Technical Report: A General Composition Theorem
for Secure Reactive Systems*

Michael Backes¹, Birgit Pfitzmann², and Michael Waidner²

¹ Saarland University
² IBM Zurich Research Lab

October 30, 2007

Abstract. We consider compositional properties of reactive systems that are secure in a cryptographic sense. We follow the well-known simulatability approach of modern cryptography, i.e., the specification is an ideal system and a real system should in some sense simulate this ideal one. We show that if a system consists of a polynomial number of arbitrary ideal subsystems such that each of them has a secure implementation in the sense of blackbox simulatability, then one can securely replace all ideal subsystems with their respective secure counterparts without destroying the blackbox simulatability relation. We further prove our theorem for universal simulatability by showing that blackbox simulatability implies universal simulatability under reasonable assumptions. We show all our results with concrete security.

1 Introduction

In recent times, the analysis of cryptographic protocols has been getting more and more attention, and thus the demand for general frameworks for representing cryptographic protocols and their security requirements has been rising. To enable a cryptographically correct analysis of cryptographic protocols, such frameworks have to capture probabilistic behaviors, complexity-theoretically bounded adversaries as well as a reactive environment of the protocol, i.e., continuous interaction with users and an adversary, e.g., in many protocol runs. Clearly, such frameworks further have to be rigorously defined to avoid ambiguities and to enable convincing proofs. Moreover, it is highly desirable that such frameworks provide a link to formal methods, i.e., to tool-supported verification of cryptographic protocols. Tool support can minimize flaws, which occur quite often if the distributed-systems aspects of cryptographic protocols are analyzed by hand. One ingredient for this is that the model should contain an abstract machine model besides Turing machines. The model of Backes, Pfitzmann and Waidner [2, 3] is suitable for all these requirements and we use it as a rigorous foundation of this work.

The model of [2, 3] introduced a notion of security-preserving refinement, called reactive simulatability. This notion captures the idea of refinement that preserves not only integrity properties but also confidentiality properties. Intuitively it can be stated

* An earlier version of this work appeared in [1].
as follows, when applied to the relation between a real and an ideal system: Everything that can happen to users of the real system in the presence of an arbitrary adversary $A$ can also happen to the same users with the ideal system, where attack capabilities are usually much more restricted, in the presence of another adversary $A'$. In particular, it comprises confidentiality because the notion of what happens to users, called their view, not only includes their in- and outputs to the system, but also their communication with the adversary. This includes whether the adversary can guess secrets of the users or partial information about them. As it is often desirable to impose further restrictions on how the adversary $A'$ against the ideal service is constructed, simulatability comes in different flavors. The two most prominent ones (besides general simulatability as described above, which does not impose any restriction on $A'$) are universal simulatability, which states that $A'$ has to be independent of the actual users of the protocol, and the (seemingly) more restrictive notion of black-box simulatability, which states that $A'$ consists of the original adversary $A$ and a simulator that may only depend on the protocol itself.

One of the key results in the considered model is a composition theorem [2, 3]. It states that if a larger system is designed based on a specification of a subsystem, and the implementation of the subsystem is later plugged in, the entire implementation of the larger system is as secure as its design in the same sense of reactive simulatability. This theorem (as well as its predecessor [4] for a synchronous reactive model) holds for all variants of simulatability (general, universal, and black-box), but it is restricted to replacing one system. Obviously, a constant number of systems can then be replaced by applying the theorem multiple times.

In this work, we present a more comprehensive composition theorem for black-box simulatability by showing that a polynomial number (in a security parameter) of arbitrary systems can be composed without destroying the simulatability relation. The proof relies on what is often called a “standard hybrid argument” as first used in [5]. We further show that universal simulatability implies black-box simulatability under reasonable assumptions. This is of independent interest, but it in particular allows us to prove our theorem also for universal simulatability. We show all our results with concrete security.

Related Literature. Simulatability was first sketched for secure multi-party function evaluation, i.e., for the computation of one output tuple from one tuple of secret inputs from each participant in [6] and defined (with different degrees of generality and rigorosity) in [7–10]. While composition theorems for special cases were proven in [8, 9], the first general composition theorem for non-reactive simulatability was proven in [10].

An important step towards compositionality results of reactive systems was taken in [11, 12], where the cryptographic security of specific systems was directly defined and verified using a formal language, the $\pi$-calculus, and security was expressed using observational equivalence. This notion is even stronger than reactive simulatability because the entire environment (corresponding to users and adversary together for reactive

---

1 Other terms are implementation and specification, or in special cases cryptographic and abstract system.
simulatability) must not be able to distinguish the implementation and the specification. Correspondingly, the concrete specifications used were not abstract; they essentially comprise the actual protocols including all cryptographic details. Composition was defined in the calculus by defining processes with “holes” for other processes, which then allows for composing a constant number of systems.

A reactive simulatability definition was first proposed (after some earlier sketches, in particular in [13, 14, 10]) in [15]. It is synchronous, covers a restricted class of protocols (straightline programs with restricted operators, in view of the constructive result of this paper), and simulatability is defined for the information-theoretic case only, where it can be done with a quantification over input sequences instead of active honest users.

The first composition theorem for reactive simulatability was given in [4] for a general synchronous reactive model, followed by essentially the same composition theorem [2] in the corresponding asynchronous model. Later than [2] but independently, another model of asynchronous reactive systems together with a composition theorem for reactive simulatability was developed in [16]. The theorem is specific for universal simulatability, but for this case it is more general than the ones in [4, 2] since it additionally allows for securely composing a polynomial number of copies of an ideal service, which naturally correspond to different protocol instances in the real implementation. We stress that our composition theorem in this paper not only captures secure composition of a polynomial number of copies of one single ideal system but also of a polynomial number of truly arbitrary systems. However, our work was inspired by [16].

Besides considering composition as secure refinement, property-based composition has received interest in the literature: It considers the question whether systems that individually provide certain security properties still have these properties when they are run in parallel with other systems. For safety and liveness, general theories of this kind of compositionality exist [17–19], which are sufficient to reason about most functional system properties. However, many security properties are not safety and liveness properties, in particular confidentiality. Compositional information flow properties were first investigated in [20]. After that, much work has been devoted to identifying properties which are preserved under composition like, e.g., restrictiveness [20, 21], forward correctability [22], or separability [23]. For certain security properties that are in general not preserved under composition, it is known how to restrict composition in order to preserve these properties [23, 24]. More recent work concentrated on a uniform basis to reason about property-based composition [25, 26].

Somewhere between both notions of composition, so-called preservation theorems exist, which state that specific properties are preserved under (reactive) simulatability. Such theorems exist for integrity, fairness, liveness, and non-interference [27–32]. Moreover, relaxations of reactive simulatability have recently been proposed [33].

Finally, we mention that reactive simulatability has formed the basis for allowing cryptographically accurate, formal analyses of security protocols: If cryptographic protocols should be verified using formal methods, cryptographic primitives are abstracted based on the so-called Dolev-Yao abstraction [34], which considers cryptographic primitives, e.g., $E$ for encryption and $D$ for decryption, as operators in a free algebra where only predefined cancellation rules hold. Since this line of work turned out to be very successful, the interesting question arose whether these abstractions are indeed justified
from the view of cryptography, i.e., whether properties proved for the abstractions are still valid for the cryptographic implementation. Such cryptographic underpinnings of a Dolev-Yao model were first addressed by Abadi and Rogaway in [35] and Laud [36] for passive adversaries and symmetric encryption. A full cryptographic justification for a Dolev-Yao model, i.e., for arbitrary active attacks and within arbitrary surrounding interactive protocols, was first given in [37, 38], with extensions in [39, 40], based on the notion of reactive simulatability. Moreover, tailored tool support for this library was subsequently added [43, 44], and several well-known security protocols were proved in a computationally sound manner [45–50].

