol{\sigma})\in\Omega} (\boldsymbol{\sigma}\mathcal{S})^{[P_1,P_2]:\epsilon}.$$

That is, each element in $\Omega$ describes a time-interval in which the elements in $\boldsymbol{\pi}\mathcal{R}$ can be abstracted as elements in $\mathcal{S}$—with respect to the simulator $\boldsymbol{\sigma}$ and error $\epsilon$.

**Application to the running example.** In our example, we want to phrase that the symmetric encryption protocol constructs the secure channel from the authenticated one and the key in the corresponding intervals.

**Proposition 6.** *Let $\boldsymbol{\pi}_{\mathsf{ENC}} = (\pi_{\mathsf{enc}}, \pi_{\mathsf{dec}})$ denote the protocol securing communication using a symmetric encryption scheme. Then, for the resources in [Figure 1](#), there exist (efficient) simulators $\boldsymbol{\sigma_1}$, $\boldsymbol{\sigma_2}$, and $\boldsymbol{\sigma_3}$ such that*

$$\left[\mathsf{EncKey}, \mathsf{AuthCh}\right] \xrightarrow{\boldsymbol{\pi}_{\mathsf{ENC}}} \bigcap_{(P_1,P_2,\epsilon,\boldsymbol{\sigma})\in\Omega} (\boldsymbol{\sigma}\mathsf{SecCh})^{[P_1,P_2]:\epsilon}$$

*for*

$$\Omega := \big\{\big(\mathtt{true}, \mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}} \vee \mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}, \epsilon_{\mathsf{CPA}}, \boldsymbol{\sigma}_1\big), \big(\mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}}, \mathtt{false}, 0, \boldsymbol{\sigma}_2\big),$$
$$\big(\mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}, \mathtt{false}, 0, \boldsymbol{\sigma}_3\big)\big\},$$

*where $\epsilon_{\mathsf{CPA}}$ denotes a simple reduction from distinguishing the secure and authenticated channel (without key leakage) to the IND-CPA game.*

*Proof.* A proof sketch is presented in [Appendix B.7](#).

**Composition.** Finally, we finish the section by stating the composition guarantees of this type of construction statement. It follows directly from the properties proven about the interval-wise relaxation in [Theorems 13](#) and [14](#).

**Theorem 15.** *Let $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$ be arbitrary specifications, let $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$ be arbitrary protocols, and let $\Omega$ and $\Omega'$ be arbitrary interval-wise guarantees. Then, we have*

$$\mathcal{R} \xrightarrow{\boldsymbol{\pi}} \bigcap_{(P_1, P_2, \epsilon, \boldsymbol{\sigma}) \in \Omega} (\boldsymbol{\sigma}\mathcal{S})^{[P_1, P_2]:\epsilon} \quad \wedge \quad \mathcal{S} \xrightarrow{\boldsymbol{\pi}'} \bigcap_{(P_1', P_2', \epsilon', \boldsymbol{\sigma}') \in \Omega'} (\boldsymbol{\sigma}'\mathcal{T})^{[P_1', P_2']:\epsilon'}$$
$$\implies \quad \mathcal{R} \xrightarrow{\boldsymbol{\pi}' \circ \boldsymbol{\pi}} \bigcap_{\substack{(P_1, P_2, \epsilon, \boldsymbol{\sigma}) \in \Omega \\ (P_1', P_2', \epsilon', \boldsymbol{\sigma}') \in \Omega'}} (\boldsymbol{\sigma}\boldsymbol{\sigma}'\mathcal{T})^{[P_1 \wedge P_1', P_2 \vee P_2']:\tilde{\epsilon}},$$

*where $\tilde{\epsilon} := (\epsilon_{\boldsymbol{\pi}'})_{[P_1 \wedge P_1', P_2 \vee P_2']} + (\epsilon'_{\boldsymbol{\sigma}})_{[P_1 \wedge P_1', P_2 \vee P_2']}$. Furthermore, we have*

$$\mathcal{R} \xrightarrow{\boldsymbol{\pi}} \bigcap_{(P_1, P_2, \epsilon, \boldsymbol{\sigma}) \in \Omega} (\boldsymbol{\sigma}\mathcal{S})^{[P_1, P_2]:\epsilon} \implies [\mathcal{R}, \mathcal{T}] \xrightarrow{\boldsymbol{\pi}} \bigcap_{(P_1, P_2, \epsilon, \boldsymbol{\sigma}) \in \Omega} \big(\boldsymbol{\sigma}[\mathcal{S}, \mathcal{T}]\big)^{[P_1, P_2]:\epsilon_{\mathcal{T}}}.$$

*Proof.* The proof is stated in [Appendix B.8](#).

Note that this construction notion subsumes all those previously introduced in this work. In particular, instantiating $P_1 = \mathtt{true}$, $P_2 = \mathtt{false}$, $\epsilon(\mathsf{D}) = 0$, and $\boldsymbol{\sigma} = \mathsf{id}$, i.e., the identity converter, yields $(\mathsf{id}\mathcal{S})^{[\mathtt{true}, \mathtt{false}]:0} = \mathcal{S}$. As a consequence, the above composition theorem also allows to combine constructions according to each of the notions introduced in this work. For instance, in our example, we can compose the construction of $\mathsf{AuthCh}$ from [Proposition 4](#) (according to the standard notion) with the interval-wise construction of $\mathsf{SecCh}$ from [Proposition 6](#).

## 5    Application to Commitments and Coin-Tossing

In this section, we present a composable formalization of (perfectly binding) commitments that can be constructed in the plain model. To this end, we formalize the properties of commitment schemes—correctness, binding, and hiding—each as individual specifications. Thereby, hiding is formalized using the interval-wise guarantees introduced in the previous section. We then apply Blum's coin-tossing protocol on top of it. While, obviously, the resulting specifications are not sufficient to be used as a CRS, we show that it is unbiased.

### 5.1 Information-Theoretically Binding Commitments

While UC commitments [6] provide clean and strong guarantees, unfortunately they intrinsically require setup assumptions such as a common reference string. Nevertheless, for many protocols, regular commitments only satisfying the classical game-based properties seem to suffice. This raises the question: can we formalize a weaker yet composable security notion for (non-interactive) commitments?

In Constructive Cryptography, the security of a commitment scheme is formalized using three different constructions [17], for each set of potentially dishonest parties (ignoring the case of both parties being dishonest). Typically, this is presented as one construction parametrized in the set of honest parties, where the ideal specification consists of a filtered resource. That is, for each party $\mathsf{P}$, a filter $\phi_\mathsf{P}$ is specified that when connected to the resource limits the honest party's capabilities. However, there is no fundamental reason for those three construction statements' specifications to be of some unified type. As a result, we henceforth focus on specifying each property—hiding, binding, and correctness—individually, starting with correctness.

**Definition 10.** *Let $\boldsymbol{\pi}_{\mathsf{com}} = (\pi_{\mathsf{com}}^{\mathsf{A}}, \pi_{\mathsf{com}}^{\mathsf{B}})$ denote a non-interactive commitment protocol where $\mathsf{A}$ commits a value $m \in \mathcal{M}$ towards $\mathsf{B}$. The scheme is said to be (perfectly)* correct *if*

$$\left[\mathsf{Ch}_1^{\mathsf{A}\to\mathsf{B}}, \mathsf{Ch}_2^{\mathsf{A}\to\mathsf{B}}\right] \xrightarrow{\boldsymbol{\pi}_{\mathsf{com}}} \mathsf{Com}_{\mathcal{M}}^{\mathsf{A}\to\mathsf{B}},$$

*where $\mathsf{Com}_{\mathcal{M}}^{\mathsf{A}\to\mathsf{B}}$ denotes the commitment resource defined in Figure 3, and $\mathsf{Ch}_1^{\mathsf{A}\to\mathsf{B}}$ and $\mathsf{Ch}_2^{\mathsf{A}\to\mathsf{B}}$ denote two single-message communications channels[3] from $\mathsf{A}$ to $\mathsf{B}$.*

Now we proceed to formalize the hiding property. On an intuitive level, (computational) hiding of a non-interactive commitment scheme requires that the commitment string must not reveal any information about the committed value to the receiver $\mathsf{B}$, until the commitment is opened. Clearly, we can directly apply our notion from Section 4 and formalize this using an interval-wise relaxation.

**Definition 11.** *Let $\boldsymbol{\pi}_{\mathsf{com}} = (\pi_{\mathsf{com}}^{\mathsf{A}}, \pi_{\mathsf{com}}^{\mathsf{B}})$ denote a non-interactive commitment protocol. Then, the scheme is said to be (computationally)* hiding *if*

$$\left[\mathsf{Ch}_1^{\mathsf{A}\to\mathsf{B}}, \mathsf{Ch}_2^{\mathsf{A}\to\mathsf{B}}\right] \xrightarrow{\pi_{\mathsf{com}}^{\mathsf{A}}} \left(\sigma_{\mathsf{com}}^{\mathsf{B}} \mathsf{Com}_{\mathcal{M}}^{\mathsf{A}\to\mathsf{B}}\right)^{[\texttt{true}, \mathcal{E}^{\mathsf{opened}}]:\epsilon},$$

*for some simulator $\sigma_{\mathsf{com}}^{\mathsf{B}}$ and some computational assumption encoded in $\epsilon$.*

The situation is more challenging with binding. The UC formalization, and analogously $\mathsf{Com}_{\mathcal{M}}^{\mathsf{A}\to\mathsf{B}}$, requires that the adversary inputs the value to which it commits to in the initial phase, in order to formalize that it then cannot be altered anymore. This, however implies that the simulator must be able to extract the value from the commitment string, fundamentally contradicting

---

[3] That is, a channel that allows the sender to input a single message once. For simplicity, assume that the channel has guaranteed immediate delivery, i.e., whenever the sender has input a message, the receiver can fetch it.

**Initialization**
 $v \leftarrow \bot$

**Interface A**
**Input:** $(\mathtt{commit}, x) \in \mathcal{M}$
  **require** $\neg\mathcal{E}^{\mathsf{committed}}$
  $v \leftarrow x$
  $\mathcal{E} \overset{+}{\leftarrow} \mathcal{E}^{\mathsf{committed}}$
  **output** ok

**Input:** $(\mathtt{commitRaw}, c) \in \{0,1\}^*$
  **require** $\neg\mathcal{E}^{\mathsf{committed}}$
  $v \leftarrow f(c)$
  $\mathcal{E} \overset{+}{\leftarrow} \mathcal{E}^{\mathsf{committed}}$
  **output** ok

**Interface A (continued)**
**Input:** $(\mathtt{open}, x) \in \mathcal{M}$
  **require** $\mathcal{E}^{\mathsf{committed}}$ and only called once
  **if** $x = v$ **then**
  $\quad \mathcal{E} \overset{+}{\leftarrow} \mathcal{E}^{\mathsf{opened}}$
  **output** ok

**Interface B**
**Input:** isCommitted
  **output** $\mathcal{E}^{\mathsf{committed}}$
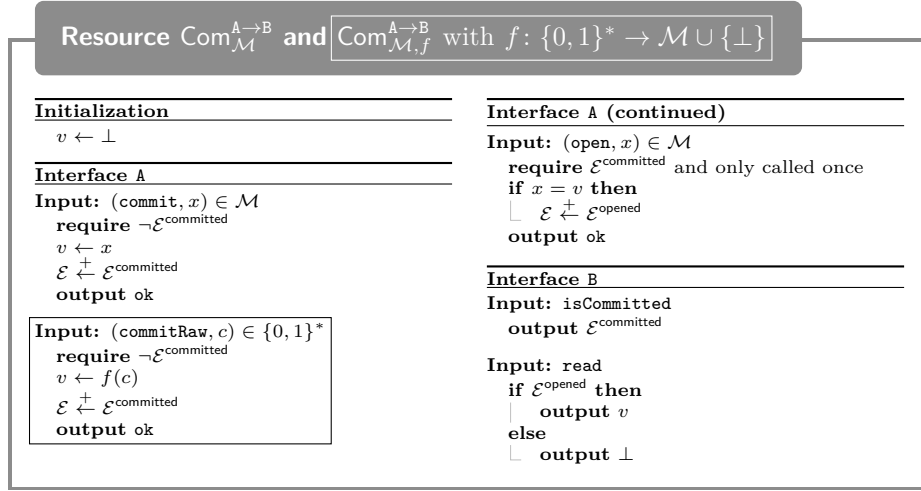
**Input:** read
  **if** $\mathcal{E}^{\mathsf{opened}}$ **then**
  $\quad$ **output** $v$
  **else**
  $\quad$ **output** $\bot$

Fig. 3: The commitment resources for message space $\mathcal{M}$. In the basic version, Alice has to specify the value at the time of commitment, whereas in the unfiltered version she additionally has the ability to commit to $f(c)$.

the hiding property in the plain model. Since such a formalization is just one (albeit convenient) manner to specify that the value is fixed at the end of the commitment phase, we circumvent this impossibility in another manner. To this end, we consider perfect (or information-theoretically secure) commitments only, where the commitment string uniquely determines the committed value. We leverage this considering a resource $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\mathcal{M},f}$, depicted in Figure 3, which allows the dishonest A to input an arbitrary string $x$ in order to commit to the value $v = f(x)$. Here, $f\colon \{0,1\}^* \to \mathcal{M}\cup\{\bot\}$ denotes an arbitrary function that maps the commitment string either to a message $m$, or to $\bot$ indicating that it is malformed.

