A General Composition Theorem for Secure Reactive Systems

Michael Backes, Birgit Pfitzmann, and Michael Waidner

IBM Zurich Research Lab
{mbc,bpf,wmi}@zurich.ibm.com

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 Pfitzmann and Waidner [31] is suitable for all these requirements and we use it as a rigorous foundation of this work.

The model of [31] 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 as follows, when applied to the relation between a real and an ideal system:[1] Everything that can happen to users of the real system in the presence of an arbitrary adversary A

[1] Other terms are implementation and specification, or in special cases cryptographic and abstract system.

M. Naor (Ed.): First Theory of Cryptography Conference (TCC 2004), LCNS 2951, pages 336 - 354, February 2004.

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 [31]. 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 [30] for a synchronous reactive model) holds for all variants of simulatability (general, universal, and blackbox), 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 [15]. 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 [33] and defined (with different degrees of generality and rigorosity) in [14, 6, 27, 9].

While composition theorems for special cases were proven in [6, 27], the first general composition theorem for non-reactive simulatability was proven in [9].

An important step towards compositionality results of reactive systems was taken in [19, 20], where the cryptographic security of specific systems was directly defined and verified using a formal language, the π-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 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, 29, 9]) in [16]. 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 [30] for a general synchronous reactive model, followed by essentially the same composition theorem [31] in the corresponding asynchronous model. Later than [31] but independently, another model of asynchronous reactive systems together with a composition theorem for reactive simulatability was developed in [10]. The theorem is specific for universal simulatability, but for this case it is more general than the ones in [30, 31] 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 [10].

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 [28, 32, 1], 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 [23]. After that, much work has been devoted to identifying properties which are preserved under composition like, e.g., restrictiveness [23, 24], forward correctability [18], or separability [25]. 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 [25, 26]. More recent work concentrated on a uniform basis to reason about property-based composition [22, 11].

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 [2], transitive and non-transitive non-interference [3, 4], i.e., absence of information flow, and a class of liveness properties [5].

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 [31]. 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 Σ. Inspired by the CSP-Notation [17], we write output and input ports as $\mathsf{q}!$ and $\mathsf{q}?$ respectively. As in similar models, channels are defined implicitly by naming convention (and not by a separate graph), that is port $\mathsf{q}!$ sends messages to $\mathsf{q}?$. For asynchronous timing, a message is not immediately delivered to its recipient, but first stored in a special machine $\widetilde{\mathsf{q}}$ called a buffer. If a machine wants to schedule the i-th message of buffer $\widetilde{\mathsf{q}}$, it must have the unique clock-out port $\mathsf{q}^{\triangleleft}!$, and it sends i at $\mathsf{q}^{\triangleleft}!$, 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., $\mathsf{q}!^c = \mathsf{q}^{\leftrightarrow}?$, $\mathsf{q}^{\leftrightarrow}!^c = \mathsf{q}?$, $\mathsf{q}^{\triangleleft}!^c = \mathsf{q}^{\triangleleft}?$ and vice versa. The in- and output ports in a port set or port sequence P are denoted $\mathsf{in}(P)$ and $\mathsf{out}(P)$.

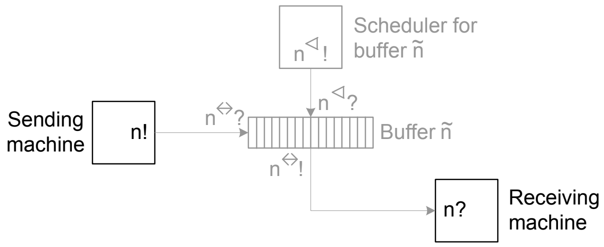

Fig. 1. Ports and buffers. Specifications only need to spell out the black part.

Our primary machine model is probabilistic state-transition machines, similar to probabilistic I/O automata as in Lynch [21] (and also essentially in [6, 27]). 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 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*

$$\mathsf{M} = (name_{\mathsf{M}}, Ports_{\mathsf{M}}, States_{\mathsf{M}}, \delta_{\mathsf{M}}, l_{\mathsf{M}}, Ini_{\mathsf{M}}, Fin_{\mathsf{M}})$$

of a name $name_{\mathsf{M}} \in \Sigma^+$, a finite sequence $Ports_{\mathsf{M}}$ of ports, a set $States_{\mathsf{M}} \subseteq \Sigma^*$ of states, a probabilistic state-transition function δ_{M}, a length function $l_{\mathsf{M}} : States_{\mathsf{M}} \to (\mathbb{N} \cup \{\infty\})^{|\mathsf{in}(Ports_{\mathsf{M}})|}$, and sets $Ini_{\mathsf{M}}, Fin_{\mathsf{M}} \subseteq States_{\mathsf{M}}$ of initial and final states. Its input set is $\mathcal{I}_{\mathsf{M}} := (\Sigma^*)^{|\mathsf{in}(Ports_{\mathsf{M}})|}$; the i-th element of an input tuple denotes the input at the i-th in-port. Its output set is $\mathcal{O}_{\mathsf{M}} := (\Sigma^*)^{|\mathsf{out}(Ports_{\mathsf{M}})|}$. The empty word, ϵ, denotes no in- or output at a port. δ_{M} probabilistically maps each pair $(s, I) \in States_{\mathsf{M}} \times \mathcal{I}_{\mathsf{M}}$ to a pair $(s', O) \in States_{\mathsf{M}} \times \mathcal{O}_{\mathsf{M}}$.