Outline. In Section 2 we review the model of reactive systems in asynchronous networks. Section 3 contains our composition theorem and its proof for black-box simulatability. In Section 4, we show that universal simulatability implies black-box simulatability and reasonable assumptions. In particular, this can be used to carry over our composition theorem for universal simulatability.

2 Asynchronous Reactive Systems

In this section, we review our model for secure reactive systems in an asynchronous network from [2]. Several definitions are only sketched whereas those that are important for understanding our results are given in full detail. All other details can be looked up in the original paper.

2.1 General System Model

Systems mainly consist of several interactive machines. Machines communicate via ports (local endpoints for different potential channels) and messages are strings over an alphabet $\Sigma$. Inspired by the CSP-Notation [51], we write output and input ports as $q!$ and $q?$ respectively. As in similar models, channels are defined implicitly by naming convention (and not by a separate graph), that is port $q!$ sends messages to $q?$. For asynchronous timing, a message is not immediately delivered to its recipient, but first stored in a special machine $\tilde{q}$ called a buffer. If a machine wants to schedule the $i$-th message of buffer $\tilde{q}$, it must have the unique clock-out port $q!$, and it sends $i$ at $q!$, see Figure 1. The buffer then outputs and deletes its $i$-th message. For a port $p$, we write $p^c$ to denote the port which it connects to according to Figure 1, i.e., $q!^c = q?^c$, $q?^c = q!^c$, $q!^c = q$ and vice versa. The in- and output ports in a port set or port sequence $P$ are denoted $\text{in}(P)$ and $\text{out}(P)$.

Our primary machine model is probabilistic state-transition machines, similar to probabilistic I/O automata as in Lynch [52] (and also essentially in [8, 9]). If a machine is switched, it receives an input tuple at its input ports and performs its transition function. This yields a new state and an output tuple in the deterministic case, or a finite

---

In more recent work, drawing upon insights gained from the proof of the cryptographic library, we showed that widely considered symbolic abstractions of hash functions and of the XOR operation cannot be proven computationally sound in general, hence indicating that their current symbolic representations might be overly simplistic [41, 42].
distribution over such pairs in the probabilistic case. Moreover, each machine has a function bounding the length of the considered inputs; this allows flexible time bounds independent of the environment.

Definition 1. (Machines) A machine is a tuple

\[ M = (name_M, Ports_M, States_M, \delta_M, l_M, Ini_M, Fin_M) \]

of a name \( name_M \in \Sigma^+ \), a finite sequence \( Ports_M \) of ports, a set \( States_M \subseteq \Sigma^* \) of states, a probabilistic state-transition function \( \delta_M \), a length function \( l_M : \) \( States_M \rightarrow (\mathbb{N} \cup \{\infty\}) \)\( |in(Ports_M)| \), and sets \( Ini_M, Fin_M \subseteq States_M \) of initial and final states. Its input set is \( \mathcal{I}_M := (\Sigma^*)^{\mid in(Ports_M)\mid} \); the \( i \)-th element of an input tuple denotes the input at the \( i \)-th in-port. Its output set is \( \mathcal{O}_M := (\Sigma^*)^{\mid out(Ports_M)\mid} \). The empty word, \( \epsilon \), denotes no in- or output at a port. \( \delta_M \) probabilistically maps each pair \( (s, I) \in States_M \times \mathcal{I}_M \) to a pair \( (s', O) \in States_M \times \mathcal{O}_M \).

Two restrictions apply to \( \delta_M \): Every output distribution has to be finite and if \( I = (\epsilon, \ldots, \epsilon) \), then \( \delta_M(s, I) = (s, (\epsilon, \ldots, \epsilon)) \). Inputs are ignored beyond the length bounds, i.e., \( \delta_M(s, I) = \delta_M(s, I_{\mid in(s)}) \) for all \( I \in \mathcal{I}_M \), where \( \epsilon \mid l \) for \( l \in \mathbb{N}, r \in \Sigma^* \) denotes the \( l \)-symbol prefix, and the notation is lifted to tuples. We further demand \( l_M(s) = (0, \ldots, 0) \) for every \( s \in Fin_M \).

In the text, we often write “\( M \)” for \( name_M \). The set (in contrast to the sequence) of ports of a machine \( M \) is denoted by \( \text{ports}(M) \), and similar for sets of machines.

A collection \( \hat{C} \) of machines is a set of machines with pairwise different machine names and disjoint sets of ports. The completion \( [\hat{C}] \) of a collection \( \hat{C} \) is the union of all machines of \( \hat{C} \) and the buffers needed for every channel. A port of a collection is called free if its connecting port is not in the collection. These ports will be connected to the users and the adversary. The free ports of a completion \( [\hat{C}] \) are denoted as \( \text{free}([\hat{C}]) \). A collection \( \hat{C} \) is called closed if its completion \( [\hat{C}] \) has no free ports except a special master clock-in port \( \text{clk}^\triangleleft \).

A closed collection represents a “runnable” system and a probability space of runs (sometimes called traces or executions) is defined for it. Machines switch sequentially, i.e., we have exactly one active machine \( M \) at any time. If this machine has clock out-ports, it can select the next message to be delivered by scheduling a buffer via one of these clock out-ports. If the buffer contains a message at the selected position, it delivers this message, and the receiving machine is the next active machine. If \( M \) tries to
schedule multiple messages, only one is taken, and if it schedules none or the message
does not exist, the master scheduler X becomes active. Formally, runs are defined as
follows.

Definition 2. (Runs and Views) Let \( \hat{C} \) be a closed collection with master scheduler \( X \).
Runs and their probability spaces are defined inductively by the following algorithm for each
tuple \( \text{ini} \in \times_{M \in \hat{C}} \text{Init}_M \) of initial states. The algorithm maintains variables for
the state of each machine and treats each port as a variable over \( \Sigma^* \), initialized with \( \epsilon \)
except for \( \text{clk}^? := 1 \). It further maintains a variable \( M_{CS} \) (“current scheduler”) over
machine names, initially \( M_{CS} := X \), for the currently active non-buffer machine, and a
variable \( r \) for the resulting run, an initially empty list. The algorithm has five phases. Probabilistic
choices only occur in Phase 1.

1. Switch current scheduler: Switch the current machine \( M_{CS} \), i.e., set \( (s', O) :=
   \delta_{M_{CS}}(s, I) \) for its current state \( s \) and in-port values \( I \). Then assign \( \epsilon \) to all in-ports
   of \( M_{CS} \).
2. Termination: If \( X \) is in a final state, the run stops. (As \( X \) made no outputs in this
case, this only prevents repeated master clock inputs.)
3. Store outputs: For each simple out-port \( o! \) of \( M_{CS} \) with \( o! \neq \epsilon \), in their given order,
   switch buffer \( o \) with input \( o^{++} \) := \( o! \). Then assign \( \epsilon \) to these ports \( o! \) and \( o^{++} \).
4. Clean up scheduling: If at least one clock out-port of \( M_{CS} \) has a value \( \neq \epsilon \), let \( n^? \)
denote the first such port and assign \( \epsilon \) to the others. Otherwise let \( \text{clk}^? := 1 \) and
   \( M_{CS} := X \) and go to Phase 1.
5. Deliver scheduled message: Switch buffer \( n \) with input \( n^? := n^! \), set \( n^? := n^{++} \)
   and then assign \( \epsilon \) to all ports of \( n \) and to \( n^! \). If \( n^? = \epsilon \) let \( \text{clk}^? := 1 \) and \( M_{CS} := X \).
   Else let \( M_{CS} := M' \) for the unique machine \( M' \) with \( n^? \in \text{ports}(M') \). Go to Phase 1.