**Definition 12.** *Let* $\boldsymbol{\pi}_{\mathsf{com}} = (\pi^{\mathtt{A}}_{\mathsf{com}}, \pi^{\mathtt{B}}_{\mathsf{com}})$ *denote a non-interactive commitment protocol where* A *commits a value* $m \in \mathcal{M}$ *towards* B*. Then, the scheme is said to be* perfectly binding *if there exists an efficient simulator* $\sigma^{\mathtt{A}}_{\mathsf{com}}$ *such that*

$$\left[\mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_1, \mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_2\right] \xrightarrow{\pi^{\mathtt{B}}_{\mathsf{com}}} \left\{\sigma^{\mathtt{A}}_{\mathsf{com}}\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\mathcal{M},f} \;\middle|\; f\colon \{0,1\}^* \to \mathcal{M}\cup\{\bot\}\right\},$$

*where* $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\mathcal{M},f}$ *denotes the extended commitment resource defined in Figure 3.*

As a side note, note that the resource $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\mathcal{M}}$ can trivially be expressed as a filtered version of $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\mathcal{M},f}$, where the filter $\phi_{\mathtt{A}}$ removes access to the `commitRaw` oracle. That is, we obviously have $\phi_{\mathtt{A}}\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\mathcal{M},f} = \mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\mathcal{M}}$ for every function $f$.

*Remark 1.* Observe that the function $f$ is not necessary efficiently computable. Actually, for a hiding scheme, $f$ cannot be efficiently computable. This, however,

does not imply that the overall specification has to contain resources that are not efficiently implementable, as clearly the real-world resource is efficient, and yet corresponds to $\sigma^{\mathsf{A}}\mathsf{Com}^{\mathsf{A}\to\mathsf{B}}_{\mathcal{M},f}$ for an inefficient $f$. In other words, the decomposition, containing the inefficient resource, is just one way of describing an overall efficient resource. The specification could, thus, be restricted to consist of only efficient resources, which we did not make explicit here focusing only on the security properties. This is somewhat reminiscent of the solution proposed by Broadnax et al. [4] to deal with inefficient simulators in a manner that retains the expected composition guarantees.

**ElGamal commitments.** We briefly consider a variant of ElGamal commitments as a concrete instantiation of the above formalized notion. Let $\mathbb{G} = \langle g \rangle$ denote a cyclic group of order $n$ with generator $g$.

- To commit to a message $m \in \mathbb{G}$, $((g^a, g^b, m \cdot g^{ab}), (a, b)) \leftarrow \mathrm{Commit}(m)$ for $a, b \in \mathbb{Z}_n$ uniformly at random. That is, the commitment string is $(g^a, g^b, m \cdot g^{ab})$ and the opening value $(a, b)$.
- $\mathrm{Open}((c, A, B), (a, b)) \coloneqq c \cdot g^{-ab}$ if $A = g^a$ and $B = g^b$, and $\perp$ otherwise.

**Proposition 7.** *Let $\boldsymbol{\pi}_{\mathsf{ElG\text{-}com}}$ denote the pair of converters implementing the aforementioned ElGamal commitment scheme (c.f. Appendix C.1 for a formal definition). Then, $\boldsymbol{\pi}_{\mathsf{ElG\text{-}com}}$ satisfies correctness, hiding (under the DDH assumption), and binding according to Definitions 10 to 12, respectively.*

*Proof (Sketch).* It is easy to see that the our correctness condition holds. Furthermore, with the simulator $\sigma^{\mathsf{B}}_{\mathsf{com}}$ outputting a random triple of group elements as commitment string, hiding holds under the DDH assumption, i.e., for $\epsilon$ encoding an appropriate reduction to the DDH problem. Finally, consider the function $f$ that maps $(U, V, W) \in \mathbb{G}^3$ to $W \cdot g^{-\mathrm{DL}_g(U) \cdot \mathrm{DL}_g(V)}$ and all other bit-strings to $\perp$. For this function, it is easy to see that a simulator $\sigma^{\mathsf{A}}_{\mathsf{com}}$ exists such that the construction that formalizes binding holds. See Appendix C.1 for a formal description of the respective simulators. □

### 5.2 Coin-Tossing

In this section, we consider Blum's simple coin-tossing protocol [2]. The protocol assumes to have a commitment resource from Alice to Bob, and a communication channel in the reverse direction, at its disposal. It then proceeds as follows: Alice chooses $X \in \{0, 1\}$ uniformly at random and commits to it. Once Bob is sure that Alice committed, he chooses $Y \in \{0, 1\}$ uniformly at random and sends it over to Alice (in clear). Finally, Alice opens the commitment and both parties output $Z = X \oplus Y$.

Clearly, this protocol does not provide fairness—even when instantiated with a UC-secure commitment. This is due to the fact that both parties can always choose to abort the protocol by not responding, and in particular Alice can do so *after* she has seen the result. When instantiating the commitment with the

resource constructed in the last section, one even obtains a weaker resource. Note that this is inherent for our construction being in the plain model, as otherwise it could be used as the bit of a CRS, contradicting well-known impossibility results.

In a nutshell, the resource obtained by our construction guarantees that the output is not biased, but does not exclude that during the opening phase, one of the parties learns some trapdoor allowing it to distinguish it from a uniformly random value. For example, our formalization would allow the resulting bit to be the first bit of a PRG's output, while leaking the seed during the opening phase. Note that such a coin toss resource is still useful, for instance for lotteries. First, if the resulting bit is just used to determine which party gets some good, then bias-resistance is obviously good enough irrespective of the fact that the parties might be aware that the result is only pseudo-random. Second, in a simple lottery where people's preferences are obvious, fairness can be achieved by declaring the party that caused the abort to have lost.

**The coin-toss resource.** The ideal specification is expressed in terms of the resources $\mathsf{CT}_{\mathcal{M}}^{\mathsf{A,B}}$ and $\mathsf{CT}_{\mathcal{M},f}^{\mathsf{A,B}}$, where the former denotes a restricted version of the latter. The resource $\mathsf{CT}_{\mathcal{M}}^{\mathsf{A,B}}$ initially draws an element $Z \in \mathcal{M}$ uniformly at random. In order for the coin-toss $Z$ to become available to the parties, $\mathsf{A}$ has to initiate it, and $\mathsf{B}$ has to respond afterwards. From this point on, $\mathsf{A}$ can obtain $Z$ and then decide whether the value should also be released to $\mathsf{B}$. In the resource $\mathsf{CT}_{\mathcal{M},f}^{\mathsf{A,B}}$, $\mathsf{A}$ furthermore can query once a leakage $f(c)$, of some potentially inefficient function $f$. A formal definition of the resources can be found in Figure 4.

**The constructions.** First, consider correctness. It is easy to see that the following construction holds, i.e., two honest parties actually get to agree on a uniform random bit.

**Proposition 8.** *Let* $\boldsymbol{\pi}_{\mathsf{CT}} \coloneqq (\pi_{\mathsf{CT}}^{\mathsf{A}}, \pi_{\mathsf{CT}}^{\mathsf{B}})$ *denote the pair of converters implementing Blum's protocol (c.f. Appendix C.2 for a formal definition). Then, we have*

$$\left[\mathsf{Com}_{\{0,1\}}^{\mathsf{A}\to\mathsf{B}}, \mathsf{Ch}^{\mathsf{B}\to\mathsf{A}}\right] \xrightarrow{\boldsymbol{\pi}_{\mathsf{CT}}} \mathsf{CT}_{\{0,1\}}^{\mathsf{A,B}},$$

*and thus*

$$\left[\mathsf{Ch}_1^{\mathsf{A}\to\mathsf{B}}, \mathsf{Ch}_2^{\mathsf{A}\to\mathsf{B}}, \mathsf{Ch}^{\mathsf{B}\to\mathsf{A}}\right] \xrightarrow{\boldsymbol{\pi}_{\mathsf{CT}}\circ\boldsymbol{\pi}_{\mathsf{com}}} \mathsf{CT}_{\{0,1\}}^{\mathsf{A,B}},$$

*for any commitment scheme* $\boldsymbol{\pi}_{\mathsf{com}}$ *satisfying Definition 10 (correctness).*

Second, consider the guarantee for an honest initiator $\mathsf{A}$.

**Proposition 9.** *Let* $\boldsymbol{\pi}_{\mathsf{CT}} \coloneqq (\pi_{\mathsf{CT}}^{\mathsf{A}}, \pi_{\mathsf{CT}}^{\mathsf{B}})$ *denote the pair of converters implementing Blum's protocol. Then, there exists an efficient simulator* $\sigma_{\mathsf{CT}}^{\mathsf{B}}$ *such that*

$$\left[\mathsf{Com}_{\{0,1\}}^{\mathsf{A}\to\mathsf{B}}, \mathsf{Ch}^{\mathsf{B}\to\mathsf{A}}\right] \xrightarrow{\pi_{\mathsf{CT}}^{\mathsf{A}}} \sigma_{\mathsf{CT}}^{\mathsf{B}}\mathsf{CT}_{\{0,1\}}^{\mathsf{A,B}},$$

*and thus, for any commitment scheme* $\boldsymbol{\pi}_{\mathsf{com}}$ *satisfying Definition 11 (hiding), we have*

$$\left[\mathsf{Ch}_1^{\mathsf{A}\to\mathsf{B}}, \mathsf{Ch}_2^{\mathsf{A}\to\mathsf{B}}, \mathsf{Ch}^{\mathsf{B}\to\mathsf{A}}\right] \xrightarrow{\pi_{\mathsf{CT}}^{\mathsf{A}}\circ\pi_{\mathsf{com}}^{\mathsf{A}}} \left(\sigma_{\mathsf{com}}^{\mathsf{B}}\sigma_{\mathsf{CT}}^{\mathsf{B}}\mathsf{CT}_{\{0,1\}}^{\mathsf{A,B}}\right)^{[\mathsf{true},\mathcal{E}^{\mathsf{opened}}]:\tilde{\epsilon}},$$

| **Resource** $\mathsf{CT}^{A,B}_{\mathcal{M}}$ **and** $\boxed{\mathsf{CT}^{A,B}_{\mathcal{M},f} \text{ with } f\colon \{0,1\}^* \to \mathcal{M} \cup \{\bot\}}$ |
|---|

**Initialization**

  $Z \leftarrow\!\!\!\!\shortmid\ \mathcal{M}$
  initiated, responded, released $\leftarrow$ `false`

**Interface A**

**Input:** `init`
  **require** $\neg\mathcal{E}^{\text{committed}}$
  initiated $\leftarrow$ `true`
  $\mathcal{E} \overset{+}{\leftarrow} \mathcal{E}^{\text{committed}}$
  **output** ok

**Input:** `read`
  **require** responded
  **output** $Z$

  | **Input:** $(\text{getLeakage}, c) \in \mathcal{M}$
  |   **require** only called once
  |   **output** $f(c)$

**Interface A (continued)**

**Input:** `release`
  **require** responded and only called once
  $\mathcal{E} \overset{+}{\leftarrow} \mathcal{E}^{\text{opened}}$
  released $\leftarrow$ `true`
  **output** ok

**Interface B**

**Input:** `isInitiated`
  **output** initiated

**Input:** `respond`
  **require** initiated
  responded $\leftarrow$ `true`
  **output** ok

**Input:** `read`
  **require** released
  **output** $Z$

Fig. 4: The coin-toss resources for coin space $\mathcal{M}$. In the unfiltered version, Alice additionally has the capability to once obtain a leakage to $f(c)$, where $f$ is a parameter of the resource. Note that neither version provides fairness, as Alice can always chooses to not release the value after having seen it.

*with* $\tilde{\epsilon} := \left(\epsilon_{\sigma^B_{\mathsf{CT}}}\right)_{[\texttt{true}, \mathcal{E}^{\text{opened}}]}$.

*Proof.* Recall that $\mathsf{Com}^{A \to B}_{\{0,1\}}$ only reveals the value $X$ to Bob after he sent his value $Y$. Hence, $X$ and $Y$ are independent and with $X$ chosen uniform at random by Alice, implying that $Z = X \oplus Y$ is a uniform random value. Hence, using the simple simulator $\sigma^B_{\mathsf{CT}}$ that simulates the output of the commitment resource as $X := Z \oplus Y$ (see Appendix C.2 for a formal definition), it is easy to see that the construction actually achieves the coin-toss resource perfectly. $\qquad\square$

Note that this implies that the output $Z$ that Alice obtains looks indistinguishable from a uniform random value until the value is released for the dishonest party. Hence, while it is not guaranteed that the dishonest party does not learn some trapdoor afterwards, the value $Z$ is at least unbiased.

    Finally, consider the security guarantees for an honest party B against a potentially dishonest party A. To this end, we turn to the unfiltered resources $\mathsf{Com}^{A \to B}_{\{0,1\},f}$ and $\mathsf{CT}^{A,B}_{\{0,1\},f}$, where the latter once allows Alice to obtain $f(c)$ for a $c$ of her choice.