Two restrictions apply to δ_{M}: Every output distribution has to be finite and if $I = (\epsilon, \ldots, \epsilon)$, then $\delta_{\mathsf{M}}(s, I) = (s, (\epsilon, \ldots, \epsilon))$. Inputs are ignored beyond the length bounds, i.e., $\delta_{\mathsf{M}}(s, I) = \delta_{\mathsf{M}}(s, I\lceil_{l_{\mathsf{M}}(s)})$ for all $I \in \mathcal{I}_{\mathsf{M}}$, where $r\lceil_l$ for $l \in \mathbb{N}, r \in \Sigma^*$ denotes the l-symbol prefix, and the notation is lifted to tuples. We further demand $l_{\mathsf{M}}(s) = (0, \ldots, 0)$ for every $s \in Fin_{\mathsf{M}}$. ◇

In the text, we often write "M" for $name_{\mathsf{M}}$. The set (in contrast to the sequence) of ports of a machine M is denoted by 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 $\mathsf{free}([\hat{C}])$. A collection $\hat{C}$ is called *closed* if its completion $[\hat{C}]$ has no free ports except a special master clock-in port $\mathsf{clk}^{\triangleleft}?$.

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 $ini \in \times_{\mathsf{M} \in \hat{C}} Ini_{\mathsf{M}}$ of initial states. The algorithm maintains variables for the state of each machine and treats each port as a variable over Σ^*, initialized with ϵ except for $\mathsf{clk}^{\triangleleft}? := 1$. It further maintains a variable $\mathsf{M_{CS}}$ ("current scheduler") over machine names, initially $\mathsf{M_{CS}} := \mathsf{X}$, 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 $\mathsf{M_{CS}}$, i.e., set $(s', O) \leftarrow \delta_{\mathsf{M_{CS}}}(s, I)$ for its current state s and in-port values I. Then assign ϵ to all in-ports of $\mathsf{M_{CS}}$.*
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 $\mathsf{o}!$ of $\mathsf{M_{CS}}$ with $\mathsf{o}! \neq \epsilon$, in their given order, switch buffer $\widetilde{\mathsf{o}}$ with input $\mathsf{o}^{\leftrightarrow}? := \mathsf{o}!$. Then assign ϵ to these ports $\mathsf{o}!$ and $\mathsf{o}^{\leftrightarrow}?$.*
4. Clean up scheduling: *If at least one clock out-port of $\mathsf{M_{CS}}$ has a value $\neq \epsilon$, let $\mathsf{n}^{\triangleleft}!$ denote the first such port and assign ϵ to the others. Otherwise let $\mathsf{clk}^{\triangleleft}? := 1$ and $\mathsf{M_{CS}} := \mathsf{X}$ and go to Phase 1.*
5. Deliver scheduled message: *Switch buffer $\widetilde{\mathsf{n}}$ with input $\mathsf{n}^{\triangleleft}? := \mathsf{n}^{\triangleleft}!$, set $\mathsf{n}? := \mathsf{n}^{\leftrightarrow}!$ and then assign ϵ to all ports of $\widetilde{\mathsf{n}}$ and to $\mathsf{n}^{\triangleleft}!$. If $\mathsf{n}? = \epsilon$ let $\mathsf{clk}^{\triangleleft}? := 1$ and $\mathsf{M_{CS}} := \mathsf{X}$. Else let $\mathsf{M_{CS}} := \mathsf{M}'$ for the unique machine M' with $\mathsf{n}? \in \mathsf{ports}(\mathsf{M}')$. Go to Phase 1.*

Whenever a machine (this may be a buffer) M *switches from* (s, I) *to* (s', O)*, we add a* step $(name_{\mathsf{M}}, s, I, s', O)$ *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* $I' := I\lceil_{l_{\mathsf{M}}(s)}$*. 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* $run_{\hat{C},ini}$ *for each tuple* $ini \in \times_{\mathsf{M}\in\hat{C}}$ *of initial states, and similarly for* l*-step prefixes* $run_{\hat{C},ini,l}$*.*

The view *of a subset* $\hat{M} \subseteq \hat{C}$ *of machines in a run* r *is the subsequence of* r *consisting of those steps where a machine of* $\hat{M}$ *switches. This gives a random variable* $view_{\hat{C},ini}(\hat{M})$ *for each tuple* ini *of initial states, and similarly for* l*-step prefixes* $view_{\hat{C},ini,l}(\hat{M})$ *of the view. For a singleton* $\hat{M} = \{\mathsf{H}\}$ *we write* $view_{\hat{C},ini}(\mathsf{H})$ *for* $view_{\hat{C},ini}(\{\mathsf{H}\})$*.* ◇

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, intuitively, 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* $struc = (\hat{M}, S)$ *where* $\hat{M}$ *is a collection of simple machines (i.e., with only normal in- and out-ports and clock out-ports) with* $\{1\}^* \subseteq Ini_{\mathsf{M}}$ *for all* $\mathsf{M} \in \hat{M}$*, and* $S \subseteq \mathsf{free}([\hat{M}])$*. 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* $\bar{S}_{\hat{M}} := \mathsf{free}([\hat{M}]) \setminus S$*. We call* $\mathsf{forb}(\hat{M}, S) := \mathsf{ports}(\hat{M}) \cup \bar{S}^c_{\hat{M}}$ *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 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* $conf = (\hat{M}, S, \mathsf{H}, \mathsf{A})$ *where*

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

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

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* Sys *together with a partitioning* $(Sys_k)_{k\in\mathbb{N}}$, *i.e., the elements* Sys_k *are pairwise disjoint systems with* $Sys = \bigcup_{k\in\mathbb{N}} Sys_k$. *In slight abuse of notation we also call the sequence of partitions* Sys, *and if the system is called* Sys, *the notation* 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* $(\hat{M}, S) \in Sys_k$ *we have* $|\hat{M}| \leq P(k)$ *and the runtime of every* $\mathsf{M} \in \hat{M}$ *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.* $\diamond$

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 [8]), 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.