Whenever a machine (this may be a buffer) \( M \) switches from \( (s, I) \) to \( (s', O) \), we add
a step \((\text{name}_M, s, I, s', O)\) to the run \( r \) with the following two restrictions. First, we
cut each input according to the respective length function, i.e., we replace \( I \) by \( I' :=
\lceil I \rceil_{\text{len}(s)} \). Secondly, we do not add the step to the run if \( I' = (\epsilon, \ldots, \epsilon) \), i.e., if nothing
happens in reality. This gives a random variable \( \text{run}_{\hat{C}, \text{ini}} \) for each tuple \( \text{ini} \in \times_{M \in \hat{C}} \)
of initial states, and similarly for \( l \)-step prefixes \( \text{run}_{\hat{C}, \text{ini}}(l) \) of the view. For a singleton \( \hat{M} := \{H\} \) we write \( \text{view}_{\hat{C}, \text{ini}}(H) \) for
\( \text{view}_{\hat{C}, \text{ini}}(\{H\}) \).

2.2 Security-specific System Model

We now define specific collections for security purposes. We start with the definition
of structures. Intuitively, these are the machines that execute a security protocol. They
have a distinguished set of service ports. This is a subset of the free ports where, in-
tuitively, a certain service is guaranteed, while remaining free ports are meant only for
the adversary. Typical examples of inputs at service ports are “send message m to participant id” for a message transmission system or “pay amount x to participant id” for a payment system, while typical non-service ports are those of insecure network connections in a real system. For cryptographic purposes, the initial state of all machines in a structure is a security parameter $k$ in unary representation.

**Definition 3.** (Structures and Service Ports) A structure is a pair $\mathbf{struc} = (\hat{M}, S)$ where $\hat{M}$ is a collection of simple machines (i.e., with only normal in- and out-ports and clock out-ports) with $\{1\}^* \subseteq \text{Ini}_M$ for all $M \in \hat{M}$, and $S \subseteq \text{free}([\hat{M}])$. The set $S$ is called service ports.

Forbidden ports for users of a structure are those that clash with port names of given machines and those that would link the user to a non-service port.

**Definition 4.** (Forbidden Ports) For a structure $(\hat{M}, S)$ let $\bar{S}_{\hat{M}} := \text{free}([\hat{M}]) \setminus S$. We call $\text{forb}(\hat{M}, S) := \text{ports}(\hat{M}) \cup \bar{S}_{\hat{M}}$ the forbidden ports.

A system is a set of structures. The idea behind systems is that there may be different actual structures depending on the set of actually malicious participants.

**Definition 5.** (Systems) A system $\mathbf{Sys}$ is a set of structures.

A structure can be complemented to a configuration by adding a user machine and an adversary machine. The user is restricted to connecting to the service ports. The adversary closes the collection, i.e., it connects to the remaining service ports, the other free ports $\bar{S}_{\hat{M}}$ of the collection, and the free ports of the user. Thus, user and adversary can interact, e.g., for modeling active attacks.

**Definition 6.** (Configurations) A configuration of a structure $(\hat{M}, S)$ is a tuple $\mathbf{conf} = (\hat{M}, S, H, A)$ where

- $H$ is a machine called user with $\text{ports}(H) \cap \text{forb}(\hat{M}, S) = \emptyset$ and $\{1\}^* \subseteq \text{Ini}_H$,
- $A$ is a machine called adversary with $\{1\}^* \subseteq \text{Ini}_A$,
- and the completion $\hat{C} := [\hat{M} \cup \{H, A\}]$ is a closed collection.

The set of configurations of $(\hat{M}, S)$ is written $\text{Conf}(\hat{M}, S)$. The notation $\text{Conf}()$ is lifted to sets of structures, i.e., systems. We write $\text{conf}\hat{.} \hat{M}$ for $\text{conf}[1]$ (component selection function) and similarly $\text{conf} . S$, $\text{conf} . H$, and $\text{conf} . A$, and $\text{conf} \cdot \text{struc}$ for $\text{conf}[1, 2]$.

### 2.3 Parametrized Systems

In many typical systems, the structures only depend on the trust model, but not on the security parameter $k$. In a parametrized system this is different. Hence such a system is partitioned into different subsystems for different values of $k$. “Normal” systems can naturally be identified with parametrized systems where all subsystems are equal.

**Definition 7.** (Parametrized Systems) A parametrized system is a system $\mathbf{Sys}$ together with a partitioning $(\mathbf{Sys}_k)_{k \in \mathbb{N}}$, i.e., the elements $\mathbf{Sys}_k$ are pairwise disjoint systems with $\mathbf{Sys} = \bigcup_{k \in \mathbb{N}} \mathbf{Sys}_k$. In slight abuse of notation we also call the sequence of partitions...
Sys, and if the system is called Sys, the notation \( \text{Sys}_k \) always refers to the \( k \)-th element in the partition sequence.

A bounding function for a parametrized system is a function \( P \) such that for all \( k \in \mathbb{N} \) and \((M, S) \in \text{Sys}_k \) we have \( |M| \leq P(k) \) and the runtime of every \( M \in \hat{M} \) on initial input \( 1^k \) is bounded by \( P(k) \) in the sense of circuit complexity (more precisely, circuit size). A parametrized system is polynomial-time if it has a polynomial bounding function.

Circuit complexity, i.e., non-uniform complexity, is natural for this definition because one can consider every machine \( M \), used only for security parameter \( k \), as a separate circuit. Meaningful uniform complexity for such a definition requires a universal machine that simulates all these structures, and a generation algorithm for structures. However, our results are reductions with concrete security (as first introduced as a general concept with special notation in [53]), and usable for a wide range of complexity measures. In those reductions we actually work with Turing complexity because it is defined in full detail for our interacting machines.

A parametrized system considers the potentially used subsystems as potentially available from the start. This is also implicitly the case in [16] because although a subsystem is said to be generated there, it springs up magically in distributed locations by this operation. This means that all the connections must be assumed to be predefined. A truly dynamic system would need to distribute port or machine names of new machines, like the \( \pi \)-calculus does. We do not see any specific reason while our theorem should not hold for this case but it would require a rigorous definition first.

We now define user and adversary of a parametrized system.

**Definition 8.** (User and Adversary of a Parametrized System) A user and an adversary of a parametrized system \( \text{Sys} \) are families \((H_{\text{struct}})_{\text{struct} \in \text{Sys}}, (A_{\text{struct}})_{\text{struct} \in \text{Sys}} \) such that \((\hat{M}, S, H_{(\hat{M}, S)}, A_{(\hat{M}, S)}) \in \text{Conf}(\text{Sys}) \) for all \((\hat{M}, S) \in \text{Sys}\).

To reason about the complexity of users and adversaries, or more generally families of machines, we define the parametrized complexity.

**Definition 9.** (Parametrized Complexity) Let \( X = \bigcup_{k \in \mathbb{N}} X_k \) be a partitioned index set (with the same conventions as for systems) and let \( A = (A_x)_{x \in X} \) be a family of machines with \( \{1\}^* \subseteq \text{Ini}_{A_x} \) for every \( x \in X \). We say that \( A \) is of complexity \( t : \mathbb{N} \rightarrow \mathbb{N} \) if for all \( x \in X_k \), the runtime of \( A_x \) on initial input \( 1^k \) is bounded by \( t(k) \) in the sense of circuit complexity. We sometimes write \( t_A \) for “the” complexity of \( A \).

### 2.4 Defining Security with Simulatability

Reactive simulatability essentially means that whatever might happen to an honest user in a real system \( \text{Sys}_1 \) can also happen in an ideal system \( \text{Sys}_2 \). More precisely, for every configuration \( \text{conf}_1 \) of \( \text{Sys}_1 \), there exists a configuration \( \text{conf}_2 \) of \( \text{Sys}_2 \) with the same user yielding indistinguishable views for this user. A typical situation is illustrated in Figure 2.