**Proposition 10.** *Let* $\boldsymbol{\pi}_{\mathsf{CT}} := (\pi^A_{\mathsf{CT}}, \pi^B_{\mathsf{CT}})$ *denote the pair of converters implementing Blum's protocol. Then, there exists an efficient simulator* $\sigma^A_{\mathsf{CT}}$ *such*

*that*

$$\left\{ \left[ \mathsf{Com}^{\mathsf{A}\to\mathsf{B}}_{\{0,1\},f}, \mathsf{Ch}^{\mathsf{B}\to\mathsf{A}} \right] \,\middle|\, f\colon \{0,1\}^* \to \{0,1,\bot\} \right\}$$

$$\xrightarrow{\pi^{\mathsf{B}}_{\mathsf{CT}}} \left\{ \sigma^{\mathsf{A}}_{\mathsf{CT}} \mathsf{CT}^{\mathsf{A},\mathsf{B}}_{\{0,1\},f} \,\middle|\, f\colon \{0,1\}^* \to \{0,1,\bot\} \right\},$$

*and thus, for any commitment scheme* $\boldsymbol{\pi}_{\mathsf{com}}$ *satisfying Definition 12 (binding), we have*

$$\left[ \mathsf{Ch}^{\mathsf{A}\to\mathsf{B}}_1, \mathsf{Ch}^{\mathsf{A}\to\mathsf{B}}_2, \mathsf{Ch}^{\mathsf{B}\to\mathsf{A}} \right] \xrightarrow{\pi^{\mathsf{B}}_{\mathsf{CT}}\circ\pi^{\mathsf{B}}_{\mathsf{com}}} \left\{ \sigma^{\mathsf{A}}_{\mathsf{com}} \sigma^{\mathsf{A}}_{\mathsf{CT}} \mathsf{CT}^{\mathsf{A},\mathsf{B}}_{\{0,1\},f} \,\middle|\, f\colon \{0,1\}^* \to \{0,1,\bot\} \right\}.$$

*Proof.* Consider the real-world system resulting from attaching Bob's converter only, for some function $f$. Interacting with this resource, the environment can input a commitment string $C$ at Alice's interface, then see Bob's bit $Y$ at Alice's channel interface, and finally see the resulting bit $Z = f(C) \oplus Y$ as the output of Bob's converter. In the following, consider the ideal-world system with the same function $f$ as in the real world. It is now easy to see that a simulator can easily replicate the real-world behavior by getting $Z$ from the resource, querying the leakage-oracle on $C$ getting $f(C)$, and then setting $Y = Z \oplus f(C)$. A formal definition of the simulator $\sigma^{\mathsf{A}}_{\mathsf{com}}$ can be found in Appendix C.2. □

As a final note, observe that formalizing Bob's security guarantees for the commitment resource in terms of an interval-wise relaxation, rather than introducing the unfiltered resource $\mathsf{CT}^{\mathsf{A},\mathsf{B}}_{\{0,1\},f}$, would not work. This is due to the fact that in the real world $Y$ (requiring the additional capabilities to simulate) is output at Alice's interface before Bob sees $Z$. Hence, simulating only until Alice sends $Y$ would not give any guarantees on Bob's output. In summary, this demonstrates Constructive Cryptography's advantage of being able to consider different types of statements within one (meta-)framework.

## 6 Revisiting Composable Identity-Based Encryption

In this section, we reexamine a result by Hofheinz, Matt, and Maurer [10] implying that IND-ID-CPA security is not the right notion for identity-based encryption, unmasking this claim as an unnecessary framework artifact.

### 6.1 Background and Motivation

Identity-based encryption (IBE) is a generalization of public-key encryption that allows to encrypt messages using a master public key and the identity of the receiver, e.g., the e-mail address. This stands in contrast to a regular public-key encryption scheme, where the encryption needs the receiver's public key, suggesting IBE as a solution to the key-distribution problem.

An IBE scheme $\mathsf{IBE} \coloneqq (\mathsf{Gen}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$ consists of four algorithms. The key generation algorithm $(mpk, msk) \leftarrow \mathsf{Gen}(1^\lambda)$ outputs a master public and a master secret key (given the security parameter as input). The extraction

algorithm $sk_{id} \leftarrow \mathsf{Ext}(msk, id)$ outputs a user secret key given the master secret key and the user's identity. Encryption $c \leftarrow \mathsf{Enc}(mpk, id, m)$ outputs a ciphertext, and $m' \leftarrow \mathsf{Dec}(sk_{id}, id, c)$ the corresponding plain-text. For correctness, it is required that for all $(mpk, msk) \leftarrow \mathsf{Gen}()$, all identities $id$, all messages $m$, and all $sk_{id} \leftarrow \mathsf{Ext}(msk, id)$, we always have $\mathsf{Dec}(sk_{id}, id, \mathsf{Enc}(mpk, id, m)) = m$. Security, on the other hand is classically formalized via game-based definitions. For security against passive attacks, the standard notion is ind-id-cpa, as depicted in Figure 5. Weaker notions have been proposed as well, such as a version ind-sid-cpa, depicted in Figure 5 as well, where the adversary has to choose the identity under attack without knowing the master public key.

| **Game** $\mathsf{Exp}_{\mathsf{IBE},\mathcal{A}}^{\mathsf{ind\text{-}id\text{-}cpa}}$ | **Game** $\mathsf{Exp}_{\mathsf{IBE},\mathcal{A}}^{\mathsf{ind\text{-}sid\text{-}cpa}}$ | **Game** $\mathsf{Exp}_{\mathsf{IBE},\mathcal{A}}^{\mathsf{ind\text{-}id1\text{-}cpa}}$ |
|---|---|---|
| $b \leftarrow \{0,1\}$ | $b \leftarrow \{0,1\}$ | $b \leftarrow \{0,1\}$ |
| $(mpk, msk) \leftarrow \mathsf{Gen}()$ | $(mpk, msk) \leftarrow \mathsf{Gen}()$ | $(mpk, msk) \leftarrow \mathsf{Gen}()$ |
| $(st, id, m_0, m_1)$ | $(st, id) \leftarrow \mathcal{A}()$ | $st \leftarrow \mathcal{A}^{\mathsf{Ext}(msk,\cdot)}()$ |
| $\quad \leftarrow \mathcal{A}^{\mathsf{Ext}(msk,\cdot)}(mpk)$ | $(st', m_0, m_1)$ | $(st', id, m_0, m_1)$ |
| $c \leftarrow \mathsf{Enc}(mpk, id, m_b)$ | $\quad \leftarrow \mathcal{A}^{\mathsf{Ext}(msk,\cdot)}(st, mpk)$ | $\quad \leftarrow \mathcal{A}(st, mpk)$ |
| $b' \leftarrow \mathcal{A}^{\mathsf{Ext}(msk,\cdot)}(st, c)$ | $c \leftarrow \mathsf{Enc}(mpk, id, m_b)$ | $c \leftarrow \mathsf{Enc}(mpk, id, m_b)$ |
| **return** $b = b'$ | $b' \leftarrow \mathcal{A}^{\mathsf{Ext}(msk,\cdot)}(st', c)$ | $b' \leftarrow \mathcal{A}^{\mathsf{Ext}(msk,\cdot)}(st', c)$ |
| | **return** $b = b'$ | **return** $b = b'$ |
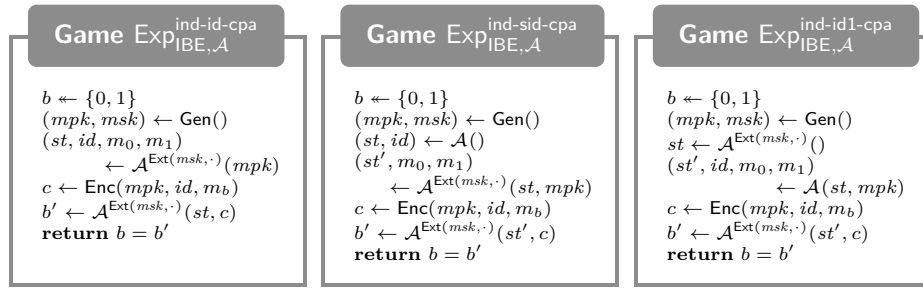
Fig. 5: Three game-based security definition of identity-based encryption against passive attacks. The standard security definition (left) has been put forward by Boneh and Franklin [3]. Later, the weaker static security notion (middle) has been introduced in [7] by Canetti, Halevi, and Katz. Finally, Hofheinz, Matt, and Maurer proposed a version that loosely corresponds to the standard definition under "lunchtime attack" (right). Note that in all of them, $\mathcal{A}$ is not allowed to query the $\mathsf{Ext}$ oracle on the identity it output.

Hofheinz, Matt, and Maurer [10] investigated the composable security of IBE in Constructive Cryptography from an application centric point of view. To this end, they considered non-interactive communication, the apparent standard application of IBE. Assume that there is a trusted party that stores the master secret key and handles registration, i.e, hands out the user secret keys to the correct users. A set of users, each knowing the master public key and his user secret key(s), can now confidentially send messages to each other by encrypting it under the receiver's identity. The obvious security goal is that only the legitimate receiver can read the message. In turned out, however, that the standard ind-id-cpa security does not imply such a construction in the standard simulation-based construction notion—even when considering *static corruptions* only—due to the commitment problem. They, however, managed to show that such a scheme suffices in a weaker setting where it is guaranteed that all identities are registered before the first ciphertext is ever sent. Furthermore, the authors also introduced a new weaker security notion ind-id1-cpa, that essentially considers "lunchtime attack"

and proved that it suffices for the same construction. They hence concluded, that the standard notion is at the same time both too strong (to achieve the weaker construction) and too weak (to achieve the desired construction).

Since the weaker construction is not very realistic, e.g., in a company it is natural that new employees join long after the first ciphertext has been sent, the question about the right security definition remained open. In the remainder of this section, we devise a natural composable formalization based on interval-wise guarantees whose security exactly corresponds to the standard ind-id-cpa notion—resolving the issue.

## 6.2 The Real and Ideal Worlds

On a high level, Hofheinz et al. considered non-interactive secure communication in a setting with one honest sender $A$, n potentially dishonest receivers $B_i$, and one honest party $C$ deriving and distributing the user secret keys. We here consider the same setting with essentially the same resources as in the original work.

In the real world, we assume that the sender $A$ has a broadcast channel $\mathsf{BCAST}_n$ available, through which the ciphertexts are sent. For simplicity, we will assume guaranteed delivery throughout the rest of the example. Furthermore, we assume the existence of an authenticated channel $\mathsf{AUTH}_{\mathcal{P}}^{C \to A}$ from $C$ to $A$ to transmit the master public key Alice needs for encryption, and $n$ secure channels $\mathsf{SEC}_{\mathcal{P}}^{C \to B_i}$ from $C$ to $B_i$ to transmit the user secret keys. A formal description of the corresponding resources can be found in Figure 6. Recall that we consider static corruptions here. Hence, the set of corrupted parties $\mathcal{P}$ appears as an explicit parameter of the resources.

The protocol securing the communication works as expected: whenever Alice wants to send a message to a certain identity, she encrypts it under the given *id* and broadcasts the ciphertext together with the identity. Each honest receiver then checks whether he has the corresponding decryption key, i.e., has been registered for this identity, and either decrypts the message or discards it. Finally, Charlie's protocol not only sends the master public key to Alice, but also allows to register identities for each receiver. Note that each identity can in principle be registered to many interfaces, i.e, many parties can posses the same user secret key. Furthermore, the assignment is not fixed but chosen by the environment, modeling an arbitrary or even adversarially chosen assignment. For completeness, a formal description of the corresponding converters $\pi_{\mathsf{Enc},t}^{A}$, $\pi_{\mathsf{Dec}}^{B_i}$, and $\pi_{\mathsf{Ext}}^{C}$ is given in Figure 7.

Finally, consider the ideal resource $\mathsf{DCC}_{n,t,\mathcal{P}}^{\mathcal{ID}}$, which stands for *delivery controlled channel*. The name derives from the fact that the encryption can be seen as a mechanism with which the sender can specify to whom the message should be delivered. Hence, the resource acts like the broadcast channel, except that each message is only received by parties registered for the specified identity. Other dishonest parties may learn the message length, while honest parties ignore messages not intended for them. Furthermore, dishonest party might also learn which receiver is registered for identities (as the secure channel leaks the length of the user keys, which might depend on the identity). A formal definition of the

**Resource AUTH$_{\mathcal{P}}^{\texttt{C}\to\texttt{A}}$**

**Initialization**
$M \leftarrow$ empty array
$j \leftarrow 0$

**Interface C**
**Input:** $(\texttt{send}, m) \in \mathcal{M}$
$j \leftarrow j + 1$
$M[j] \leftarrow m$
**output** ok

**Interface A**
**Input:** $(\texttt{read}, k) \in \mathbb{N}$
**require** $k \leq j$
**output** $M[k]$

**Interface B$_i$, for $i \in \mathcal{P}$**
**Input:** $(\texttt{read}, k) \in \mathbb{N}$
**require** $k \leq j$
**output** $M[k]$

---

**Resource SEC$_{\mathcal{P}}^{\texttt{C}\to\texttt{B}_r}$**

**Initialization**
$M \leftarrow$ empty array
$j \leftarrow 0$

**Interface C**
**Input:** $(\texttt{send}, m) \in \mathcal{M}$
$j \leftarrow j + 1$
$M[j] \leftarrow m$
**output** ok

**Interface B$_r$**
**Input:** $(\texttt{read}, k) \in \mathbb{N}$
**require** $k \leq j$
**output** $M[k]$

**Interface B$_i$ for $i \in \mathcal{P} \setminus \{r\}$**
**Input:** $(\texttt{read}, k) \in \mathbb{N}$
**require** $k \leq j$
**output** $|M[k]|$

---

**Resource BCAST$_n$**

**Initialization**
$M \leftarrow$ empty array
$j \leftarrow 0$

**Interface A**
**Input:** $(\texttt{send}, m) \in \mathcal{M}$
$j \leftarrow j + 1$
$M[j] \leftarrow m$
**output** ok

**Interface B$_i$, for $i \in \{1, \ldots, n\}$**
**Input:** $(\texttt{read}, k) \in \mathbb{N}$
**require** $k \leq j$
$\mathcal{E} \xleftarrow{+} \mathcal{E}^{\texttt{read}(i,k)}$
**output** $M[k]$

Fig. 6: Description of the assumed resources. The set $\mathcal{P} \subseteq \{1, \ldots, n\}$ thereby specifies the set of statically corrupted receivers $\texttt{B}_i$, while the sender $\texttt{A}$ and registrar $\texttt{C}$ are assumed to be honest.