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* Sys *are families* $(\mathsf{H}_{struc})_{struc\in Sys}$, $(\mathsf{A}_{struc})_{struc\in Sys}$ *such that* $(\hat{M}, S, \mathsf{H}_{(\hat{M},S)}, \mathsf{A}_{(\hat{M},S)}) \in \mathsf{Conf}(Sys)$ *for all* $(\hat{M}, S) \in Sys$. $\diamond$

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* $\{1\}^* \subseteq Ini_{A_x}$ *for every* $x \in X$. *We say that* A *is of* complexity $t\colon \mathbb{N} \to \mathbb{N}$ *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. $\diamond$

2.4 Defining Security with Simulatability

Reactive simulatability essentially means that whatever might happen to an honest user in a real system Sys_1 can also happen in an ideal system Sys_2. More precisely, for every configuration $conf_1$ of Sys_1, there exists a configuration $conf_2$ of Sys_2 with the same user yielding *indistinguishable views* for this user. A typical situation is illustrated in Figure 2.

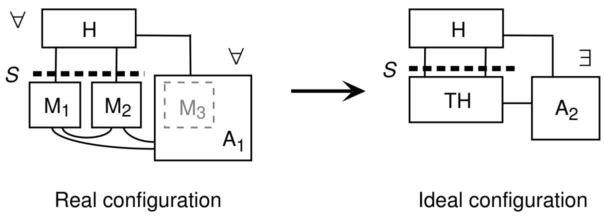

Fig. 2. Example of simulatability. The view of H is compared.

However, we do not want to compare a structure of Sys_1 with arbitrary structures of Sys_2, but only with certain suitable ones. What suitable means in a concrete situation can be defined by a mapping f from Sys_1 to 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 Sys_1 and Sys_2 is a function $f\colon Sys_1 \rightarrow Sys_2$ with $(\hat{M}_2, S_2) = f((\hat{M}_1, S_1)) \Rightarrow S_1 = S_2$. We call $f((\hat{M}_1, S_1))$ the* corresponding structure *of $(\hat{M}_1, S_1)$. If the systems are parametrized, we also require $f(Sys_{1,k}) \subseteq Sys_{2,k}$ for all $k \in \mathbb{N}$.* $\diamond$

A technical problem for reactive simulatability is that a correct user of a structure from 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 $f\colon Sys_1 \rightarrow Sys_2$, let $\mathsf{Conf}^f(Sys_1)$ 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 [34, 7, 12].

Definition 11. *(Indistinguishability) Let two families $(\mathsf{var}_x)_{x\in X}$ and $(\mathsf{var}'_x)_{x\in X}$ 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 $\mathsf{var}_x = \mathsf{var}'_x$ for all $x \in X$.*
- *They are called* statistically δ-indistinguishable *for a function $\delta\colon \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ iff the statistical distance $\Delta_{\mathsf{stat}}(\mathsf{var}_x, \mathsf{var}'_x) := \frac{1}{2}\sum_{d\in D_x} |\mathsf{Pr}(\mathsf{var}_x = d) - \mathsf{Pr}(\mathsf{var}'_x = d)|$ is at most $\delta(k)$ for all k and all $x \in X_k$.*

– *An algorithm* Dis *is called a* (t, δ)-distinguisher *for* var_x *and* 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*

$$\delta_x^{\mathsf{Dis}} := |\mathsf{Pr}(\mathsf{Dis}(1^k, \mathsf{var}_x) = 1) - \mathsf{Pr}(\mathsf{Dis}(1^k, \mathsf{var}'_x) = 1)| \geq \delta.$$

– *The distributions are called* polynomially indistinguishable *iff for all polynomials* t *and all distinguishers* $(\mathsf{Dis}_x)_{x \in X}$ *with complexity* t *in their first parameter, there exists a negligible function* δ *such that* $\delta_x^{\mathsf{Dis}} \leq \delta(k)$ *for all* k *and all* $x \in X_k$.

◇

We write "$\approx_y$" for indistinguishability with $y = \mathsf{perf}$, δ, or poly, respectively. We write "$\approx$" if we want to treat all cases together, and we often write "$=$" for "$\approx_{\mathsf{perf}}$".

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 2. *(Indistinguishability of Derived Distributions) Let* var, 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* ϕ_x *on* D_x *(to strings, but encoding domains as strings is not made explicit). Then the following holds:*

– $\mathsf{var} \approx_y \mathsf{var}' \Rightarrow \phi(\mathsf{var}) \approx_y \phi(\mathsf{var}')$ *if* y *is* perf, *or a function* δ.
– *Every* (t, δ)-*distinguisher* Dis_ϕ *for* $\phi(\mathsf{var}_x)$ *and* $\phi(\mathsf{var}'_x)$ *gives rise to a* (t', δ)-*distinguisher* Dis *for* $\mathsf{var_x}$ *and* var'_x *with* $t' = t + t_\phi(b(k))$, *where* $t_\phi \colon \mathbb{N} \to \mathbb{N}$ *denotes the complexity of* ϕ, *and* $b \colon \mathbb{N} \to \mathbb{N}$ *bounds the length of the random variables, i.e.,* $|v| \leq b(k)$ *for all* $v \in D_x$ *and* $x \in X_k$.
– $\mathsf{var} \approx_{\mathsf{poly}} \mathsf{var}' \Rightarrow \phi(\mathsf{var}) \approx_{\mathsf{poly}} \phi(\mathsf{var}')$ *if the random variables are of polynomial length, and* ϕ *is of polynomial complexity.*

□

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 Dis for the original distributions is defined by $\mathsf{Dis}_x(1^k, v) := \mathsf{Dis}_{\phi,x}(1^k, \phi(v))$ 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 $\mathit{view}_{\mathit{conf}_1}(\mathsf{H})$ and $\mathit{view}_{f_{\mathsf{C}}(\mathit{conf}_1)}(\mathsf{H})$ 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 conf_1. Overall these are two families of distributions with the partitioned index set $\mathsf{Conf}^f(\mathit{Sys}_1) = \bigcup_{k \in \mathbb{N}} \mathsf{Conf}^f(\mathit{Sys}_{1,k})$. Similarly, we obtain two families $\mathit{view}_{\mathit{conf}_1,l}(\mathsf{H})$ and $\mathit{view}_{f_{\mathsf{C}}(\mathit{conf}_1),l}(\mathsf{H})$ for l-step prefixes of user views.