However, we do not want to compare a structure of \( \text{Sys}_1 \) with arbitrary structures of \( \text{Sys}_2 \), but only with certain suitable ones. What suitable means in a concrete situation
can be defined by a mapping \( f \) from \( \text{Sys}_1 \) to \( \text{Sys}_2 \). The mapping \( f \) is called valid if it maps structures with the same service ports, so that the same user can connect.

**Definition 10.** (Valid Mappings) A valid mapping between two systems \( \text{Sys}_1 \) and \( \text{Sys}_2 \) is a function \( f : \text{Sys}_1 \rightarrow \text{Sys}_2 \) with \( (M_2, S_2) = f((M_1, S_1)) \Rightarrow S_1 = S_2 \). We call \( f((M_1, S_1)) \) the corresponding structure of \( (M_1, S_1) \). If the systems are parametrized, we also require \( f(\text{Sys}_{1,k}) \subseteq \text{Sys}_{2,k} \) for all \( k \in \mathbb{N} \).

A technical problem for reactive simulatability is that a correct user of a structure from \( \text{Sys}_1 \) might have forbidden ports in the corresponding structure. Configurations where this does not happen are called suitable; we restrict the simulatability definition to those.

We omit a rigorous definition for brevity. For a valid mapping \( f : \text{Sys}_1 \rightarrow \text{Sys}_2 \), let \( \text{Conf}^f(\text{Sys}_1) \) be the set of suitable configurations.

We present the definition of indistinguishability for two families of random variables with a common partitioned index set and with versions for concrete security, following [54–56].

**Definition 11.** (Indistinguishability) Let two families \( (\text{var}_x)_{x \in X} \) and \( (\text{var}'_x)_{x \in X} \) of discrete probability distributions (random variables) on common domains \( (D_x)_{x \in X} \) be given with a partitioned index set \( X = \bigcup_{k \in \mathbb{N}} X_k \) (with the same conventions as for systems).

- They are called perfectly indistinguishable iff \( \text{var}_x = \text{var}'_x \) for all \( x \in X \).
- They are called statistically \( \delta \)-indistinguishable for a function \( \delta : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0} \) iff the statistical distance \( \Delta_{\text{stat}}(\text{var}_x, \text{var}'_x) := \frac{1}{2} \sum_{d \in D_x} |\Pr(\text{var}_x = d) - \Pr(\text{var}'_x = d)| \) is at most \( \delta(k) \) for all \( k \) and all \( x \in X_k \).
- An algorithm \( \text{Dis} \) is called a \( (t, \delta) \)-distinguisher for \( \text{var}_x \) and \( \text{var}'_x \) for \( t \in \mathbb{N} \), \( \delta \in \mathbb{R}_{\geq 0} \), and \( x \in X_k \) iff its complexity is at most \( t \) and
  \[
  \delta^\text{Dis}_x := |\Pr(\text{Dis}(1^k, \text{var}_x) = 1) - \Pr(\text{Dis}(1^k, \text{var}'_x) = 1)| \geq \delta.
  \]
- The distributions are called polynomially indistinguishable iff for all polynomials \( t \) and all distinguishers \( (\text{Dis}_x)_{x \in X} \) with complexity \( t \) in their first parameter, there exists a negligible function \( \delta \) such that \( \delta^\text{Dis}_x \leq \delta(k) \) for all \( k \) and all \( x \in X_k \).
We write \(\approx_y\) for indistinguishability with \(y = \text{perf}, \delta, \text{or poly}\), respectively. We write \(\approx\) if we want to treat all cases together, and we often write \(=\) for \(\approx_{\text{perf}}\).

We later need that indistinguishability of families of random variables implies indistinguishability of functions of them, e.g., of “parts” of the random variables.

**Lemma 1.** (Indistinguishability of Derived Distributions) Let \(\text{var}, \text{var}'\) be families of probability distributions with partitioned index set \(X\) and a common family of domains \(D\), and let \(\phi = (\phi_x)_{x \in X}\) be a family of functions \(\phi_x\) on \(D_x\) (to strings, but encoding domains as strings is not made explicit). Then the following holds:

- \(\text{var} \approx_y \text{var}'\) if \(y\) is perf, or a function \(\delta\).
- Every \((t, \delta)\)-distinguisher \(\text{Dis}_\phi\) for \(\phi(\text{var})\) and \(\phi(\text{var}')\) gives rise to a \((t', \delta')\)-distinguisher \(\text{Dis}\) for \(\text{var}\) and \(\text{var}'\) with \(t' = t + t_\phi(b(k))\), where \(t_\phi: \mathbb{N} \rightarrow \mathbb{N}\) denotes the complexity of \(\phi\), and \(b: \mathbb{N} \rightarrow \mathbb{N}\) bounds the length of the random variables, i.e., \(|v| \leq b(k)\) for all \(v \in D_x\) and \(x \in X_k\).
- \(\text{var} \approx_{\text{poly}} \text{var}'\) if the random variables are of polynomial length, and \(\phi\) is of polynomial complexity.

\(\square\)

This is clear for the perfect case, and can be easily shown by computations on statistical distances for the statistical case. For concrete complexity and the computational case, the distinguisher family \(\text{Dis}\) for the original distributions is defined by \(\text{Dis}_\phi(1^k, v) := \text{Dis}_\phi (1^k, \phi(v))\) for all \(k\) and \(x \in X_k\), and for \(v\) of length at most \(b(k)\).

We are now ready to define reactive simulatability for parametrized systems. We require that there exists an extension \(f_c\) of the valid structure mapping \(f\) to a configuration mapping that leaves the user unchanged, i.e., we skolemize the existence of corresponding adversaries in Figure 2. We then consider the families of user views \(\text{view}_{\text{conf}_1}(H)\) and \(\text{view}_{f_c(\text{conf}_1)}(H)\) where all machines have initial input \(1^k\) for the security parameter \(k\) to which this configuration belongs. Each of these two families contains one well-defined probability distribution for each configuration \(\text{conf}_1\). Overall these are two families of distributions with the partitioned index set \(\text{Conf}^f(\text{Sys}_1) = \bigcup_{k \in \mathbb{N}} \text{Conf}^f(\text{Sys}_{1,k})\). Similarly, we obtain two families \(\text{view}_{\text{conf}_1,l}(H)\) and \(\text{view}_{f_c(\text{conf}_1),l}(H)\) for \(l\)-step prefixes of user views.

**Definition 12.** (Reactive Simulatability) Let parametrized systems \(\text{Sys}_1\) and \(\text{Sys}_2\) with a valid mapping \(f\) be given. For reactive simulatability, we require that there exists a function \(f_c: \text{Conf}^f(\text{Sys}_1) \rightarrow \text{Conf}^f(\text{Sys}_2)\) with \(f_c(\text{conf}_1), \text{struc} = f(\text{conf}_1), \text{struc}\) and \(f_c(\text{conf}_1), \text{H} = \text{conf}_1, \text{H}\) for all \(\text{conf}_1 \in \text{Conf}^f(\text{Sys}_1)\), and with the following properties. We say that \(f_c\) is a \(\tau\)-mapping for a structure \(\text{struc}_1\) and a function \(\tau: \mathbb{N} \rightarrow \mathbb{N}\) if the complexity \(t_{f_c(\text{conf}_1), A}\) is bounded by \(\tau(t_{\text{conf}_1, A})\) for all \(\text{conf}_1 \in \text{Conf}^{f}(\text{struc}_1)\). The entire \(f_c\) is a \(\tau\)-mapping for a function \(\tau: \mathbb{N}^2 \rightarrow \mathbb{N}\) if for all \(\text{conf}_1 \in \text{Conf}^f(\text{Sys}_1)\) we have \(t_{f_c(\text{conf}_1), A} \leq \tau(t_{\text{conf}_1, A})\).