**Converter $\pi_{\mathsf{Enc},t}^{\mathtt{A}}$**

**Initialization**
$mpk \leftarrow \perp$
$j \leftarrow 0$

**Emulating Interface $\mathtt{A}$ of $\mathrm{DCC}_{n,t,\mathcal{P}}^{\mathcal{ID}}$**

**Input:** $(\mathbf{send}, id, m) \in \mathcal{ID} \times \mathcal{M}$
  **require** $j < t$
  **if** $mpk = \perp$ **then**
    └ **call** $mpk \leftarrow (\mathbf{read})$
       at interface $\mathsf{A}$ of $\mathsf{AUTH}_{\mathcal{P}}^{\mathtt{C} \to \mathtt{A}}$
  **if** $mpk \neq \perp$ **then**
    $c \leftarrow \mathsf{Enc}(mpk, id, m)$
    **call** $(\mathbf{send}, (id, c))$
       at interface $\mathsf{A}$ of $\mathsf{BCAST}_n$
    ┆ $j \leftarrow j + 1$
    ┆ $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{sent}(j, id)}$
    **return** ok
  **else**
  └ **return** $\perp$

**Converter $\pi_{\mathsf{Dec}}^{\mathtt{B}_i}$**

**Initialization**
$SK \leftarrow$ empty array
$j \leftarrow 1$

**Emulating Interface $\mathtt{B}_i$ of $\mathrm{DCC}_{n,t,\mathcal{P}}^{\mathcal{ID}}$**

**Input:** $(\mathbf{read}, id, k) \in \mathcal{ID} \times \mathbb{N}$
  // fetch all pending user keys
  **repeat**
    **call** $(id', sk) \leftarrow (\mathbf{read}, j)$
       at interface $\mathsf{B}_i$ of $\mathsf{SEC}_{\mathcal{P}}^{\mathtt{C} \to \mathtt{B}_i}$
    **if** $(id', sk) \neq \perp$ **then**
      └ $SK[id'] \leftarrow sk$
        $j \leftarrow j + 1$
  **until** $(id', sk) = \perp$

  // fetch message and decrypt if possible
  **call** $(id, m) \leftarrow (\mathbf{read}, k)$
     at interface $\mathsf{B}_i$ of $\mathsf{BCAST}_n$
  **if** $(id, m) \neq \perp \wedge SK[id] \neq \perp$ **then**
    **return** $\mathsf{Dec}(SK[id], id, c)$
  **else**
  └ **return** $\perp$

**Converter $\pi_{\mathsf{Ext}}^{\mathtt{C}}$**

**Initialization**
$mpk, msk \leftarrow \perp$

**Emulating Interface $\mathtt{C}$ of $\mathrm{DCC}_{n,t,\mathcal{P}}^{\mathcal{ID}}$**

**Input:** init
  **require** only called once
  $(mpk, msk) \leftarrow \mathsf{Gen}()$
  **call** $(\mathbf{send}, msk)$
     at interface $\mathsf{C}$ of $\mathsf{AUTH}_{\mathcal{P}}^{\mathtt{C} \to \mathtt{A}}$
  **return** ok

**Emulating Interface $\mathtt{C}$ of $\mathrm{DCC}_{n,t,\mathcal{P}}^{\mathcal{ID}}$**

**Input:** $(\mathbf{register}, id, i) \in \mathcal{ID} \times \{1, \ldots, n\}$
  **require** $msk \neq \perp$
  $sk \leftarrow \mathsf{Ext}(msk, id)$
  **call** $(\mathbf{send}, (id, sk))$
     at interface $\mathsf{C}$ of $\mathsf{SEC}_{\mathcal{P}}^{\mathtt{C} \to \mathtt{B}_i}$
  ┆ $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{registred}(id, i)}$
  **return** ok

Fig. 7: A formal description of the converters using IBE to achieve confidentiality.

**Resource** $\mathsf{DCC}^{\mathcal{ID}}_{n,t,\mathcal{P}}$

**Initialization**
  $M \leftarrow$ empty array, $j \leftarrow 0$, initiated $\leftarrow$ false
  **for** $i \in \{1, \ldots, n\}$ **do**
    $n_i \leftarrow 0$, $I_i \leftarrow$ empty array

**Interface A**
**Input:** $(\mathtt{send}, id, m) \in \mathcal{ID} \times \mathcal{M}$
  **require** initiated $\wedge\ j < t$
  $j \leftarrow j + 1$, $M[j] \leftarrow (id, m)$
  $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{sent}(j,id)}$
  **output** ok

**Interface C**
**Input:** init
  **require** only called once
  initiated $\leftarrow$ true

**Input:** $(\mathtt{register}, id, i) \in \mathcal{ID} \times \{1, \ldots, n\}$
  **require** initiated
  $n_i \leftarrow n_i + 1$
  $I_i[n_i] \leftarrow id$
  $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{registred}(id,i)}$
  **output** ok

**Interface** $\mathtt{B}_i$ **for** $i \in \{1, \ldots, n\}$
**Input:** $(\mathtt{read}, id, k) \in \mathcal{ID} \times \mathbb{N}$
  **require** $k \leq j$
  $(id, m) \leftarrow M[k]$
  $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{read}(k,i)}$

  **if** $\exists \ell : I_i[\ell] = id \vee \mathcal{E}^{\mathsf{insec}(k)}$ **then**
    **output** $(id, m)$
  **else if** $i \in \mathcal{P}$ **then**
    **output** $(id, |m|)$
  **else**
    **output** $\perp$

**Interface** $\mathtt{B}_i$ **for** $i \in \mathcal{P}$ **(additional)**
**Input:** $(\mathtt{identities}, k) \in \{1, \ldots, n\}$
  **output** $I_k$

Fig. 8: The delivery controlled channel with an honest sender $\mathtt{A}$, an honest registrar $\mathtt{C}$, and $n$ receivers $\mathtt{B}_i$, with $\mathcal{P} \subseteq \{1, \ldots, n\}$ denoting the statically corrupted ones.

resource is depicted in Figure 8. Note that compared to [10], we will use global simulators rather than local ones, what allowed us to simplify the resource a bit.

### 6.3 The Composable Statement

One of the main results in [10] has been showing that, due to the commitment problem, an IBE scheme does not construct the delivery controlled channel even under static corruptions.

**Proposition 11 (Theorem 5.1 in [10]).** *Let* $\mathcal{P} \subseteq \{1, \ldots, n\}$ *denote the set of statically corrupted receivers. For every efficient simulator* $\boldsymbol{\sigma}^{\mathcal{P}}_{\mathsf{IBE}}$*, we have*

$$\left[ \mathsf{BCAST}_n, \mathsf{AUTH}^{\mathtt{C} \to \mathtt{A}}_{\mathcal{P}}, \mathsf{SEC}^{\mathtt{C} \to \mathtt{B}_1}_{\mathcal{P}}, \ldots, \mathsf{SEC}^{\mathtt{C} \to \mathtt{B}_n}_{\mathcal{P}} \right] \xrightarrow{\ \boldsymbol{\pi}^{n,t,\mathcal{P}}_{\mathsf{IBE}}\ } \left( \boldsymbol{\sigma}^{\mathcal{P}}_{\mathsf{IBE}} \mathsf{DCC}^{\mathcal{ID}}_{n,t,\mathcal{P}} \right)^{\epsilon_{\mathsf{ind\text{-}id\text{-}cpa}}},$$

*where* $\epsilon_{\mathsf{ind\text{-}id\text{-}cpa}}$ *denotes a reduction to the* ind-id-cpa *game and*

$$\boldsymbol{\pi}^{n,t,\mathcal{P}}_{\mathsf{IBE}} := \left\{ \pi^{\mathtt{A}}_{\mathsf{Enc},t}, \pi^{\mathtt{C}}_{\mathsf{Ext}}, \right\} \cup \bigcup_{i \in \{1, \ldots, n\} \setminus \mathcal{P}} \pi^{\mathtt{B}_i}_{\mathsf{Dec}}$$

*denotes the protocol where all honest parties apply their converter.*

In the end, one is however not interested in such a strong guarantee. Rather, the above construction was just intended as a proxy to formalize confidentiality of the messages. Using our approach, hence immediately yields a better statement based on interval-wise guarantees. More concretely, we consider each message individually and guarantee its confidentiality *until it trivially leaks* because a dishonest party is registered for the corresponding receiver's identity.

In the following, let $\mathcal{E}^{\mathsf{sent}(\mathsf{j},id)}$ indicate that the $j$-th message has been encrypted for identity $id$, $\mathcal{E}^{\mathsf{registred}(id,\mathsf{i})}$ indicate that the $i$-th receiver has been registered for identity $id$, and $\mathcal{E}^{\mathsf{read}(\mathsf{j},\mathsf{i})}$ that the $i$-th receiver read the $j$-th message. Moreover, let $\mathcal{E}^{\mathsf{insec}(\mathsf{j})}$ indicate that the $j$-th message is not confidential. See the dashed boxed in Figure 8 for a formal definition of when those events get triggered in the ideal world. Note that in the real world the former two events get triggered in the converters $\pi_{\mathsf{Enc},t}^{\mathsf{A}}$ and $\pi_{\mathsf{Ext}}^{\mathsf{C}}$, respectively (c.f. Figure 7), whereas the third one gets triggered in the underlying broadcast channel $\mathsf{BCAST}_n$ (c.f. Figure 6). This is due to the fact that Alice and Charlie are assumed to always apply their converter, while the dishonest receivers can access the broadcast channel directly.

**Theorem 16.** *Let $t \in \mathbb{N}$ denote an upper bound on the number of messages, let $\mathcal{P} \subseteq \{1, \ldots, n\}$ denote an arbitrary set of statically corrupted receivers, and let $\boldsymbol{\pi}_{\mathsf{IBE}}^{n,t,\mathcal{P}}$ denote the protocol where all honest parties apply their converter as in Proposition 11. For each $\mathcal{P}$, there exists a sequence of efficient simulators $\boldsymbol{\sigma}_{\mathsf{IBE}}^{\mathcal{P},j}$ and a reduction $\epsilon_{\mathsf{ind\text{-}id\text{-}cpa}}$ to the* $\mathsf{ind\text{-}id\text{-}cpa}$ *game, such that*

$$\left[ \mathsf{BCAST}_n, \mathsf{AUTH}_{\mathcal{P}}^{\mathsf{C}\to\mathsf{A}}, \mathsf{SEC}_{\mathcal{P}}^{\mathsf{C}\to\mathsf{B}_1}, \ldots, \mathsf{SEC}_{\mathcal{P}}^{\mathsf{C}\to\mathsf{B}_n} \right]$$

$$\xrightarrow{\boldsymbol{\pi}_{\mathsf{IBE}}^{n,t,\mathcal{P}}} \bigcap_{j \in \{1,\ldots,t\}} \left( \boldsymbol{\sigma}_{\mathsf{IBE}}^{\mathcal{P},j} \mathsf{DCC}_{n,t,\mathcal{P}}^{\mathcal{ID}} \right)^{[P_{\mathsf{only}(\mathsf{j})}, P_{\mathsf{leaked}(\mathsf{j})}]:\epsilon_{\mathsf{ind\text{-}id\text{-}cpa}}}.$$

*where*

$$P_{\mathsf{leaked}(\mathsf{j})}(\mathcal{E}) \coloneqq \exists i \in \mathcal{P} : \mathcal{E}^{\mathsf{sent}(\mathsf{j},id)} \wedge \mathcal{E}^{\mathsf{registred}(id,\mathsf{i})} \wedge \mathcal{E}^{\mathsf{read}(\mathsf{j},\mathsf{i})}$$

*formalizes the event that the $j$-th message inherently leaked, and*

$$P_{\mathsf{only}(\mathsf{j})}(\mathcal{E}) \coloneqq \bigwedge_{\ell \in \{1,\ldots,t\} \setminus \{j\}} \mathcal{E}^{\mathsf{insec}(\ell)}$$

*formalizes that we do not consider confidentiality of the other messages.*

*Proof (Sketch).* The simulator initially generates a master secret-key and master public-key pair. Observe that for all messages except the $j$-th one, the ideal-world resource outputs the actual message (together with the corresponding identity). Hence, the simulator can simply encrypt it himself under the correct identity. For the $j$-th message, observe that the simulator only has to work if none of the dishonest receivers obtained the corresponding user key, and this key will then never get revealed. Hence, the simulator can encrypt the zero-string of the appropriate length. See Figure 9 for a formal definition of the simulator.