Definition 12. *(Reactive Simulatability) Let parametrized systems* Sys_1 *and* Sys_2 *with a valid mapping* f *be given. For reactive simulatability, we require that there exists a* *function* $f_\mathsf{C}\colon \mathsf{Conf}^f(Sys_1) \to \mathsf{Conf}(Sys_2)$ *with* $f_\mathsf{C}(conf_1).struc = f(conf_1.struc)$ *and* $f_\mathsf{C}(conf_1).\mathsf{H} = conf_1.\mathsf{H}$ *for all* $conf_1 \in \mathsf{Conf}^f(Sys_1)$*, and with the following properties. We say that* f_C *is a* τ-mapping *for a structure* $struc_1$ *and a function* $\tau\colon \mathbb{N} \to \mathbb{N}$ *if the complexity* $t_{f_\mathsf{C}(conf_1).\mathsf{A}}$ *is bounded by* $\tau(t_{conf_1.\mathsf{A}})$ *for all* $conf_1 \in \mathsf{Conf}(struc_1)$*. The entire* f_C *is a* τ*-mapping for a function* $\tau\colon \mathbb{N}^2 \to \mathbb{N}$ *if for all* $conf_1 \in \mathsf{Conf}^f(Sys_1)$ *we have* $t_{f_\mathsf{C}(conf_1).\mathsf{A}} \le \tau(k, t_{conf_1})$*.*

We say that $Sys_1 \ge_{\mathsf{sec}}^{f,y} Sys_2$*, spoken "*$y'$-at least as secure as*", under the following conditions for different cases of* y *and* y'*, where we abbreviate* $\mathsf{H} := conf_1.\mathsf{H}$*:*

a) $y = \mathsf{perf}$ *and* y' = *"perfectly" iff* $view_{conf_1}(\mathsf{H})$ *and* $view_{f_\mathsf{C}(conf_1)}(\mathsf{H})$ *are perfectly indistinguishable for every* $conf_1 \in \mathsf{Conf}^f(Sys_1)$*.*
b) $y = \delta$ *and* y' = *"*δ*-statistically" for a function* $\delta\colon \mathbb{N}^2 \to \mathbb{R}_{\ge 0}$ *iff for every* $conf_1 \in \mathsf{Conf}^f(Sys_{1,k})$ *and every* $l \in \mathbb{N}$ *we have* $view_{conf_1,l}(\mathsf{H}) \approx_{\delta(k,l)} view_{f_\mathsf{C}(conf_1),l}(\mathsf{H})$*.*
c) Concrete security: An algorithm Dis *is called a* (t,δ)-distinguisher *for* $conf_1 \in \mathsf{Conf}^f(Sys_{1,k})$ *and* $f_\mathsf{C}(conf_1)$ *where* $t \in \mathbb{N}$ *and* $\delta \in \mathbb{R}_{\ge 0}$ *iff its complexity is at most* t *and* $\delta_{conf_1}^{\mathsf{Dis}} \ge \delta$ *where*

$$\delta_{conf_1}^{\mathsf{Dis}} := |\Pr(\mathsf{Dis}(1^k, view_{conf_1}(\mathsf{H})) = 1) - \Pr(\mathsf{Dis}(1^k, view_{f_\mathsf{C}(conf_1)}(\mathsf{H})) = 1)|.$$

e) $y = \mathsf{poly}$ *and* y' = *"polynomially" iff for all users* H *and adversary* A *of polynomial complexity, the views* $(view_{(\hat{M},S,\mathsf{H}_{(\hat{M},S)},\mathsf{A}_{(\hat{M},S)})}(\mathsf{H}_{(\hat{M},S)}))_{(\hat{M},S)\in Sys_1}$ *and* $(view_{f_\mathsf{C}((\hat{M},S,\mathsf{H}_{(\hat{M},S)},\mathsf{A}_{(\hat{M},S)}))}(\mathsf{H}_{(\hat{M},S)}))_{(\hat{M},S)\in Sys_1}$ *are polynomially indistinguishable and* f_C *is a* P*-mapping for a polynomial* P*.*

Universal simulatability *means that* $f_\mathsf{C}(conf_1).\mathsf{A}$ *(i.e.,* A_2 *in Figure 2) for* $conf_1 = (\hat{M}_1, S, \mathsf{H}, \mathsf{A}_1)$ *only depends on* $\hat{M}_1$*,* S*, and* A_1*. We write* $\ge_{\mathsf{sec}}^{\mathsf{uni},f,y}$ *instead of* $\ge_{\mathsf{sec}}^{f,y}$ *if we want to emphasize this case.* $\diamond$

Where the difference between the types of security is irrelevant, we only write $\ge_{\mathsf{sec}}^f$, and we omit the indices f and 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 $\hat{M}$ of machines is written $\mathsf{comb}(\hat{M})$ and we sometimes write $\mathsf{comb}(\mathsf{M}_1, \ldots, \mathsf{M}_j)$ for $\mathsf{comb}(\{\mathsf{M}_1, \ldots, \mathsf{M}_j\})$.