We say that \(\text{Sys}_1 \geq_y \text{Sys}_2\), spoken “\(y\)-at least as secure as”, under the following conditions for different cases of \(y\) and \(y'\), where we abbreviate \(\text{H} := \text{conf}_1, \text{H}\):

a) \(y = \text{perf}\) and \(y' = \text{“perfectly”}\) if \(\text{view}_{\text{conf}_1, l}(H)\) and \(\text{view}_{f_c(\text{conf}_1), l}(H)\) are perfectly indistinguishable for every \(\text{conf}_1 \in \text{Conf}^f(\text{Sys}_1)\).
b) \( y = \delta \) and \( y' = \text{"\(\delta\)-statistically" for a function \( \delta: \mathbb{N}^2 \rightarrow \mathbb{R}_{\geq 0} \) iff for every conf\(_1 \in \text{Conf}^f(Sys_{1,k}) \) and every \( l \in \mathbb{N} \) we have view\(_{\text{conf}_1,l}(H) \approx_{\delta(k,l)}\) view\(_{f_{\text{conf}_1,l}}(H)\).}

(c) Concrete security: An algorithm Dis is called a \((t, \delta)\)-distinguisher for conf\(_1 \in \text{Conf}^f(Sys_{1,k}) \) and \( f_C(\text{conf}_1) \) where \( t \in \mathbb{N} \) and \( \delta \in \mathbb{R}_{\geq 0} \) iff its complexity is at most \( t \) and \( \delta_{\text{conf}_1} \geq \delta \) where

\[
\delta_{\text{conf}_1} := |\Pr(\text{Dis}(1^k, \text{view}_{\text{conf}_1}(H)) = 1) - \Pr(\text{Dis}(1^k, \text{view}_{f_C(\text{conf}_1)}(H)) = 1)|.
\]

Where the difference between the types of security is irrelevant, we only write \( \geq_{\text{sec}} \) and we omit the indices \( f \) and \( \text{sec} \) if they are clear from the context.

An essential ingredient in the composition theorem and other uses of the model is a notion of combining several machines into one, and a lemma that this makes no essential difference in views. The combination is defined in a canonical way by considering a combined state space and letting each transition function operate on its respective part. We omit details for brevity. The combination of a set \( M \) of machines is written \( \text{comb}(M) \) and we sometimes write \( \text{comb}(M_1, \ldots, M_j) \) for \( \text{comb}(\{M_1, \ldots, M_j\}) \).

**Lemma 2.** (Machine Combination) Let \( \hat{C} \) be a collection without buffers, and \( \hat{D} \subseteq \hat{C} \). The view of every set of original machines in \((\hat{C} \setminus \hat{D}) \cup \{\text{comb}(\hat{D})\}\) is the same as in \( \hat{C} \). This includes the view of the submachines in \( \text{comb}(\hat{D}) \), which is well-defined given \( \hat{C} \) and \( \hat{D} \). The Turing complexity of \( \text{comb}(\hat{D}) \) is the sum of the complexities of the machines in \( \text{comb}(\hat{D}) \). \( \square \)

We can now add the notion of blackbox simulatability to Definition 12. Here \( A_2 \) is given as the combination of a fixed “simulator” Sim and a machine \( A'_1 \) that is identical to \( A_1 \) up to port renaming.

**Definition 13.** (Blackbox Simulatability) With the notation of Definition 12, blackbox simulatability means that we have functions \( f_{\text{sim}} \) from \( Sys_1 \) to machines (the simulators for the structures) and \( f_o \) from \( Sys_1 \) to port renaming functions such that for all conf\(_1 = (M_1, S, H, A_1) \in \text{Conf}^f(Sys_1) \) we have \( f_C(\text{conf}_1) = (M_2, S, H, A_2) \) with \( (M_2, S) = f((M_1, S)) \) and \( A_2 = \text{comb}(\text{Sim}, A'_1) \) with \( \text{Sim} := f_{\text{sim}}((M_1, S_1)) \) and \( A'_1 := f_o((M_1, S_1))(A_1) \). For computational security, we require that \( \text{Sim} \) is polynomial-time, i.e., that the parametrized complexity of \( (f_{\text{sim}}((M_1, S_1)))(M_1, S) \in Sys_1 \) is polynomially bounded. We write \( \geq_{\text{sec}}^{\text{bb}} \) instead of \( \geq_{\text{sec}} \) if we want to emphasize this case (with the respective superscripts). \( \diamond \)
2.5 Composition

When composing several systems, one typically does not want to compose every structure of one system with every structure of the others, but only with certain matching ones. For instance, if the individual machines of Sys$_2$ are implemented on the same physical devices as those of Sys$_1$, as usual in a layered distributed system, we only compose structures corresponding to the same set of corrupted physical devices. However, this is not the only conceivable situation. Hence we do not define a composition operator that produces one specific composition, but a set of possible compositions.

**Definition 14.** (Composability and Composition of Structures) We call structures $(\hat{M}_1, S_1), \ldots, (\hat{M}_n, S_n)$ composable if ports$(\hat{M}_i) \cap \text{forb}(\hat{M}_j, S_j) = \emptyset$ and $S_i \cap \text{free}(\hat{M}_j) = S_j \cap \text{free}(\hat{M}_i)$ for all $i \neq j$. We then define their composition as $(\hat{M}_1, S_1)\ldots|||\ldots||(\hat{M}_n, S_n) := (\hat{M}, S)$ with $\hat{M} := M_1 \cup \ldots \cup M_n$ and $S := (S_1 \cup \ldots \cup S_n) \cap \text{free}(\hat{M})$.

We now define the composition of variably many systems, i.e., there is a potentially infinite supply of systems from which a finite number $P(k)$ is chosen for composition for each security parameter $k$.

**Definition 15.** (Parametrized Composition of Systems) Let a sequence Sysseq = $(\text{Sys}^{(i)})_{i \in \mathbb{N}}$ be given where each Sys$^{(i)}$ is a parametrized system, and let $P: \mathbb{N} \rightarrow \mathbb{N}$ be a function. Then a $P$-sized composition of Sysseq is a parametrized system Sys$^*$ where for all $k \in \mathbb{N}$, every structure $(\hat{M}^*, S^*) \in \text{Sys}_k^*$ has a unique representation $(\hat{M}^*, S^*) = (\hat{M}_1, S_1)\ldots|||\ldots||(\hat{M}_{P(k)}, S_{P(k)})$ with composable structures $(\hat{M}_i, S_i) \in \text{Sys}_k^{(i)}$ for $i = 1, \ldots, P(k)$. We call $(\hat{M}_i, S_i)$ the restriction of $(\hat{M}^*, S^*)$ to Sys$^{(i)}$ and write $(\hat{M}_i, S_i) = (\hat{M}^*, S^*)|_{\text{Sys}^{(i)}}$.

If the systems Sys$^{(i)}$ have a joint bounding function $Q$, then $P \cdot Q$ is a bounding function for Sys$^*$. In particular, if $P$ and $Q$ are polynomials, then Sys$^*$ is polynomial-time.

3 General Composition Theorem for Blackbox Simulatability

In this section, we show that reactive blackbox simulatability is consistent with the composition of a parametrized number of systems, in particular polynomially many in the computational case. The basic idea is the following: Assume that we have proven that a potentially infinite supply of systems Sys$^{(i)}$ are as secure as systems Sys$^{(i)}$ in the sense of black-box simulatability. Now we want to use Sys$^{(i)}$ as a secure replacement for Sys$^{(i)}$, i.e., as an implementation of the ideal system Sys$^{(i)}$. The following theorem shows that such modular proofs are possible. The situation is shown in the upper part of Figure 3.

Additional conditions in the theorem are that all corresponding structures are composable and that, for the polynomial case, the security of the system is in certain sense uniform.