Observe that the reduction from distinguishing the real- and idea-world to the $\mathsf{ind\text{-}id\text{-}cpa}$ game is straight-forward. For all but the $j$-th message, the reduction

**Initialization**
$(mpk, msk) \leftarrow \mathsf{Gen}()$

**Emulating Int. $\mathsf{B}_i$ of $\mathsf{BCAST}_n$ for $i \in \mathcal{P}$**
**Input:** $(\mathsf{read}, k) \in \mathbb{N}$
  **if** $M[k] = \bot$ **then**
    **if** $k \neq j$ **then**
      **call** $(id, m) \leftarrow (\mathsf{read}, k)$
          at interface $\mathsf{B}_i$ of $\mathsf{BCAST}_n$
      **if** $(id, m) \neq \bot$ **then**
        $M[k] \leftarrow (id, \mathsf{Enc}(mpk, id, m))$
    **else**
      **call** $(id, len) \leftarrow (\mathsf{read}, k)$
          at interface $\mathsf{B}_i$ of $\mathsf{BCAST}_n$
      **if** $(id, len) \neq \bot$ **then**
        $M[k] \leftarrow (id, \mathsf{Enc}(mpk, id, 0^{len}))$
  **else**
    $\mathcal{E} \overset{+}{\leftarrow} \mathcal{E}^{\mathsf{read}(k,i)}$
  **return** $M[k]$

**Emulating Int. $\mathsf{B}_i$ of $\mathsf{AUTH}_{\mathcal{P}}^{\mathsf{C} \to \mathsf{A}}$ for $i \in \mathcal{P}$**
**Input:** $(\mathsf{read}, k) \in \mathbb{N}$
  **if** $k = 1 \wedge \exists id : \mathcal{E}^{\mathsf{sent}(1, id)}$ **then**
    **return** $mpk$
  **else**
    **return** $\bot$

**Emulating Int. $\mathsf{B}_i$ of $\mathsf{SEC}_{\mathcal{P}}^{\mathsf{C} \to \mathsf{B}} \ell$ for $i \in \mathcal{P}$**
**Input:** $(\mathsf{read}, k) \in \mathbb{N}$
  **call** $I_\ell \leftarrow (\mathtt{identities}, \ell)$
      at interface $\mathsf{B}_i$ of $\mathsf{DCC}_{n,t,\mathcal{P}}^{\mathcal{ID}}$
  **if** $I_\ell[k] = \bot$ **then**
    **return** $\bot$
  **else**
    $sk \leftarrow \mathsf{Ext}(msk, I_\ell[k])$
    **if** $\ell = i$ **then return** $sk$
    **else return** $|sk|$

Fig. 9: The simulator from Theorem 16.

can simply query the game for the corresponding user key, and the challenge can be chosen as $(m_j, 0^{|m_j|})$. If $b = 0$, this corresponds to the real-world behavior, and if $b = 1$ to the ideal-world behavior. $\qquad\square$

*Remark 2 (On interpreting conjunction specifications).* Observe that phrasing our statement for a bounded number of messages is without loss of generality in any actual application. We, however, have chosen to do so for good reasons. Assume for the moment that $\epsilon$ is simply a constant in $[0, 1]$. If we have a specification of the form $\bigcap_{i \in \mathcal{J}} \mathcal{S}_i^\epsilon$, then this can be read as the guarantee given by $S_i$ holding with probability $1 - \epsilon$. In the end, we are however interested to know the probability such that all guarantees hold. In general, the best we can do is to apply the union bound, implying that the error is bounded by $\epsilon |\mathcal{J}|$. As a result, it is important that the number of conjunctions is small for the statement to be meaningful. Especially, while we could prove an analogous statement with intervals terminated by the leakage of each user key (rather than the messages), this would only make sense if the identity-space is small, in which case ind-sid-cpa security would suffice.

## 7 Conclusion and Future Work

We have demonstrated that considering new types of resource specifications can lead to security notions that are composable, yet do not suffer the artificial impossibilities exhibited by the classical simulation-based definitions. We have introduced a type of specification that formalizes guarantees that hold in a certain time-interval (between two events), which has clean semantics, comes with a

natural syntactical composition theorem, and integrates well with the existing Constructive Cryptography framework. The usefulness of this type of security guarantees has been shown by revisiting several examples where a composable statement previously either needed additional trusted setup or less efficient protocols compared to the corresponding game-based notions.

While our novel type of relaxation does not resolve every issue of composable security, we ultimately believe that all (meaningful) security statements can be expressed as an as assumed specification being contained in an ideal one. Further work is, hence, needed to identify additional types of specifications that allow to express more properties—while retaining strong syntactical composition rules and clear semantics.

# References

1. Backes, M., Dürmuth, M., Hofheinz, D., Küsters, R.: Conditional reactive simulatability. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) Computer Security – ESORICS 2006. pp. 424–443. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
2. Blum, M.: Coin flipping by telephone a protocol for solving impossible problems. SIGACT News **15**(1), 23–27 (Jan 1983). https://doi.org/10.1145/1008908.1008911
3. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) Advances in Cryptology — CRYPTO 2001. pp. 213–229. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
4. Broadnax, B., Döttling, N., Hartung, G., Müller-Quade, J., Nagel, M.: Concurrently composable security with shielded super-polynomial simulators. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017. pp. 351–381. Springer International Publishing, Cham (2017)
5. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: 42nd IEEE Symposium on Foundations of Computer Science – FOCS 2001. pp. 136–145. IEEE Computer Society (2001)
6. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) Advances in Cryptology — CRYPTO 2001. pp. 19–40. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
7. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) Advances in Cryptology — EUROCRYPT 2003. pp. 255–271. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
8. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L.R. (ed.) Advances in Cryptology – EUROCRYPT 2002. pp. 337–351. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
9. Demay, G., Gaži, P., Maurer, U., Tackmann, B.: Per-session security: Password-based cryptography revisited. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) Computer Security – ESORICS 2017. pp. 408–426. Springer International Publishing, Cham (2017)
10. Hofheinz, D., Matt, C., Maurer, U.: Idealizing identity-based encryption. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015. Lecture Notes in Computer Science, vol. 9452, pp. 495–520. Springer Berlin Heidelberg (2015)
11. Jost, D., Maurer, U., Mularczyk, M.: A unified and composable take on ratcheting. In: Hofheinz, D., Rosen, A. (eds.) Theory of Cryptography — TCC 2019. pp. 180–210. Springer International Publishing, Cham (Dec 2019)

12. Kuesters, R., Tuengerthal, M., Rausch, D.: The iitm model: a simple and expressive model for universal composability. Cryptology ePrint Archive, Report 2013/025 (2013), https://eprint.iacr.org/2013/025

13. Lindell, Y.: General composition and universal composability in secure multiparty computation. Journal of Cryptology **22**(3), 395–428 (Jul 2009)

14. Maurer, U.: Constructive Cryptography–A New Paradigm for Security Definitions and Proofs. In: Theory of Security and Applications – TOSCA 2011, pp. 33–56. Springer Berlin Heidelberg (2011)

15. Maurer, U.: Indistinguishability of random systems. In: Knudsen, L.R. (ed.) Advances in Cryptology – EUROCRYPT 2002. pp. 110–132. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)

16. Maurer, U., Pietrzak, K., Renner, R.: Indistinguishability amplification. In: Menezes, A. (ed.) Advances in Cryptology – CRYPTO 2007. pp. 130–149. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

17. Maurer, U., Renner, R.: Abstract cryptography. In: In Innovations in Computer Science – ICS 2011, pp. 1–21. Tsinghua University (2011)

18. Maurer, U., Renner, R.: From Indifferentiability to Constructive Cryptography (and Back). In: Theory of Cryptography – TCC 2016, pp. 3–24. Springer Berlin Heidelberg (2016)

19. Maurer, U., Tackmann, B.: On the soundness of authenticate-then-encrypt: Formalizing the malleability of symmetric encryption. In: Keromytis, A.D., Shmatikov, V. (eds.) Proceedings of the 17th ACM Conference on Computer and Communication Security. pp. 505–515. ACM, ACM (Oct 2010)

20. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) Advances in Cryptology — CRYPTO 2002. pp. 111–126. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)

21. Pass, R.: Simulation in quasi-polynomial time, and its application to protocol composition. In: Biham, E. (ed.) Advances in Cryptology — EUROCRYPT 2003. pp. 160–176. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)

22. Pfitzmann, B., Waidner, M.: A model for asynchronous reactive systems and its application to secure message transmission. In: Proceedings 2001 IEEE Symposium on Security and Privacy – S&P 2001. pp. 184–200 (May 2001). https://doi.org/10.1109/SECPRI.2001.924298

23. Prabhakaran, M., Sahai, A.: New notions of security: Achieving universal composability without trusted setup. In: Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing. pp. 242–251. STOC '04, ACM, New York, NY, USA (2004). https://doi.org/10.1145/1007352.1007394

# A   Details of Section 2

## A.1   Proof of Theorem 2

**Theorem 2.** *Let $\mathcal{R}$ be an arbitrary specification, and let $\epsilon_1$ and $\epsilon_2$ be arbitrary $\epsilon$-relaxations. Then we have $\left(\mathcal{R}^{\epsilon_1}\right)^{\epsilon_2} \subseteq \mathcal{R}^{\epsilon_1+\epsilon_2}$.*

*Proof.* Let $\mathsf{T} \in (\mathcal{R}^{\epsilon_1})^{\epsilon_2}$ be arbitrary. By the definition of the $\epsilon$-relaxation,

$$(\mathcal{R}^{\epsilon_1})^{\epsilon_2} = \bigcup_{\mathsf{S} \in \mathcal{R}^{\epsilon_1}} \left\{\mathsf{T} \mid \forall \mathsf{D} : |\Delta^{\mathsf{D}}(\mathsf{S}, \mathsf{T})| \le \epsilon_2(\mathsf{D})\right\},$$

there exists a $\mathsf{S} \in \mathcal{R}^{\epsilon_1}$, such that for all $\mathsf{D}$, $\Delta^{\mathsf{D}}(\mathsf{S}, \mathsf{T}) \le \epsilon_2(\mathsf{D})$. Analogously, invoking the definition of the $\epsilon$-relaxation once more, there moreover must exist a $\mathsf{R} \in \mathcal{R}$, such that that for all $\mathsf{D}$, $\Delta^{\mathsf{D}}(\mathsf{R}, \mathsf{S}) \le \epsilon_1(\mathsf{D})$. Using the triangle inequality we, thus, obtain

$$\Delta^{\mathsf{D}}(\mathsf{R}, \mathsf{T}) \le \Delta^{\mathsf{D}}(\mathsf{R}, \mathsf{S}) + \Delta^{\mathsf{D}}(\mathsf{S}, \mathsf{T}) \le \epsilon_1(\mathsf{D}) + \epsilon_1(\mathsf{D}),$$

and, hence, $\mathsf{T} \in (\mathsf{R}^{\epsilon_1+\epsilon_2}) \subseteq (\mathcal{R}^{\epsilon_1+\epsilon_2})$.   □

## A.2   Proof of Theorem 3

**Theorem 3.** *The $\epsilon$-relaxation is* compatible *with protocol application in the following sense that $\boldsymbol{\pi}\left(\mathcal{R}^{\epsilon}\right) \subseteq (\boldsymbol{\pi}\mathcal{R})^{\epsilon_{\boldsymbol{\pi}}}$, for $\epsilon_{\boldsymbol{\pi}}(\mathsf{D}) := \epsilon(\mathsf{D}\boldsymbol{\pi}(\cdot))$, where $\mathsf{D}\boldsymbol{\pi}(\cdot)$ denotes the distinguisher that first attaches $\boldsymbol{\pi}$ to the given resource and then executes $\mathsf{D}$. Moreover, it is compatible with parallel composition, i.e., $\left[\mathcal{R}^{\epsilon}, \mathcal{S}\right] \subseteq [\mathcal{R}, \mathcal{S}]^{\epsilon_{\mathcal{S}}}$, for $\epsilon_{\mathcal{S}}(\mathsf{D}) := \sup_{\mathsf{S} \in \mathcal{S}} \epsilon(\mathsf{D}[\,\cdot\,, \mathsf{S}])$, where $\mathsf{D}[\,\cdot\,, \mathsf{S}]$ denotes the distinguisher that emulates $\mathsf{S}$ in parallel to the given resource and then lets $\mathsf{D}$ interact with them.*

*Proof.* Let $\mathsf{S} \in \boldsymbol{\pi}\left(\mathcal{R}^{\epsilon}\right)$ be arbitrary. Then, by definition, there exists a $\mathsf{T} \in \mathcal{R}^{\epsilon}$ such that $\mathsf{S} = \boldsymbol{\pi}\mathsf{T}$. Observe that $\mathsf{T} \in \mathcal{R}^{\epsilon}$ directly implies $|\Delta^{\mathsf{D}}(\mathsf{R}, \mathsf{S})| \le \epsilon$, and thus

$$|\Delta^{\mathsf{D}}(\boldsymbol{\pi}\mathsf{R}, \mathsf{S})| = |\Delta^{\mathsf{D}}(\boldsymbol{\pi}\mathsf{R}, \boldsymbol{\pi}\mathsf{T})| = |\Delta^{\mathsf{D}\boldsymbol{\pi}(\cdot)}(\mathsf{R}, \mathsf{T})| \le \epsilon(\mathsf{D}\boldsymbol{\pi}(\cdot)) = \epsilon_{\boldsymbol{\pi}}(\mathsf{D}),$$

implying $\mathsf{S} \in (\boldsymbol{\pi}\mathcal{R})^{\epsilon_{\boldsymbol{\pi}}}$, i.e., compatibility with protocol application.