Lemma 4. *(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* $(\hat{C} \setminus \hat{D}) \cup \{\mathsf{comb}(\hat{D})\}$ *is the same as in* $\hat{C}$*. This includes the view of the submachines in* $\mathsf{comb}(\hat{D})$*, which is well-defined given* $\hat{C}$ *and* $\hat{D}$*. The Turing complexity of* $\mathsf{comb}(\hat{D})$ *is the sum of the complexities of the machines in* $\mathsf{comb}(\hat{D})$*.*

$\square$

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 14. *(Blackbox Simulatability) With the notation of Definition 12, blackbox simulatability means that we have functions f_{Sim} from Sys_1 to machines (the simulators for the structures) and f_σ from Sys_1 to port renaming functions such that for all $conf_1 = (\hat{M}_1, S, \mathsf{H}, \mathsf{A}_1) \in \mathsf{Conf}^f(Sys_1)$ we have $f_{\mathsf{C}}(conf_1) = (\hat{M}_2, S, \mathsf{H}, \mathsf{A}_2)$ with $(\hat{M}_2, S) = f((\hat{M}_1, S))$ and $\mathsf{A}_2 = \mathsf{comb}(\mathsf{Sim}, \mathsf{A}'_1)$ with $\mathsf{Sim} := f_{\mathsf{Sim}}((\hat{M}_1, S_1))$ and $\mathsf{A}'_1 := f_\sigma((\hat{M}_1, S_1))(\mathsf{A}_1)$. For computational security, we require that Sim is polynomial-time, i.e., that the parametrized complexity of $(f_{\mathsf{Sim}}((\hat{M}_1, S)))_{(\hat{M}_1, S) \in Sys_1}$ is polynomially bounded. We write $\geq^{\mathsf{bb}}_{\mathsf{sec}}$ instead of $\geq_{\mathsf{sec}}$ 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 15. *(Composability and Composition of Structures) We call structures $(\hat{M}_1, S_1), \ldots, (\hat{M}_n, S_n)$* composable *if $\mathsf{ports}(\hat{M}_i) \cap \mathsf{forb}(\hat{M}_j, S_j) = \emptyset$ and $S_i \cap \mathsf{free}([\hat{M}_j]) = S_j \cap \mathsf{free}([\hat{M}_i])$ for all $i \neq j$.*[2] *We then define their composition as $(\hat{M}_1, S_1)||\ldots||(\hat{M}_n, S_n) := (\hat{M}, S)$ with $\hat{M} := \hat{M}_1 \cup \ldots \cup \hat{M}_n$ and $S := (S_1 \cup \ldots \cup S_n) \cap \mathsf{free}([\hat{M}])$.* $\diamond$

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

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.

[2] The first condition makes one structure a valid user of another. The second one excludes cases where $p \in \mathsf{free}([\hat{M}_i]) \cap \mathsf{free}([\hat{M}_j])$ (e.g., a clock port for a connection between these structures) and $p \in S_i$ but $p \notin S_j$.

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.

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

Let $P\colon \mathbb{N} \to \mathbb{N}$*, and let* $Sys^{\#}$ *and* Sys^* *denote the* P*-sized compositions of* $Sysseq$ *and* $Sysseq'$*, respectively. Assume that the following structural conditions hold for all* $k \in \mathbb{N}$ *and every structure* $(\hat{M}^{\#}, S) \in Sys_k^{\#}$*: Let its restrictions be* $(\hat{M}_i, S_i) := (\hat{M}^{\#}, S)\lceil_{Sys^{(i)}}$ *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*

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

exists and lies in Sys_k^**. Furthermore,* $(\hat{M}_i, S_i)$ *and* $(\hat{M}_j', S_j)$ *must be composable for* $j \neq i$*, and* $\mathsf{ports}(\hat{M}_i') \cap S_j^c = \mathsf{ports}(\hat{M}_i) \cap S_j^c$ *for all* $j \neq i$*. Then we have*

$$Sys^{\#} \geq_{\mathsf{sec}}^{\mathsf{bb}, f^{\#}, y} Sys^*$$

- *a) with* $y = \mathsf{perf}$ *if* $y_i = \mathsf{perf}$ *for all* $i \in \mathbb{N}$*.*
- *b) with* $y = P(k) \cdot \delta(k, b(k))$ *if all* y_i *are a function* $\delta\colon \mathbb{N}^2 \to \mathbb{R}_{\geq 0}$*, and where* $b(k)$ *is the sum of the complexity of the systems, the user, and the simulators.*
- *c) With concrete complexity: For every* $conf^{\#} \in \mathsf{Conf}^{f^{\#}}(Sys_k^{\#})$*, a* (t, δ)*-distinguisher for* $conf^{\#}$ *and* $f_{\mathsf{C}}(conf^{\#})$ *gives rise to a* (t', δ')*-distinguisher for* $conf^{(i)}$ *and* $f_{\mathsf{C}}(conf^{(i)})$ *for a* $conf^{(i)} \in \mathsf{Conf}(Sys_k^{(i)})$ *with* $\delta' = \frac{\delta}{P(k)}$ *and* $t' = t + b'(k)$*, where* $b'(k)$ *is a polynomial independent of* $t_{conf^{\#}.\mathsf{A}}$*. (Details are given in the proof.)*
- *d) with* $y = \mathsf{poly}$ *if* $y_i = \mathsf{poly}$ *for all* $i \in \mathbb{N}$ *and under the following conditions: The function* P *is polynomially bounded, and the systems* $Sys^{(i)}$ *have a joint bounding polynomial* Q*. The complexities of the simulator families induced by the mappings* $f_{\mathsf{Sim}}^{(i)}$ *are bounded by a joint polynomial* Q_{Sim}*. The distinguishing probabilities of the system pairs* $(Sys^{(i)}, Sys'^{(i)})$ *are uniformly bounded, i.e., for all polynomials* t *there exists a negligible function* δ *such that for all distinguishers* Dis*, all* $i, k \in \mathbb{N}$*,*

and all $conf = (\hat{M}_i, S_i, \mathsf{H}, \mathsf{A}) \in \mathsf{Conf}^{f^{(i)}}(Sys_k^{(i)})$ *we have* $(t_{\mathsf{Dis}} \le t(k) \wedge t_{\mathsf{H}} \le t(k) \wedge t_{\mathsf{A}} \le t(k)) \Rightarrow \delta_{conf}^{\mathsf{Dis}} \le \delta(k)$ *(recall Definition 12d).*