---

1 The first condition makes one structure a valid user of another. The second one excludes cases where $p \in \text{free}(\hat{M}_i) \cap \text{free}(\hat{M}_j)$ (e.g., a clock port for a connection between these structures) and $p \in S_i$ but $p \not\in S_j$. 

Theorem 1. (Secure Parametrized Composition, Blackbox Case) Let \( \text{Sysseq} = (\text{Sys}^{(i)})_{i \in \mathbb{N}} \) and \( \text{Sysseq}' = (\text{Sys}^{(i)}')_{i \in \mathbb{N}} \) be sequences of parametrized systems. Let \( f = (f^{(i)})_{i \in \mathbb{N}} \) be a sequence of valid mappings \( f^{(i)} : \text{Sys}^{(i)} \rightarrow \text{Sys}^{(i)}' \), and let \( \text{Sys}^{(i)} \geq_{bb} f^{(i)}, \text{Sys}^{(i)}' \) for all \( i \in \mathbb{N} \).

Let \( P : \mathbb{N} \rightarrow \mathbb{N} \), and let \( \text{Sys}^{#} \) and \( \text{Sys}^{*} \) denote the \( P \)-sized compositions of \( \text{Sysseq} \) and \( \text{Sysseq}' \), respectively. Assume that the following structural conditions hold for all \( k \in \mathbb{N} \) and every structure \((M^#, S) \in \text{Sys}^{#}_k \): Let its restrictions be \((\hat{M}^#, S) \in \text{Sys}^{#}_k \) and the corresponding structures \((\hat{M}'^i, S_i) := f^{(i)}((\hat{M}_i, S_i)) \) for all \( i \leq P(k) \). Then the composition

\[
f^{#}((\hat{M}^#, S)) := (\hat{M}'_1, S_1) \ | \cdots\ | (\hat{M}'_{P(k)}, S_{P(k)})
\]

exists and lies in \( \text{Sys}^*_k \). Furthermore, \((\hat{M}_i, S_i) \) and \((\hat{M}'_j, S'_j) \) must be composable for \( j \neq i \), and \( \text{ports}(\hat{M}'_i) \cap S'_j = \text{ports}(\hat{M}_i) \cap S'_j \) for all \( j \neq i \). Then we have

\[
\text{Sys}^{#} \geq_{bb} f^{#}, y \text{ Sys}^{*}
\]

a) with \( y = \text{perf} \) if \( y_i = \text{perf} \) for all \( i \in \mathbb{N} \).

b) with \( y = P(k) \cdot \delta(k, b(k)) \) if all \( y_i \) are bounded by a function \( \delta : \mathbb{N}^2 \rightarrow \mathbb{R}_{\geq 0} \), and where \( b(k) \) is the sum of the complexity of the systems, the user, and the simulators.

c) With concrete complexity: For every \( \text{conf}^{#} \in \text{Conf}^{#}(\text{Sys}^{#}_k) \), a \((t, \delta)\)-distinguisher for \( \text{conf}^{#} \) and \( f_c(\text{conf}^{#}) \) gives rise to a \((t', \delta')\)-distinguisher for \( \text{conf}^{(i)} \) and \( f_c(\text{conf}^{(i)}) \) for a \( \text{conf}^{(i)} \in \text{Conf}(\text{Sys}^{(i)}_k) \) with \( \delta' = \frac{\delta}{\text{Poly}(k)} \) and \( t' = t + b'(k) \), where \( b'(k) \) is a polynomial independent of \( t_{\text{conf}^{*}.\text{A}} \). (Details are given in the proof.)

d) with \( y = \text{poly} \) if \( y_i = \text{poly} \) for all \( i \in \mathbb{N} \) and under the following conditions: The function \( P \) is polynomially bounded, and the systems \( \text{Sys}^{(i)} \) have a joint bounding polynomial \( Q \). The complexities of the simulator families induced by the mappings \( f^{(i)}_{\text{Sim}} \) are bounded by a joint polynomial \( Q_{\text{Sim}} \). The distinguishing probabilities of the system pairs \((\text{Sys}^{(i)}, \text{Sys}^{(i)}') \) are uniformly bounded, i.e., for all polynomials \( t \) there exists a negligible function \( \delta \) such that for all distinguishers \( \text{Dis} \), all \( i, k \in \mathbb{N} \), and all \( \text{conf} = (\hat{M}_i, S_i, H, A) \in \text{Conf}^{(i)}(\text{Sys}^{(i)}_k) \) we have \( t_{\text{Dis}} \leq t(k) \land t_{H} \leq t(k) \land t_{A} \leq t(k) \Rightarrow \delta_{\text{Dis}}(\text{conf}) \leq \delta(k) \) (recall Definition 12d).

The first statement to be proved is extracted into the following lemma.

Lemma 3. Under the conditions of Theorem 1, the mapping \( f^{#} \) is a valid mapping between \( \text{Sys}^{#} \) and \( \text{Sys}^{*} \). \hfill \square
Definition 16. (Simulator and Corresponding Configurations) Under the conditions of Theorem 1 and for all $i \in \mathbb{N}$, let $f^{(i)}_{\text{Sim}}$ and $f^{(i)}_{A}$ be the simulator and renaming functions from which $f^{(i)}_{C}$ is composed by blackbox simulatability. We compose them into functions $f^{\#}_{\text{Sim}}$ and $f^{\#}_{A}$ on $\text{Sys}^{\#}$ as follows: Given $k \in \mathbb{N}$ and $(\hat{M}^{\#}, S) \in \text{Sys}_k^{\#}$, let $\text{Sim}_i := f^{(i)}_{\text{Sim}}((\hat{M}_i, S_i))$ for all $i \leq P(k)$, and let

$$f^{\#}_{\text{Sim}}((\hat{M}^{\#}, S)) := \text{comb}(\text{Sim}_1, \ldots, \text{Sim}_{P(k)});$$

further let $f^{\#}_{A} := f^{(P(k))}_{A} \circ \cdots \circ f^{(1)}_{A}$. Let $f^{\#}_{C}$ be constructed from $f^{\#}$, $f^{\#}_{\text{Sim}}$, and $f^{\#}_{A}$ by the equations in Definition 13 (blackbox simulatability).

The complexity $t_{\text{Sim}}$ of the simulator is $t_{\text{Sim}}(k) = \sum_{i=1}^{P(k)} t_{\text{Sim}_i}(k)$ by Lemma 2. In the polynomial case, there exists a polynomial $Q_{\text{Sim}}$ such that $t_{\text{Sim}_i} \leq Q_{\text{Sim}}$ for all $i$, hence $t_{\text{Sim}}(k)$ is polynomially bounded by $P(k) \cdot Q_{\text{Sim}}(k)$.

We also omit the technical proof that indeed $f^{\#}_{C} : \text{Conf}^{\#}(\text{Sys}^{\#}) \to \text{Conf}(\text{Sys}^{\ast})$ in Definition 16. It is nevertheless interesting that these proof parts that verify the compatibility of channels and the difference of service ports and adversary ports in compositions make up the major part of a rigorous proof, while the cryptographic aspects are shorter and more standard.

Now we can concentrate on proving that the simulator simulates correctly. The proof consists of a hybrid argument as first used in [5], i.e., we construct intermediate configurations that differ only in the machines of one system.

Fig. 3. Configurations in the composition theorem for blackbox simulatability.
Proof (Theorem 1). Let a configuration $\text{conf}^\# = (\hat{M}^\#, S, H, A) \in \text{Conf}^\# (\text{Sys}^\#)$ be given and $\text{conf}^* := f^*_C (\text{conf}^\#)$ the corresponding configuration according to Definition 16. Let the sub-structures $(\hat{M}_i, S_i)$ and $(\hat{M}'_i, S'_i)$, the simulators $\text{Sim}_i$, and functions $f^*_i$ with various indices be defined as in the formulation of the theorem and Definition 16. Furthermore, let $(\hat{M}^*, S) := f^*((\hat{M}^\#, S))$ and $\text{Sim} := f^*_{\text{Sim}}((\hat{M}^\#, S))$. Then we have $\text{conf}^* = (\hat{M}^*, S, H, \text{comb}(\text{Sim}, f^*_A (A)))$; recall that $f^*_\text{A}$ is just a port renaming; hence Figure 3 simplifies it to $A$.