Now, let $\mathsf{U} \in \left[\mathcal{R}^{\epsilon}, \mathcal{S}\right]$ be arbitrary. Then, by definition there exists $\mathsf{V} \in \mathcal{R}^{\epsilon}$ and $\mathsf{S} \in \mathcal{S}$ such that $\mathsf{U} = [\mathsf{V}, \mathsf{S}]$. Moreover, by definition there exists $\mathsf{R} \in \mathcal{R}$ such that $|\Delta^{\mathsf{D}}(\mathsf{R}, \mathsf{V})| \le \epsilon$. As a consequence we obtain

$$|\Delta^{\mathsf{D}}([\mathsf{R}, \mathsf{S}], \mathsf{U})| = |\Delta^{\mathsf{D}}([\mathsf{R}, \mathsf{S}], [\mathsf{V}, \mathsf{S}])| = |\Delta^{\mathsf{D}[\,\cdot\,, \mathsf{V}]}(\mathsf{R}, \mathsf{V})| \le \epsilon(\mathsf{D}[\,\cdot\,, \mathsf{V}]) \le \epsilon_{\mathcal{S}}(\mathsf{D}),$$

and thus $\mathsf{U} \in [\mathcal{R}, \mathcal{S}]^{\epsilon_{\mathcal{S}}}$, concluding the proof.   □

### A.3 Proof of Corollary 1

**Corollary 1.** *For any specifications $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$, any protocols $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$, and any $\epsilon$-relaxation $\epsilon$ and $\epsilon'$, we have*

*1. $\mathcal{R} \xrightarrow{\boldsymbol{\pi}} \mathcal{S}^{\epsilon} \wedge \mathcal{S} \xrightarrow{\boldsymbol{\pi}'} \mathcal{T}^{\epsilon'} \implies \mathcal{R} \xrightarrow{\boldsymbol{\pi}' \circ \boldsymbol{\pi}} \mathcal{T}^{\epsilon_{\boldsymbol{\pi}'} + \epsilon'}$,*
*2. $\mathcal{R} \xrightarrow{\boldsymbol{\pi}} \mathcal{S}^{\epsilon} \implies [\mathcal{R}, \mathcal{T}] \xrightarrow{\boldsymbol{\pi}} [\mathcal{S}, \mathcal{T}]^{\epsilon_{\mathcal{T}}}$,*

*where $\epsilon_{\boldsymbol{\pi}}$ and $\epsilon_{\mathcal{S}}$ are defined as in Theorem 3, respectively.*

*Proof.* First, observe that by Theorem 3 we have

$$\boldsymbol{\pi}\left(\mathcal{R}^{\epsilon}\right) \subseteq (\boldsymbol{\pi}\mathcal{R})^{\epsilon_{\boldsymbol{\pi}}} \subseteq \mathcal{S}^{\epsilon_{\boldsymbol{\pi}}},$$

where in the second step we used Definition 1, i.e., $\mathcal{R} \xrightarrow{\boldsymbol{\pi}} \mathcal{S} :\iff \boldsymbol{\pi}\mathcal{R} \subseteq \mathcal{S}$, and Proposition 3. Now, observe that for the first property, by Theorem 2, we have $\left(\mathcal{T}^{\epsilon'}\right)^{\epsilon_{\boldsymbol{\pi}'}} \subseteq \mathcal{T}^{\epsilon_{\boldsymbol{\pi}'} + \epsilon'}$. Hence, we obtain $\mathcal{S}^{\epsilon} \xrightarrow{\boldsymbol{\pi}'} \mathcal{T}^{\epsilon_{\boldsymbol{\pi}'} + \epsilon'}$ and, thus, the property then follows by Theorem 1. The second property follows from Theorem 3:

$$\boldsymbol{\pi}\left[\mathcal{R}, \mathcal{T}\right] = \left[\boldsymbol{\pi}\mathcal{R}, \mathcal{T}\right] \subseteq \left[\mathcal{S}^{\epsilon}, \mathcal{T}\right] \subseteq [\mathcal{S}, \mathcal{T}]^{\epsilon_{\mathcal{S}}}.$$

## B Details of Section 4

### B.1 Example: The Behavior of a Resource After an Event

Consider the following resource $\text{from}_{\mathcal{E}^{\text{leaked}}_{\text{EncKey}}}(\mathsf{SecCh})$, which only answers queries once the environment triggered the event $\mathcal{E}^{\text{leaked}}_{\text{EncKey}}$. Thus, in contrast to $\mathsf{SecCh}$, this resource always leaks the full message of the adversary, in line with our intuition that it describes the behavior after the key has been exposed.

---

**Resource** $\text{from}_{\mathcal{E}^{\text{leaked}}_{\text{EncKey}}}(\mathsf{SecCh})$

| | |
|---|---|
| **Initialization** | **Interface E** |
| $M_{\mathsf{A}}[\cdot] \leftarrow \perp$ | **Input:** $(\texttt{leak}, i)$ |
| $m_{\mathsf{B}} \leftarrow \perp$ | **require** $\mathcal{E}^{\text{leaked}}_{\text{EncKey}}$ |
| $n \leftarrow 0$ | **output** $M_{\mathsf{A}}[i]$ |
| **Interface A** | **Input:** $(\texttt{inject}, m)$ |
| **Input:** $(\texttt{send}, m)$ | **require** $\mathcal{E}^{\text{leaked}}_{\text{EncKey}}$ |
| **require** $\mathcal{E}^{\text{leaked}}_{\text{EncKey}}$ | [rest as in SecCh] |
| [rest as in SecCh] | |
| | **Interfaces** $\{\texttt{E}, \texttt{W}\}$ |
| **Interface B** | **Input:** $(\texttt{deliver}, i)$ |
| **Input:** receive | **require** $\mathcal{E}^{\text{leaked}}_{\text{EncKey}}$ |
| **require** $\mathcal{E}^{\text{leaked}}_{\text{EncKey}}$ | [rest as in SecCh] |
| [rest as in SecCh] | |

---

## B.2 Proof of Theorem 5

**Theorem 5.** *The until-relaxation is compatible with protocol attachment, i.e.,* $\boldsymbol{\pi}\big(\mathcal{R}^{P]}\big) \subseteq \big(\boldsymbol{\pi}\mathcal{R}\big)^{P]}$ *and with parallel composition, i.e.,* $\big[\mathcal{R}^{P]}, \mathcal{S}\big] \subseteq \big[\mathcal{R}, \mathcal{S}\big]^{P]}$.

*Proof.* For the first property, consider an arbitrary element of $\boldsymbol{\pi}\big(\mathcal{R}^{P]}\big)$, i.e., $\boldsymbol{\pi}\mathsf{T}$ for an arbitrary $\mathsf{T} \in \mathcal{R}^{P]}$. Hence, there must exist a $\mathsf{R} \in \mathcal{R}$ such that $\mathrm{until}_P(\mathsf{R}) = \mathrm{until}_P(\mathsf{T})$. Using that $\mathrm{until}_P(\boldsymbol{\pi}\,\mathrm{until}_P(\mathsf{R})) = \mathrm{until}_P(\boldsymbol{\pi}\mathsf{R})$ for any resource $\mathsf{R}$, implies

$$\mathrm{until}_P(\boldsymbol{\pi}\mathsf{R}) = \mathrm{until}_P(\boldsymbol{\pi}\,\mathrm{until}_P(\mathsf{R})) = \mathrm{until}_P(\boldsymbol{\pi}\,\mathrm{until}_P(\mathsf{T})) = \mathrm{until}_P(\boldsymbol{\pi}\mathsf{T})$$

and, thus, $\boldsymbol{\pi}\mathsf{T} \in \big(\boldsymbol{\pi}\mathsf{R}\big)^{P]} \subseteq \big(\boldsymbol{\pi}\mathcal{R}\big)^{P]}$. The second property follows analogously observing that $\mathrm{until}_P([\mathrm{until}_P(\mathsf{R}), \mathsf{S}]) = \mathrm{until}_P([\mathsf{R}, \mathsf{S}])$ for all $\mathsf{R}$ and $\mathsf{S}$. □

## B.3 Proof of Theorem 6

**Theorem 6.** *There exist specifications $\mathcal{R}$ and $\mathcal{S}$, a monotone predicate $P$, and a function $\epsilon$ mapping distinguishers to values in $[0, 1]$ such that*

$$\big(\mathcal{R}^{P]}\big)^{\epsilon} \not\subseteq \big(\mathcal{R}^{\epsilon}\big)^{P]} \qquad \text{and} \qquad \big(\mathcal{S}^{\epsilon}\big)^{P]} \not\subseteq \big(\mathcal{S}^{P]}\big)^{\epsilon}.$$

*Proof.* In the following, let $\mathsf{R}$ denote the following resource that provides two interfaces: $\mathsf{R}$ initially chooses a seed $s \in \{0, 1\}^m$ uniformly at random. Upon a trigger input at the first interface it outputs $\mathrm{PRG}(s)$ (for some $m$-bits to $n$-bits PRG). Upon a trigger input at the second interface, it triggers a $\mathcal{E}^{\mathsf{leaked}}$ event and outputs $s$. Let the resource $\mathsf{S}$ work analogously except that it outputs a uniformly distributed an independent value $t \in \{0, 1\}^n$ instead of the PRG output. Furthermore, let $P(\mathcal{E}) \coloneqq \mathcal{E}^{\mathsf{leaked}}$ and let $\epsilon$ denote the reduction to the security of the PRG.

Using $\mathcal{R} \coloneqq \{\mathsf{R}\}$ it is now easy to see that $\mathsf{S} \in (\mathcal{R}^{P]})^{\epsilon}$: Consider a hybrid system $\mathsf{T}$ that outputs $\mathrm{PRG}(s')$ and $s$ for independent and $s$ and $s'$ u.a.r. Then, we have $\mathsf{T} \in \mathcal{R}^{P]}$, since the first output is equally distributed and the second blinded. Moreover, distinguishing $\mathsf{T}$ from $\mathsf{S}$ boils down to breaking the PRG-security, and thus $\mathsf{S} \in \mathsf{T}^{\epsilon}$. On the other hand, one can show that $\mathsf{S} \notin \big(\mathcal{R}^{\epsilon}\big)^{P]}$. To see this, observe that if the PRG is secure, then $\mathcal{R}^{\epsilon} \approx \mathcal{R}$. Especially, leaking the seed prevents us from replacing the first output by a truly random string. Thus, applying the until-relaxation still does not contain $\mathsf{S}$, concluding the first part of the proof. For the other direction consider $\mathcal{S} \coloneqq \{\mathsf{S}\}$. Using an analogous argument one can show that $\mathsf{R} \in \big(\mathcal{S}^{\epsilon}\big)^{P]}$ but $\mathsf{R} \notin \big(\mathcal{S}^{P]}\big)^{\epsilon}$. □

## B.4 Proof of Theorem 9

**Theorem 9.** *There exist specifications $\mathcal{R}$ and $\mathcal{S}$, a monotone predicate $P(\mathcal{E})$, and a function $\epsilon$ mapping distinguishers to values in $[0, 1]$ such that*

$$\big(\mathcal{R}^{[P]}\big)^{\epsilon} \not\subseteq \big(\mathcal{R}^{\epsilon}\big)^{[P} \qquad \text{and} \qquad \big(\mathcal{S}^{\epsilon}\big)^{[P} \not\subseteq \big(\mathcal{S}^{[P]}\big)^{\epsilon}.$$

*Proof.* The counter example works similarly to the one from Theorem 6. Consider a resource $\mathsf{R}$ with two interfaces that initially chooses a PRG-seed $s$ uniformly at random. At the first interface it outputs $\mathrm{PRG}(s)$. At the second interface it outputs $s$, but only before some event $\mathcal{E}^{\mathsf{secure}}$. The second resource $\mathsf{S}$ works analogously except that it outputs an independent uniform random string at the first interface instead of the PRG output. Let now $\mathcal{R} := \{\mathsf{R}\}$, $\mathcal{S} := \{\mathsf{S}\}$, and $P := \{\mathcal{E}^{\mathsf{secure}}\}$. Analogous to the proof from Theorem 6, one can now show that $\mathsf{S} \in \left(\mathcal{R}^{[P]}\right)^{\epsilon}$ but $\mathsf{S} \notin \left(\mathcal{R}^{\epsilon}\right)^{[P}$, and furthermore that $\mathsf{R} \in \left(\mathcal{S}^{\epsilon}\right)^{[P}$ but $\mathsf{S} \notin \left(\mathcal{S}^{[P]}\right)^{\epsilon}$. $\qquad\square$

## B.5  Proof of Theorem 10

**Theorem 10.** *For any resource $\mathsf{R}$ and any monotone predicates $P_1$ and $P_2$, we have*

$$\mathsf{R}^{[P_1, P_2]} = \bigcup_{n \in \mathbb{N}} \left( \bigcup \{ \mathsf{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n} \mid \forall i \leq n : \phi_i \in \{[P_2], [P_1]\} \} \right)$$

$$= \left( \left( \mathsf{R}^{[P_1]} \right)^{P_2]} \right)^{[P_1} = \left( \left( \mathsf{R}^{P_2]} \right)^{[P_1]} \right)^{P_2]},$$

*where $\mathsf{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n}$ is a shorthand notation for first applying $\phi_1$, then $\phi_2$, until $\phi_n$.*

*Proof.* We prove the theorem in three steps:

*Claim.* $\mathcal{R}^{[P_1, P_2]} = \left( \left( \mathcal{R}^{[P_1]} \right)^{P_2]} \right)^{[P_1}$

*Proof (of claim).* Let $\mathsf{S} \in \left( \left( \mathcal{R}^{[P_1]} \right)^{P_2]} \right)^{[P_1}$ be arbitrary. This means that there exist $\mathsf{T} \in \left( \mathcal{R}^{[P_1]} \right)^{P_2]}$, $\mathsf{U} \in \mathcal{R}^{[P_1}$ and $\mathsf{R} \in \mathcal{R}$ such that

$$\mathrm{from}_{P_1}(\mathsf{S}) = \mathrm{from}_{P_1}(\mathsf{T}), \tag{1}$$
$$\mathrm{until}_{P_2}(\mathsf{T}) = \mathrm{until}_{P_2}(\mathsf{U}), \tag{2}$$
$$\mathrm{from}_{P_1}(\mathsf{U}) = \mathrm{from}_{P_1}(\mathsf{R}). \tag{3}$$

Combining these properties with the commutativity of from and until, we obtain

$$\mathrm{until}_{P_2}(\mathrm{from}_{P_1}(\mathsf{R})) = \mathrm{until}_{P_2}(\mathrm{from}_{P_1}(\mathsf{U})) = \mathrm{from}_{P_1}(\mathrm{until}_{P_2}(\mathsf{U}))$$
$$= \mathrm{from}_{P_1}(\mathrm{until}_{P_2}(\mathsf{T})) = \mathrm{until}_{P_2}(\mathrm{from}_{P_1}(\mathsf{T}))$$
$$= \mathrm{until}_{P_2}(\mathrm{from}_{P_1}(\mathsf{S})),$$

implying that $\mathsf{S} \in \mathsf{R}^{[P_1, P_2]}$ and thus $\mathcal{R}^{[P_1, P_2]} \supseteq \left( \left( \mathcal{R}^{[P_1]} \right)^{P_2]} \right)^{[P_1}$.