□

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

Lemma 5. *Under the conditions of Theorem 1, the mapping* $f^{\#}$ *is a valid mapping between* $Sys^{\#}$ *and* Sys^**.* □

The proof is straightforward as in [30], but heavy on notation. Hence we omit it in this short version. Recall that blackbox simulatability was defined by a function that selects one fixed simulator for each structure (Definition 14).

Definition 17. *(Simulator and Corresponding Configurations) Under the conditions of Theorem 1 and for all* $i \in \mathbb{N}$*, let* $f_{\mathsf{Sim}}^{(i)}$ *and* $f_{\mathsf{A}}^{(i)}$ *be the simulator and renaming functions from which* $f_{\mathsf{C}}^{(i)}$ *is composed by blackbox simulatability. We compose them into functions* $f_{\mathsf{Sim}}^{\#}$ *and* $f_{\mathsf{A}}^{\#}$ *on* $Sys^{\#}$ *as follows: Given* $k \in \mathbb{N}$ *and* $(\hat{M}^{\#}, S) \in Sys_k^{\#}$*, let* $\mathsf{Sim}_i := f_{\mathsf{Sim}}^{(i)}((\hat{M}_i, S_i))$ *for all* $i \le P(k)$*, and let*

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

further let $f_{\mathsf{A}}^{\#} := f_{\mathsf{A}}^{(P(k))} \circ \cdots \circ f_{\mathsf{A}}^{(1)}$*. Let* $f_{\mathsf{C}}^{\#}$ *be constructed from* $f^{\#}$*,* $f_{\mathsf{Sim}}^{\#}$*, and* $f_{\mathsf{A}}^{\#}$ *by the equations in Definition 14 (blackbox simulatability).* ◇

The complexity t_{Sim} of the simulator is $t_{\mathsf{Sim}}(k) = \sum_{i=1}^{P(k)} t_{\mathsf{Sim}_i}(k)$ by Lemma 4. In the polynomial case, there exists a polynomial Q_{Sim} such that $t_{\mathsf{Sim}_i} \le Q_{\mathsf{Sim}}$ for all i, hence $t_{\mathsf{Sim}}(k)$ is polynomially bounded by $P(k) \cdot Q_{\mathsf{Sim}}(k)$.

We also omit the technical proof that indeed $f_{\mathsf{C}}^{\#}\colon \mathsf{Conf}^{f^{\#}}(Sys^{\#}) \to \mathsf{Conf}(Sys^*)$ in Definition 17. 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 [15], i.e., we construct intermediate configurations that differ only in the machines of one system.

Proof (Theorem 1). Let a configuration $conf^{\#} = (\hat{M}^{\#}, S, \mathsf{H}, \mathsf{A}) \in \mathsf{Conf}^{f^{\#}}(Sys_k^{\#})$ be given and $conf^* := f_{\mathsf{C}}^{\#}(conf^{\#})$ the corresponding configuration according to Definition 17. Let the sub-structures $(\hat{M}_i, S_i)$ and $(\hat{M}_i', S_i')$, the simulators Sim_i, and functions f_z^x with various indices be defined as in the formulation of the theorem and Definition 17. Furthermore, let $(\hat{M}^*, S) := f^*((\hat{M}^{\#}, S))$ and $\mathsf{Sim} := f_{\mathsf{Sim}}^{\#}((\hat{M}^{\#}, S))$. Then we have $conf^* = (\hat{M}^*, S, \mathsf{H}, \mathsf{comb}(\mathsf{Sim}, f_{\mathsf{A}}^{\#}(\mathsf{A})))$; recall that $f_{\mathsf{A}}^{\#}$ is just a port renaming; hence Figure 3 simplifies it to A.

The outline of the hybrid argument is as follows.

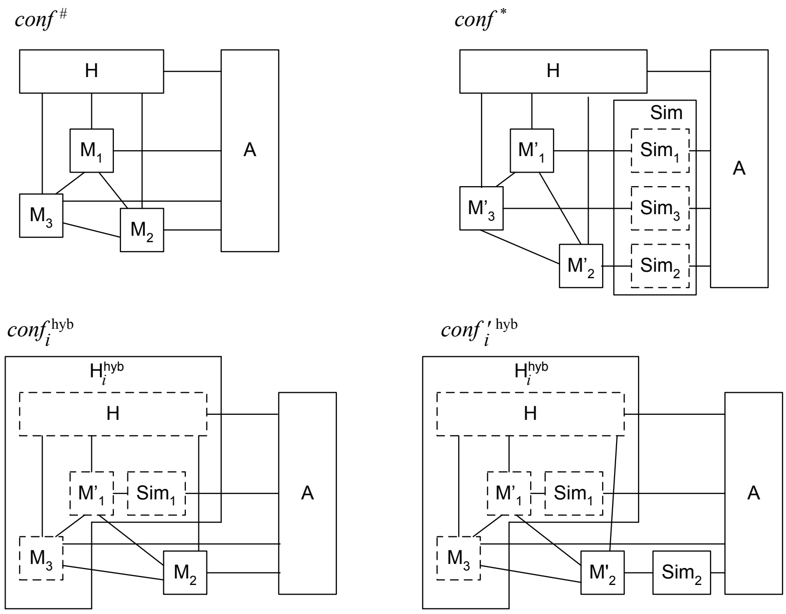

Fig. 3. Configurations in the composition theorem for blackbox simulatability.