The outline of the hybrid argument is as follows.

1. We define hybrid configurations $\text{conf}_i^{hyb}$ of $\text{Sys}^{(i)}$ and $\text{conf}_i^{hyb}$ of $\text{Sys}^{(i)}$ for $i = 1, \ldots, P(k)$. In $\text{conf}_i^{hyb}$ the first $i-1$ real structures have already been replaced with their ideal counterparts, while in $\text{conf}_i^{hyb}$ also the $i$-th structure has been replaced. To make these configurations correct configurations of the respective systems, all other machines are grouped into an overall hybrid user $H_i^{hyb}$ as shown at the bottom of Figure 3 for $i = 2$ and $P(k) = 3$.

2. We show that these are correct and corresponding configurations with respect to the given blackbox simulatability between $\text{Sys}^{(i)}$ and $\text{Sys}^{(i)}$.

3. We show that the views of $H$ in $\text{conf}_i^{hyb}$ and $\text{conf}_i^{hyb}$ are equal for $i = 1, \ldots, P(k) - 1$. Moreover, we show that the views of $H$ are equal in $\text{conf}^\#$ and $\text{conf}_i^{hyb}$, and in $\text{conf}_i^{hyb}$ and $\text{conf}^*$. This gives a kind of indistinguishability chain (for one configuration)

$$view_{\text{conf}^\#}(H) \approx view_{\text{conf}_i^{hyb}}(H) \approx \cdots \approx view_{\text{conf}_{P(k)}^{hyb}}(H) \approx view_{\text{conf}^*}(H).$$

4. We show that this implies indistinguishability between first and last elements.

We now explain these steps in more detail.

Step 1: For $i = 1, \ldots, P(k)$, let the machine collection for the $i$-th hybrid user be $\hat{H}_i := \{H\} \cup \bigcup_{1 \leq j < i} \hat{M}'_j \cup \{\text{Sim}_j | 1 \leq j < i\} \cup \bigcup_{i < j \leq P(k)} \hat{M}_j$, and let $H_i^{hyb} := \text{comb}(\hat{H}_i)$. Furthermore let $A_i := f^{(i-1)}_A \circ \cdots \circ f^{(1)}_A (A)$ and $A'_i := f^{(i)}_A (A_i)$. Then we define the hybrid configurations as

$$\text{conf}_i^{hyb} := (\hat{M}_i, S_i, H_i^{hyb}, A_i);$$

$$\text{conf}_i^{hyb} := (\hat{M}'_i, S_i, H_i^{hyb}, \text{comb}(\text{Sim}_i, A'_i)).$$

For the computational case, we have to show that the family of $H_i^{hyb}$ is polynomial-time. This holds since $t_{H^{hyb}} \leq t_H + t_{\text{Sim}} + t_{\hat{M}^\#} + t_{\hat{M}'}$, by Lemma 2, where each addend is polynomially bounded by assumption.

Step 2: We have to show that $\text{conf}_i^{hyb} \in \text{Conf}_i^{(\text{Sys}_i)}$ and $\text{conf}_i^{hyb} \in \text{Conf}(\text{Sys}'_i)$, i.e., essentially that the hybrid users do not use non-service ports. In this short version, we omit this proof. Then the definition of $\text{conf}_i^{hyb}$ and $\text{conf}_i^{hyb}$ immediately implies

$$\text{conf}_i^{hyb} = f^{(i)}_C (\text{conf}_i^{hyb}),$$

(1)
i.e., these are indistinguishable configurations under the given blackbox simulatability between $Sys^{(i)}$ and $\hat{Sys}^{(i)}$.

**Step 3:** The configurations $conf^{hyb}_i$ and $conf^{hyb}_{i+1}$ consist of the same collection of machines $\hat{C}_i := \hat{H}_i \cup \{ M'_i, \text{Sim}_i, A'_i \}$. Combining them in different ways does not alter the view of $H$ by Lemma 2. Thus we have

$$view_{conf^{hyb}_i}(H) = view_{conf^{hyb}_{i+1}}(H)$$

for all $i \in \{ 1, \ldots, P(k) \}$, and similarly

$$view_{conf^{#}_i}(H) = view_{conf^{#}_{i+1}}(H) \land view_{conf^{#}_{P(k)}}(H) = view_{conf^*}(H).$$

**Step 4:** We now distinguish the type of the given simulatability relations $Sys^{(i)} \succ_{bb, f^{(i)}, \nu} Sys^{(i)}$.

For perfect simulatability, Equation (1) gives us $view_{conf^{hyb}_i}(H) = view_{conf^{hyb}_{i+1}}(H)$ for all $i$. With Equations (2) and (3) this yields $view_{conf^{#}_i}(H) = view_{conf^*}(H)$. This result for an arbitrary fixed configuration $conf^{#}$ implies equality of all families of such views.

For statistical simulatability, let $Sys^{(i)}$ be $\delta_i$-statistically at least as secure as $Sys^{(i)}$. Let $l \in \mathbb{N}$. For prefixes of length $l$ and $v$ ranging over the potential views of this length, we abbreviate $q^{\#}_v := \Pr(view_{conf^{#}, l}(H) = v)$, and $q^*_v := \Pr(view_{conf^*, l}(H) = v)$, and $q_{l,v} := \Pr(view_{conf^{hyb}, l}(H) = v)$ and $q'_{l,v} := \Pr(view_{conf^{#}, l}(H) = v)$ for all $i$.

For all potential views $v$, we have $q'_{l,v} = q_{l+1,v}$ and $q^*_v = q_{1,v}$ and $q'_{P(k),v} = q^*_v$ by Equations (2) and (3). The desired statistical distance is

$$\delta_{stat}(conf^{#}) := \frac{1}{2} \sum_v |q^{\#}_v - q^*_v|$$

$$= \frac{1}{2} \sum_v |q_{1,v} - q_{2,v} + q_{2,v} - q_{3,v} + \cdots + q_{P(k),v} - q'_{P(k),v}|$$

$$\leq \frac{1}{2} \sum_v (|q_{1,v} - q_{2,v}| + |q_{2,v} - q_{3,v}| + \cdots + |q_{P(k),v} - q'_{P(k),v}|)$$

$$= \sum_{i=1}^{P(k)} \frac{1}{2} \sum_v |q_{i,v} - q'_{i,v}|$$

$$= \sum_{i=1}^{P(k)} \Delta_{stat}(view_{conf^{hyb}_i}(H), view_{conf^{#}_i}(H)).$$

With Lemma 1 this gives

$$\delta_{stat}(conf^{#}) \leq \sum_{i=1}^{P(k)} \Delta_{stat}(view_{conf^{hyb}_i}(H_i^{hyb}), view_{conf^{#}_i}(H_i^{hyb})) \leq \sum_{i=1}^{P(k)} \delta(k, l_i),$$
where the $l_i$ are sufficiently large numbers to ensure that the $i$-step prefix of the view of $H$ in $conf_{hyb}^i$ is a subsequence of the $l_i$-step prefix of the view of $H_{1i}^{hyb}$. A general bound is the complexity of $H_{1i}^{hyb}$, which is bounded by $b := t_H + t_{M^*} + t_{M^*} + t_{Sim}$. This implies $\delta_{stat}(conf^{\#}) \leq P(k) \cdot \delta(k, b(k))$ as desired.