Now, let $\mathsf{S} \in \mathcal{R}^{[P_1, P_2]}$ be arbitrary. By definition, we thus know that there exists a $\mathsf{R} \in \mathcal{R}$ such that $\mathrm{from}_{P_1}(\mathsf{S}) \in \left( \mathrm{from}_{P_1}(\mathsf{R}) \right)^{P_2]}$. Combining this with the basic fact that $\mathrm{from}_{P_1}(\mathsf{R}) \in \mathsf{R}^{[P_1}$, we obtain $\mathrm{from}_{P_1}(\mathsf{S}) \in \left( \mathsf{R}^{[P_1} \right)^{P_2]}$. Combining this in turn with the basic fact that $\mathsf{S} \in \mathsf{S}^{[P_1}$ then yields $\mathsf{S} \in \left( \left( \mathsf{R}^{[P_1} \right)^{P_2]} \right)^{[P_1}$ and thus $\mathcal{R}^{[P_1, P_2]} \subseteq \left( \left( \mathcal{R}^{[P_1]} \right)^{P_2]} \right)^{[P_1}$. $\qquad\diamond$

*Claim.* $\mathcal{R}^{[P_1,P_2]} = \left(\left(\mathcal{R}^{P_2}\right)^{[P_1]}\right)^{P_2]}$

*Proof (of claim).* Observing that

$$\mathcal{R}^{[P_1,P_2]} = \left\{ \mathsf{S} \mid \mathrm{until}_{P_2}(\mathrm{from}_{P_1}(\mathsf{R})) = \mathrm{until}_{P_2}(\mathrm{from}_{P_1}(\mathsf{S})) \right\}$$

(using the commutativity), it is easy to see that the proof follows analogously. $\Diamond$

*Claim.* $\mathcal{R}^{[P_1,P_2]} = \bigcup_{n\in\mathbb{N}} \left( \bigcup \{ \mathcal{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n} \mid \forall i \leq n : \phi_i \in \{P_2\}, [P_1] \} \right)$

*Proof (of claim).* Using the previous claims, it is trivial to see that

$$\mathcal{R}^{[P_1,P_2]} \subseteq \bigcup_{n\in\mathbb{N}} \left( \bigcup \{ \mathcal{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n} \mid \forall i \leq n : \phi_i \in \{P_2\}, [P_1] \} \right).$$

For the other direction, we proceed by induction over $n$. Note that without loss of generality (by Theorems 4 and 7), we can assume the order of the relaxations to strictly alternate. Furthermore, the previous claims already prove it for $n = 3$. Hence, the relation is also trivial for $n < 3$, as adding a further relaxation only enlarges the set. Assume now as the induction hypothesis that for some $n \geq 3$

$$\mathcal{R}^{[P_1,P_2]} \supseteq \bigcup \{ \mathcal{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n} \mid \forall i \leq n : \phi_i \in \{P_2\}, [P_1] \}.$$

We want to show that $\mathsf{R}^{\phi_1 \cdot \phi_2 \cdots \phi_{n+1}} \in \mathsf{R}^{[P_1,P_2]}$ as well. Assume w.l.o.g. that $\phi_{n+1} = P_2]$. By the induction hypothesis, we have that $\mathsf{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n} \in \mathsf{R}^{[P_1,P_2]}$ and thus by the second claim $\mathsf{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n} \in \left(\left(\mathsf{R}^{P_2}\right)^{[P_1]}\right)^{P_2]}$ and thus

$$\mathsf{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n} \in \left(\left(\left(\mathsf{R}^{P_2}\right)^{[P_1]}\right)^{P_2]}\right)^{P_2]} = \left(\left(\left(\mathsf{R}^{P_2}\right)^{[P_1]}\right)^{P_2]}\right)^{P_2]},$$

where the second step follows from Theorem 4. $\Diamond$

### B.6  Proof of Theorem 13

**Theorem 13.** *Let $P_1$ and $P_2$ be two monotone predicates, and let $\epsilon$ be a function mapping distinguishers to values in $[0,1]$. Then, for any specification $\mathcal{R}$ we have*

$$\left(\mathcal{R}^{[P_1,P_2]:\epsilon}\right)^{[P_1',P_2']:\epsilon'} \subseteq \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']:\epsilon_{[P_1 \wedge P_1', P_2 \vee P_2']} + \epsilon'_{[P_1 \wedge P_1', P_2 \vee P_2']}},$$

*where $\epsilon_{[P_1 \wedge P_1', P_2 \vee P_2']}(D) := \epsilon(\mathsf{D} \circ \mathrm{until}_{P_2 \vee P_2'} \circ \mathrm{from}_{P_1 \wedge P_1'})$, i.e., the performance of the distinguisher interacting with the projected resource, and analogously for $\epsilon'_{[P_1 \wedge P_1', P_2 \vee P_2']}$.*

*Proof.* Observe that it suffices to show

$$\left(\left(\left(\mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']}\right)^{\epsilon}\right)^{[P_1 \wedge P_1', P_2 \vee P_2']}\right)^{\epsilon'}$$

$$\subseteq \left(\left(\mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']}\right)^{\epsilon_{[P_1,P_2]} + \epsilon'_{[P_1,P_2]}}\right)^{[P_1 \wedge P_1', P_2 \vee P_2']}.$$

The rest then follows trivially by Theorems 2 and 11. To this end, consider an arbitrary $S \in \left( \left( \left( \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']} \right)^\epsilon \right)^{[P_1 \wedge P_1', P_2 \vee P_2']} \right)^{\epsilon'}$. Hence, there must exist a $T \in \left( \left( \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']} \right)^\epsilon \right)^{[P_1 \wedge P_1', P_2 \vee P_2']}$, a $U \in \left( \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']} \right)^\epsilon$, and a $V \in \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']}$ such that

$$\left| \Delta^D(S, T) \right| \leq \epsilon'(D) \tag{4}$$

$$\mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(T)) = \mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(U)) \tag{5}$$

$$\left| \Delta^D(U, V) \right| \leq \epsilon(D). \tag{6}$$

Using those properties, we obtain

$$\left| \Delta^D \left( \mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(V)), \mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(S)) \right) \right|$$
$$\leq \left| \Delta^D \left( \mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(V)), \mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(U)) \right) \right|$$
$$+ \left| \Delta^D \left( \mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(T)), \mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(S)) \right) \right|$$
$$\leq \left| \Delta^{D \circ \mathrm{until}_{P_2 \vee P_2'}(\cdot) \circ \mathrm{from}_{P_1 \wedge P_1'}(\cdot)}(V, U) \right| + \left| \Delta^{D \circ \mathrm{until}_{P_2 \vee P_2'}(\cdot) \circ \mathrm{from}_{P_1 \wedge P_1'}(\cdot)}(T, S) \right|$$
$$\leq \epsilon_{[P_1 \wedge P_1', P_2 \vee P_2']}(D) + \epsilon'_{[P_1 \wedge P_1', P_2 \vee P_2']}(D).$$

Now observe that $V \in \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']}$ implies

$$\mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(V)) \in \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']},$$

and thus,

$$\mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(S)) \in \left( \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']} \right)^{\epsilon_{[P_1 \wedge P_1', P_2 \vee P_2']} + \epsilon'_{[P_1 \wedge P_1', P_2 \vee P_2']}}.$$

As a result, we have

$$S \in \left( \left( \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']} \right)^{\epsilon_{[P_1 \wedge P_1', P_2 \vee P_2']} + \epsilon'_{[P_1 \wedge P_1', P_2 \vee P_2']}} \right)^{[P_1 \wedge P_1', P_2 \vee P_2']},$$

concluding the proof. $\qquad \square$

### B.7   Proof of Proposition 6

**Proposition 6.** *Let $\pi_{\mathsf{ENC}} = (\pi_{\mathsf{enc}}, \pi_{\mathsf{dec}})$ denote the protocol securing communication using a symmetric encryption scheme. Then, for the resources in Figure 1, there exist (efficient) simulators $\sigma_1$, $\sigma_2$, and $\sigma_3$ such that*

$$[\mathsf{EncKey}, \mathsf{AuthCh}] \xrightarrow{\pi_{\mathsf{ENC}}} \bigcap_{(P_1, P_2, \epsilon, \sigma) \in \Omega} (\sigma \mathsf{SecCh})^{[P_1, P_2]:\epsilon}$$

*for*

$$\Omega := \left\{ \left( \mathtt{true}, \mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}} \vee \mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}, \epsilon_{\mathsf{CPA}}, \sigma_1 \right), \left( \mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}}, \mathtt{false}, 0, \sigma_2 \right), \right.$$
$$\left. \left( \mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}, \mathtt{false}, 0, \sigma_3 \right) \right\},$$

*where $\epsilon_{\mathsf{CPA}}$ denotes a simple reduction from distinguishing the secure and authenticated channel (without key leakage) to the IND-CPA game.*

*Proof (Sketch).* Let $\boldsymbol{\sigma}_1$ be the usual simulator that creates fake ciphertexts by encrypting $0^{|m|}$ instead of $m$, when queried for the leakage of the channel. By definition, we have $\mathrm{until}_{\mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}} \vee \mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}} (\boldsymbol{\sigma}_1 \mathsf{SecCh}) \in (\boldsymbol{\sigma}_1 \mathsf{SecCh})^{[\mathtt{true}, \mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}} \vee \mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}]}$. Thus, if the encryption scheme is IND-CPA secure, then it is easy to see that

$$\mathrm{until}_{\mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}} \vee \mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}} (\boldsymbol{\pi}_{\mathsf{ENC}}[\mathsf{EncKey}, \mathsf{AuthCh}]) \in \left( (\boldsymbol{\sigma}_1 \mathsf{SecCh})^{[\mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}} \vee \mathcal{E}_{\mathsf{AuthKey}}^{\mathsf{leaked}}]} \right)^{\epsilon_{\mathsf{CPA}}},$$

which proves the construction inside the first interval. For the second interval, consider the simulator $\boldsymbol{\sigma}_2$ that encrypts the real messages. Using correctness of the encryption scheme, it is easy to see that

$$\mathrm{from}_{\mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}}} (\boldsymbol{\pi}_{\mathsf{ENC}}[\mathsf{EncKey}, \mathsf{AuthCh}]) = \mathrm{from}_{\mathcal{E}_{\mathsf{EncKey}}^{\mathsf{leaked}}} (\boldsymbol{\sigma}_2 \mathsf{SecCh}),$$

and analogous for the third interval using the same simulator $\boldsymbol{\sigma}_3 = \boldsymbol{\sigma}_2$. $\qquad\square$

## B.8 Proof of Theorem 15

**Theorem 15.** *Let $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$ be arbitrary specifications, let $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$ be arbitrary protocols, and let $\Omega$ and $\Omega'$ be arbitrary interval-wise guarantees. Then, we have*

$$\mathcal{R} \xrightarrow{\boldsymbol{\pi}} \bigcap_{(P_1, P_2, \epsilon, \boldsymbol{\sigma}) \in \Omega} (\boldsymbol{\sigma}\mathcal{S})^{[P_1, P_2]:\epsilon} \quad \wedge \quad \mathcal{S} \xrightarrow{\boldsymbol{\pi}'} \bigcap_{(P_1', P_2', \epsilon', \boldsymbol{\sigma}') \in \Omega'} (\boldsymbol{\sigma}'\mathcal{T})^{[P_1', P_2']:\epsilon'}$$

$$\implies \quad \mathcal{R} \xrightarrow{\boldsymbol{\pi}' \circ \boldsymbol{\pi}} \bigcap_{\substack{(P_1, P_2, \epsilon, \boldsymbol{\sigma}) \in \Omega \\ (P_1', P_2', \epsilon', \boldsymbol{\sigma}') \in \Omega'}} (\boldsymbol{\sigma}\boldsymbol{\sigma}'\mathcal{T})^{[P_1 \wedge P_1', P_2 \vee P_2']:\tilde{\epsilon}},$$

*where $\tilde{\epsilon} := (\epsilon_{\boldsymbol{\pi}'})_{[P_1 \wedge P_1', P_2 \vee P_2']} + (\epsilon'_{\boldsymbol{\sigma}})_{[P_1 \wedge P_1', P_2 \vee P_2']}$. Furthermore, we have*

$$\mathcal{R} \xrightarrow{\boldsymbol{\pi}} \bigcap_{(P_1, P_2, \epsilon, \boldsymbol{\sigma}) \in \Omega} (\boldsymbol{\sigma}\mathcal{S})^{[P_1, P_2]:\epsilon} \implies [\mathcal{R}, \mathcal{T}] \xrightarrow{\boldsymbol{\pi}} \bigcap_{(P_1, P_2, \epsilon, \boldsymbol{\sigma}) \in \Omega} (\boldsymbol{\sigma}[\mathcal{S}, \mathcal{T}])^{[P_1, P_2]:\epsilon_{\mathcal{T}}}.$$

*Proof.* We first prove sequential composition. In the following, let $(P_1, P_2, \epsilon, \boldsymbol{\sigma}) \in \Omega$ and $(P_1', P_2', \epsilon', \boldsymbol{\sigma}') \in \Omega'$. From the first part of the assumption, Theorem 14, and composition order invariance we obtain

$$\boldsymbol{\pi}'\boldsymbol{\pi}\mathcal{R} \subseteq \boldsymbol{\pi}'\big((\boldsymbol{\sigma}\mathcal{S})^{[P_1, P_2]:\epsilon}\big) \subseteq (\boldsymbol{\pi}'\boldsymbol{\sigma}\mathcal{S})^{[P_1, P_2]:\epsilon_{\boldsymbol{\pi}'}} \subseteq (\boldsymbol{\sigma}\boldsymbol{\pi}'\mathcal{S})^{[P_1, P_2]:\epsilon_{\boldsymbol{\pi}'}}.$$

Moreover, using the second part of the assumption and Theorem 14 yields

$$\boldsymbol{\sigma}\boldsymbol{\pi}'\mathcal{S} \subseteq \boldsymbol{\sigma}\big((\boldsymbol{\sigma}'\mathcal{T})^{[P_1', P_2']:\epsilon'}\big) \subseteq (\boldsymbol{\sigma}\boldsymbol{\sigma}'\mathcal{T})^{[P_1', P_2']:\epsilon'_{\boldsymbol{\sigma}}}.$$

Combining the two statements, and using the monotonicity of relaxations, we get

$$\boldsymbol{\pi}'\boldsymbol{\pi}\mathcal{R} \subseteq \big((\boldsymbol{\sigma}\boldsymbol{\sigma}'\mathcal{T})^{[P_1', P_2']:\epsilon'_{\boldsymbol{\sigma}}}\big)^{[P_1, P_2]:\epsilon_{\boldsymbol{\pi}'}} \subseteq (\boldsymbol{\sigma}\boldsymbol{\sigma}'\mathcal{T})^{[P_1 \wedge P_1', P_2 \vee P_2']:\tilde{\epsilon}},$$

where in the last step we used Theorem 13, directly implying the sequential composition property.