1. We define *hybrid configurations* $\mathit{conf}_i^{\mathsf{hyb}}$ of $\mathit{Sys}^{(i)}$ and $\mathit{conf}_i'^{\mathsf{hyb}}$ of $\mathit{Sys}'^{(i)}$ for $i = 1, \ldots, P(k)$. In $\mathit{conf}_i^{\mathsf{hyb}}$ the first $i-1$ real structures have already been replaced with their ideal counterparts, while in $\mathit{conf}_i'^{\mathsf{hyb}}$ 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 $\mathsf{H}_i^{\mathsf{hyb}}$ 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 $\mathit{Sys}^{(i)}$ and $\mathit{Sys}'^{(i)}$.
3. We show that the views of H in $\mathit{conf}_i'^{\mathsf{hyb}}$ and $\mathit{conf}_{i+1}^{\mathsf{hyb}}$ are equal for $i = 1, \ldots, P(k) - 1$. Moreover, we show that the views of H are equal in $\mathit{conf}^{\#}$ and $\mathit{conf}_1^{\mathsf{hyb}}$, and in $\mathit{conf}_{P(k)}'^{\mathsf{hyb}}$ and conf^*. This gives a kind of indistinguishability chain (for one configuration)

$$\mathit{view}_{\mathit{conf}^{\#}}(\mathsf{H}) \approx \mathit{view}_{\mathit{conf}_1^{\mathsf{hyb}}}(\mathsf{H}) \approx \cdots \approx \mathit{view}_{\mathit{conf}_{P(k)}'^{\mathsf{hyb}}}(\mathsf{H}) \approx \mathit{view}_{\mathit{conf}^*}(\mathsf{H}).$$

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 $\hat{H}_i := \{\mathsf{H}\} \cup \bigcup_{1 \leq j < i} \hat{M}_j' \cup \{\mathsf{Sim}_j \mid 1 \leq j < i\} \cup \bigcup_{i < j \leq P(k)} \hat{M}_j$, and let $\mathsf{H}_i^{\mathsf{hyb}} := \mathsf{comb}(\hat{H}_i)$. Furthermore let $\mathsf{A}_i := f_{\mathsf{A}}^{(i-1)} \circ \cdots \circ f_{\mathsf{A}}^{(1)}(\mathsf{A})$ and $\mathsf{A}_i' := f_{\mathsf{A}}^{(i)}(\mathsf{A}_i)$. Then we define the *hybrid configurations* as

$$conf_i^{\mathsf{hyb}} := (\hat{M}_i, S_i, \mathsf{H}_i^{\mathsf{hyb}}, \mathsf{A}_i);$$
$$conf_i'^{\mathsf{hyb}} := (\hat{M}_i', S_i, \mathsf{H}_i^{\mathsf{hyb}}, \mathsf{comb}(\mathsf{Sim}_i, \mathsf{A}_i')).$$

For the computational case, we have to show that the family of $\mathsf{H}_i^{\mathsf{hyb}}$ is polynomial-time. This holds since $t_{\mathsf{H}_i^{\mathsf{hyb}}} \leq t_{\mathsf{H}} + t_{\mathsf{Sim}} + t_{\hat{M}^\#} + t_{\hat{M}^*}$ by Lemma 4, where each addend is polynomially bounded by assumption.

Step 2: We have to show that $conf_i^{\mathsf{hyb}} \in \mathsf{Conf}^{f_i}(Sys_i)$ and $conf_i'^{\mathsf{hyb}} \in \mathsf{Conf}(Sys_i')$, 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 $conf_i^{\mathsf{hyb}}$ and $conf_i'^{\mathsf{hyb}}$ immediately implies

$$conf_i'^{\mathsf{hyb}} = f_{\mathsf{C}}^{(i)}(conf_i^{\mathsf{hyb}}), \tag{1}$$

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

Step 3: The configurations $conf_i'^{\mathsf{hyb}}$ and $conf_{i+1}^{\mathsf{hyb}}$ consist of the same collection of machines $\hat{C}_i := \hat{H}_i \cup \{\hat{M}_i', \mathsf{Sim}_i, \mathsf{A}_i'\}$. Combining them in different ways does not alter the view of H by Lemma 4. Thus we have

$$view_{conf_i'^{\mathsf{hyb}}}(\mathsf{H}) = view_{conf_{i+1}^{\mathsf{hyb}}}(\mathsf{H}) \tag{3}$$

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

$$view_{conf^\#}(\mathsf{H}) = view_{conf_1^{\mathsf{hyb}}}(\mathsf{H}) \wedge view_{conf_{P(k)}^{\mathsf{hyb}}}(\mathsf{H}) = view_{conf^*}(\mathsf{H}). \tag{4}$$

Step 4: We now distinguish the type of the given simulatability relations $Sys^{(i)} \geq_{\mathsf{sec}}^{\mathsf{bb}, f^{(i)}, y_i} Sys'^{(i)}$.

For perfect simulatability, Equation (1) gives us $view_{conf_i^{\mathsf{hyb}}}(\mathsf{H}) = view_{conf_i'^{\mathsf{hyb}}}(\mathsf{H})$ for all i. With Equations (3) and (4) this yields $view_{conf^\#}(\mathsf{H}) = view_{conf^*}(\mathsf{H})$. This result for an arbitrary fixed configuration $conf^\#$ implies equality of all families of such views.