For concrete complexity and for a $(t, \Delta_{Dis})$-distinguisher $Dis$, we have by definition
\[
\Delta_{Dis} \leq |Pr(Dis(1^k, \text{view}_{conf^{\#}}(H)) = 1) - Pr(Dis(1^k, \text{view}_{conf^{\#}}(H)) = 1)|.
\]
We abbreviate $q^{\#} := Pr(Dis(1^k, \text{view}_{conf^{\#}}(H)) = 1)$ and $q^* := Pr(Dis(1^k, \text{view}_{conf^{\#}}(H)) = 1)$ and $q_i := Pr(Dis(1^k, \text{view}_{conf^{\#}}(H)) = 1)$ and $q_i' := Pr(Dis(1^k, \text{view}_{conf^{\#}}(H)) = 1)$ for all $i$, and $\Delta_i := |q_i - q_i'|$. Now Equations (2) and (3) yield
\[
\Delta_{Dis} = |q^{\#} - q^*| = |q_1 - q_2 + q_2 - q_3 + q_3 + \cdots + q_{P(k)} - q_{P(k)}'|
\]
\[
\leq |q_1 - q_2| + |q_2 - q_3| + \cdots + |q_{P(k)} - q_{P(k)}'| = \Delta_1 + \Delta_2 + \cdots + \Delta_{P(k)}.
\]
This implies that there exists some $i$ with $\Delta_i \geq \frac{\Delta_{Dis}}{P(k)}$.

We can now consider $Dis$ as a $(t, \Delta_i)$-distinguisher $Dis_{\phi}^{(i)}$ of a function $\phi$ of views of the actual user $H_{1i}^{hyb}$ of the $i$-th hybrid systems. Here $\phi$ is defined by $\phi(v) := v \mid_H$, i.e., the restriction to the view of $H$. The complexity $t_{\phi}$ of $\phi$ is linear. Hence Lemma 1 implies that there exists a $(t_i, \Delta_i)$-distinguisher $Dis_{\phi}^{(i)}$ for $\text{view}_{conf^{\#}}(H_{1i}^{hyb})$ and $\text{view}_{conf^{\#}}(H_{1i}^{hyb})$ with $t_i = t + b'(k)$, where $b'(k)$ bounds the length of the views of $H_{1i}^{hyb}$. The complexity $t_{\phi^{(i)}}$ of $H_{1i}^{hyb}$ is bounded by $b := t_H + t_{M^*} + t_{M^*} + t_{Sim}$, and above we showed $t_{Sim} \leq P \cdot Q_{Sim}$. The length of runs and thus views in our current representation is bounded by the square of this complexity (but this might be improvable by tighter encoding). This yields the desired polynomial bound $b'(k)$ independent of the adversary complexity.

For polynomial simulatability, let $H, A$ be a user and an adversary for $Sys^{\#}$ of complexity $t_H$ and $t_A$, and let $t$ be a polynomial and $Dis$ a distinguisher family of complexity $t$. Then the functions $t_{\phi^{(i)}}, t_i$, and $t_A = t_A$ are polynomials. By assumption, there exists a negligible function $\delta$ that uniformly bounds the advantage of distinguishers for the given system pairs for the complexity function $\max(t_i, t_{\phi^{(i)}}, t_A)$. Now let a configuration $conf^{\#} = (M^*, S, H_{(M^*, S)}, A_{(M^*, S)})$ be given. The concrete security considerations and Equation (1) imply $\Delta_i = \delta_{Dis_{\phi}^{(i)}} \leq \delta(k)$, and therefore $\delta_{Dis_{conf^{\#}}} \leq P(k) \cdot \delta(k)$ is negligible. This proves the desired polynomial indistinguishability of the families of user views over $Sys^{\#}$ and $Sys^{\#}$.

### 4 From Black-Box to Universal Simulatability

We now show a relation between universal simulatability and black-box simulatability. It allows us to apply our general composition theorem to universal simulatability under reasonable assumptions, but it also is of independent interest. More precisely, we show
that universal simulatability for two parametrized systems $Sys_1$ and $Sys_2$ is equivalent to black-box simulatability if $Sys_1$ fulfills the following structural requirements: Whenever a clock-out port of a structure $(M_1, S_1) \in Sys_1$ is contained in $S_1^r$, then so is either the corresponding input or output port. This means that the adversary is not allowed to schedule messages of a connection where it is neither the sender nor the recipient. This condition is naturally fulfilled for insecure channels, since the adversary is inserted between the connections of two machines of the system.

**Theorem 2. (Relating Black-box and Universal Simulatability)** Let $Sys_1, Sys_2$ be two parametrized systems with a valid mapping $f$, where for every structure $(M_1, S_1) \in Sys_1$, we have $p^! \in S_1^r \Rightarrow (p^! \in S_1^c \vee p^! \in S_1^f)$. Then $Sys_1 \succeq_{sec}^{bb,f,y} Sys_2$ iff $Sys_1 \succeq_{uni}^{f,y} Sys_2$ for $y = perf$ or a function $\delta$ and also for $y = poly$ if $Sys_1$ is polynomial-time.

For concrete security, if $\succeq_{uni}^{f}$ is given with a $\tau$-mapping $f_C$, then we obtain $\succeq_{sec}^{bb,f}$ with simulator complexity $\tau(t_{Sys_1})$, and a $(t, \delta)$-distinguisher for the views in the black-box case gives rise to a $(t', \delta)$-distinguisher for the views in the universal case where $t'$ is the sum of $t$ and the view length of $H$ and $A$.

**Proof.** The left-to-right direction is clear by definition. The difficult direction is to show that universal simulatability implies black-box simulatability. Due to lack of space, we can only present a short sketch. This direction essentially consists of four steps:

1. Let a configuration $conf_1 = (\hat{M}_1, S, H, A_1)$ of the sub-system $Sys_{1,k}$ be given. We first derive another configuration $conf_{uni}^1 = (\hat{M}_1, S, H^{uni}, A_1')$ of $Sys_2$ as follows: We insert a machine $TS_{P,b,k}$, called transparent scheduler, into the connections between $A_1$ and the simple ports in $S_1$. It forwards messages between machines of the structure and the adversary. Its parameters $P$ and $b$ correspond to the ports that the transparent scheduler connects to and a bound on its runtime, which is the joint runtime of the machines in $\hat{M}_1$. This machine only depends on $\hat{M}_1$, $S$, and $k$. The new user is the combination $H^{uni} := comb(H, A_1)$, and the new adversary is $A_1' := TS_{P,b,k}$. We show that the views of both $H$ and $A_1$ are identical in the two configurations.

2. We now show that $conf_{uni}^1 \in Conf^f(Sys_1)$ and apply the precondition $Sys_1 \succeq_{uni}^{f} Sys_2$. This yields an indistinguishable configuration $conf_{uni}^2$ of $Sys_2$ with a new adversary $A_2$. By the definition of universal simulatability, $A_2$ only depends on $\hat{M}_1$, $S$ and on $A_1' = TS_{P,b,k}$. Since $TS_{P,b,k}$ only depends on $\hat{M}_1$ and $S$, the adversary $A_2$ also only depends on $\hat{M}_1$ and $S$.

3. We obtain a configuration $conf_2$ with the original user and a simulator from $conf_{uni}^2$ by reversing the combination of $H$ and $A_1$ into $H^{uni}$, and by defining the simulator as $Sim := A_2$. We show that this does not affect the view of $H$.

4. Combining several equalities between views of $H$ in different configurations and one indistinguishability gives the same class of indistinguishability.

Summarized statements follow from this treatment per configuration, i.e., with concrete security (although details are omitted here), as usual.  

$\blacksquare$
Acknowledgments

We thank Anupam Datta, Dennis Hofheinz, Ralf Kästers, John Mitchell, Jörn Müller-Quade, Dusko Pavlovic and Rainer Steinwandt for interesting discussions.

References