Now consider parallel composition. We directly obtain

$$
\begin{aligned}
\boldsymbol{\pi}\big[\mathcal{R},\mathcal{T}\big] &= \big[\boldsymbol{\pi}\mathcal{R},\mathcal{T}\big] \\
&\subseteq \big[(\boldsymbol{\sigma}\mathcal{S})^{[P_1,P_2]:\epsilon},\mathcal{T}\big] \\
&\subseteq \big[\boldsymbol{\sigma}\mathcal{S},\mathcal{T}\big]^{[P_1,P_2]:\epsilon_{\mathcal{T}}} \\
&= \big(\boldsymbol{\sigma}\big[\mathcal{S},\mathcal{T}\big]\big)^{[P_1,P_2]:\epsilon_{\mathcal{T}}},
\end{aligned}
$$

where in the first and the last step we used composition order invariance, in the second step the assumption, and in the third step Theorem 14.  □

## C    Details of Section 5

### C.1    ElGamal Commitments

In this section, we provide some additional details to the ElGamal-commitment example. First, a formal definition of the corresponding converter implementing the protocol is presented in Figure 10. Second, a formal definition of the two simulators involved in the construction statements of is depicted in Figure 11. The left simulator $\sigma^{\mathsf{A}}_{\mathsf{ElG\text{-}com}}$ is used to formalize security against a potentially dishonest Alice (initiator), and the right $\sigma^{\mathsf{B}}_{\mathsf{ElG\text{-}com}}$ is used to formalize security against a potentially dishonest Bob (responder), respectively.

### C.2    Coin-Tossing

In this section, we provide some additional details on the coin-tossing example. First, in Figure 12 we prove the formal definition of the two converters of Blum's protocol for constructing a single-bit coin-toss resource. Note that in the protocol Alice acts as the initiator, and Bob as the responder, respectively. The pseudo-code description is presented in Figure 12.

Second, a formal description of the two simulators involved in the construction statements of the coin-toss resource is depicted in Figure 13. The left $\sigma^{\mathsf{A}}_{\mathsf{CT}}$ is used to formalize security against a potentially dishonest Alice (initiator), and the right $\sigma^{\mathsf{B}}_{\mathsf{CT}}$ is used to formalize security against a potentially dishonest Bob (responder), respectively.

**Converter $\pi^{\mathtt{A}}_{\mathsf{ElG\text{-}com}}$**

**Initialization**
$a, b, v \leftarrow \bot$

**Emulating Interface A of $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\mathbb{G}}$**

**Input:** $(\mathtt{commit}, m)$
  **require** only called once
  $a, b \twoheadleftarrow \mathbb{Z}_{|\mathbb{G}|}$
  $v \leftarrow m$
  **call** $(\mathtt{send}, (g^a, g^b, m \cdot g^{ab}))$
      at interface A of $\mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_1$
  $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{committed}}$
  **return** $\mathtt{ok}$

**Input:** $(\mathtt{open}, m)$
  **require** $\mathcal{E}^{\mathsf{committed}}$ and only called once
  **if** $m = v$ **then**
    | **call** $(\mathtt{send}, (a, b))$ at int. A of $\mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_2$
    | $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{opened}}$
  **return** $\mathtt{ok}$

---

**Converter $\pi^{\mathtt{B}}_{\mathsf{ElG\text{-}com}}$**

**Initialization**
$c, A, B, v \leftarrow \bot$

**Emulating Interface B of $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\mathbb{G}}$**

**Input:** $\mathtt{isCommitted}$
  **if** $c = \bot$ **then**
    | **call** $(c, A, B) \leftarrow (\mathtt{receive})$
        at interface B of $\mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_1$
  **return** $c \neq \bot$

**Input:** $\mathtt{read}$
  **if** $c = \bot$ **then**
    | **call** $(c, A, B) \leftarrow (\mathtt{receive})$
        at interface B of $\mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_1$
  **if** $v = \bot$ **then**
    | **call** $(a, b) \leftarrow (\mathtt{receive})$
        at interface B of $\mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_2$
    | **if** $(a, b) \neq \bot \land A = g^a \land B = g^b$ **then**
    |   | $v \leftarrow c \cdot g^{-ab}$
  **return** $v$

Fig. 10: A formal description of the ElGamal-commitment converters.

---

**Converter $\sigma^{\mathtt{B}}_{\mathsf{ElG\text{-}com}}$**

**Initialization**
$u, v, w \twoheadleftarrow \mathbb{G}$
$x \leftarrow \bot$

**Emulating Interface B of $\mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_1$**

**Input:** $\mathtt{receive}$
  **call** $succ \leftarrow (\mathtt{isCommitted})$
    at interface B of $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\mathbb{G}}$
  **if** $succ$ **then**
    | **return** $(u, v, w)$
  **else**
    | **return** $\bot$

**Emulating Interface B of $\mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_2$**

**Input:** $\mathtt{receive}$
  **return** $\bot$

---

**Converter $\sigma^{\mathtt{A}}_{\mathsf{ElG\text{-}com}}$**

**Initialization**
$A, B, V \leftarrow \bot$

**Emulating Interface A of $\mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_1$**

**Input:** $(\mathtt{send}, x)$
  **require** only called once
  **if** $x \in \mathbb{G}^3$ **then**
    | $(A, B, V) \leftarrow x$
  **call** $(\mathtt{commitRaw}, x)$
    at interface A of $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\mathbb{G}}$
  **return** $\mathtt{ok}$

**Emulating Interface A of $\mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_2$**

**Input:** $(\mathtt{send}, x)$
  **require** only called once
  **if** $x = (a, b) \in \mathbb{Z}_{|\mathbb{G}|} \land A = g^a \land B = g^b$ **then**
    | **call** $(\mathtt{open}, V \cdot g^{-ab})$
        at interface A of $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\mathbb{G}}$
  **return** $\mathtt{ok}$

Fig. 11: A formal description of the simulators from the ElGamal-commitment construction.

**Converter $\pi_{CT}^A$**

___

**Initialization**
$X, Y \leftarrow \bot$

___

**Emulating Interface A of $CT_{\{0,1\}}^{A,B}$**

**Input:** initiate
  **require** only called once
  $X \twoheadleftarrow \{0,1\}$
  **call** commit at int. A of $Com^{A \to B}$
  **return** ok

**Input:** read
  **require** $X \neq \bot$
  **if** $Y = \bot$ **then**
    └ **call** $Y \leftarrow$ read at int. A of $Ch^{B \to A}$
  **if** $Y \neq \bot$ **then return** $X \oplus Y$
  **else return** $\bot$

**Input:** release
  **require** $Y \neq \bot$ and only called once
  **call** (open, $X$) at int. A of $Com^{A \to B}$
  **return** ok

---

**Converter $\pi_{CT}^B$**

___

**Initialization**
$X, Y \leftarrow \bot$

___

**Emulating Interface B of $CT_{\{0,1\}}^{A,B}$**

**Input:** isInitiated
  **call** $b \leftarrow$ isCommitted at int. B of $Com^{A \to B}$
  **return** $b$

**Input:** respond
  **require** isInitiated and only called once
  $Y \twoheadleftarrow \{0,1\}$
  **call** (send, $Y$) at int. B of $Ch^{B \to A}$
  **return** ok

**Input:** read
  **require** $Y \neq \bot$
  **if** $X = \bot$ **then**
    └ **call** $X \leftarrow$ read at int. B of $Com^{A \to B}$
  **if** $X \neq \bot$ **then**
    │ **return** $X \oplus Y$
  **else**
    └ **return** $\bot$

Fig. 12: A formal description of the coin-tossing converters.

---

**Simulator $\sigma_{CT}^A$**

___

**Emulating Interface A of $Com_{\{0,1\},f}^{A \to B}$**

**Input:** (commit, $x$) $\in \{0,1\}$
  **require** $\neg \mathcal{E}^{committed}$
  $X \leftarrow x$
  **call** init at int. A of $CT^{A,B}$
  **return** ok

**Input:** (commitRaw, $c$) $\in \{0,1\}^*$
  **require** $\neg \mathcal{E}^{committed}$
  **call** init at int. A of $CT^{A,B}$
  **call** $X \leftarrow$ (getLeakage, $c$) at int. B of $CT_{\{0,1\},f}^{A,B}$
  **return** ok

**Input:** (open, $x$) $\in \{0,1\}$
  **require** $\mathcal{E}^{committed}$, only called once, and $x = X$
  **call** release at int. A of $CT^{A,B}$
  **return** ok

___

**Emulating Interface A of $Ch^{B \to A}$**

**Input:** read
  **require** $X \neq \bot \lor C \neq \bot$
  **call** $Z \leftarrow$ read at int. B of $CT_{\{0,1\},f}^{A,B}$
  **if** $Z \neq \bot$ **then return** $Z \oplus X$
  **else return** $\bot$

---

**Simulator $\sigma_{CT}^B$**

___

**Initialization**
$Z, Y \leftarrow \bot$

___

**Emulating Interface B of $Com_{\{0,1\}}^{A \to B}$**

**Input:** isCommitted
  **call** $b \leftarrow$ isInitiated at int. B of $CT^{A,B}$
  **return** $b$

**Input:** read
  **require** $Y \neq \bot$
  **if** $Z = \bot$ **then**
    └ **call** $Z \leftarrow$ read at int. B of $CT^{A,B}$
  **if** $Z \neq \bot$ **then**
    │ **return** $Z \oplus Y$
  **else**
    └ **return** $\bot$

___

**Emulating Interface B of $Ch^{B \to A}$**

**Input:** (send, $Y$) $\in \{0,1\}$
  **require** isInitiated and only called once
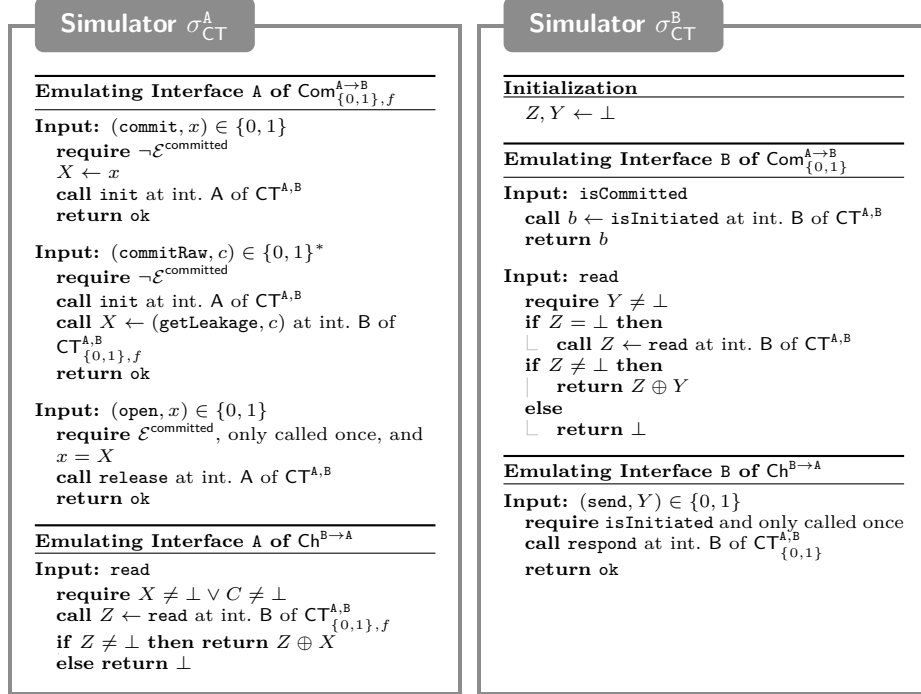  **call** respond at int. B of $CT_{\{0,1\}}^{A,B}$
  **return** ok

Fig. 13: A formal description of the respective coin-tossing simulators.