For statistical simulatability, let $Sys^{(i)}$ be δ_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 $q_v^\# := \mathsf{Pr}(view_{conf^\#,l}(\mathsf{H}) = v)$, and $q_v^* := \mathsf{Pr}(view_{conf^*,l}(\mathsf{H}) = v)$, and $q_{i,v} := \mathsf{Pr}(view_{conf_i^{\mathsf{hyb}},l}(\mathsf{H}) = v)$ and $q_{i,v}' := \mathsf{Pr}(view_{conf_i'^{\mathsf{hyb}},l}(\mathsf{H}) = v)$ for all i. For all potential views v, we have $q_{i,v}' = q_{i+1,v}$ and $q_v^\# = q_{1,v}$ and $q_{P(k),v}' = q_v^*$ by Equations (3) and (4). The desired statistical distance is

$$\begin{aligned}
\delta_{\mathsf{stat}}(conf^\#) &:= \frac{1}{2}\sum_v |q_v^\# - q_v^*| \\
&= \frac{1}{2}\sum_v |q_{1,v} - q_{2,v} + q_{2,v} - q_{3,v} + \dots + q_{P(k),v} - q_{P(k),v}'| \\
&\leq \frac{1}{2}\sum_v (|q_{1,v} - q_{2,v}| + |q_{2,v} - q_{3,v}| + \dots + |q_{P(k),v} - q_{P(k),v}'|)
\end{aligned}$$

$$= \sum_{i=1}^{P(k)} \frac{1}{2} \sum_{v} |q_{i,v} - q'_{i,v}|$$
$$= \sum_{i=1}^{P(k)} \Delta_{\mathsf{stat}}(view_{conf_i^{\mathsf{hyb}},l}(\mathsf{H}), view_{conf_i'^{\mathsf{hyb}},l}(\mathsf{H})).$$

With Lemma 2 this gives

$$\delta_{\mathsf{stat}}(conf^{\#}) \leq \sum_{i=1}^{P(k)} \Delta_{\mathsf{stat}}(view_{conf_i^{\mathsf{hyb}},l_i}(\mathsf{H}_i^{\mathsf{hyb}}), view_{conf_i'^{\mathsf{hyb}},l_i}(\mathsf{H}_i^{\mathsf{hyb}})) \leq \sum_{i=1}^{P(k)} \delta(k, l_i),$$

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

For concrete complexity and for a $(t, \Delta^{\mathsf{Dis}})$-distinguisher Dis, we have by definition

$$\Delta^{\mathsf{Dis}} \leq |\mathsf{Pr}(\mathsf{Dis}(1^k, view_{conf^{\#}}(\mathsf{H})) = 1) - \mathsf{Pr}(\mathsf{Dis}(1^k, view_{conf^*}(\mathsf{H})) = 1)|.$$

We abbreviate $q^{\#} := \mathsf{Pr}(\mathsf{Dis}(1^k, view_{conf^{\#}}(\mathsf{H})) = 1)$ and $q^* := \mathsf{Pr}(\mathsf{Dis}(1^k, view_{conf^*}(\mathsf{H})) = 1)$, and $q_i := \mathsf{Pr}(\mathsf{Dis}(1^k, view_{conf_i^{\mathsf{hyb}}}(\mathsf{H})) = 1)$ and $q'_i := \mathsf{Pr}(\mathsf{Dis}(1^k, view_{conf_i'^{\mathsf{hyb}}}(\mathsf{H})) = 1)$ for all i, and $\Delta_i := |q_i - q'_i|$. Now Equations (3) and (4) yield

$$\begin{aligned}\Delta^{\mathsf{Dis}} &= |q^{\#} - q^*| = |q_1 - q_2 + q_2 - q_3 + q_3 + \cdots + q_{P(k)} - q'_{P(k)}| \\ &\leq |q_1 - q_2| + |q_2 - q_3| + \cdots + |q_{P(k)} - q'_{P(k)}| = \Delta_1 + \Delta_2 + \cdots + \Delta_{P(k)}.\end{aligned}$$

This implies that there exists some i with $\Delta_i \geq \frac{\Delta^{\mathsf{Dis}}}{P(k)}$.

We can now consider Dis as a (t, Δ_i)-distinguisher $\mathsf{Dis}_{\phi}^{(i)}$ of a function ϕ of views of the actual user $\mathsf{H}_i^{\mathsf{hyb}}$ of the i-th hybrid systems. Here ϕ is defined by $\phi(v) := v\lceil_{\mathsf{H}}$, i.e., the restriction to the view of H. The complexity t_ϕ of ϕ is linear. Hence Lemma 2 implies that there exists a (t_i, Δ_i)-distinguisher $\mathsf{Dis}^{(i)}$ for $view_{conf_i^{\mathsf{hyb}}}(\mathsf{H}_i^{\mathsf{hyb}})$ and $view_{conf_i'^{\mathsf{hyb}}}(\mathsf{H}_i^{\mathsf{hyb}})$ with $t' = t + b'(k)$, where $b'(k)$ bounds the length of the views of $\mathsf{H}_i^{\mathsf{hyb}}$. The complexity $t_{\mathsf{H}_i^{\mathsf{hyb}}}$ of $\mathsf{H}_i^{\mathsf{hyb}}$ is bounded by $b = t_{\mathsf{H}} + t_{\hat{M}^{\#}} + t_{\hat{M}^*} + t_{\mathsf{Sim}}$, and above we showed $t_{\mathsf{Sim}} \leq P \cdot Q_{\mathsf{Sim}}$. 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 $t_{\mathsf{H}_i^{\mathsf{hyb}}}$, t_i, and $t_{\mathsf{A}_i} = t_{\mathsf{A}}$ are polynomials. By assumption, there exists a negligible function δ that uniformly bounds the advantage of distinguishers with complexity $\max(t_i, t_{\mathsf{H}_i^{\mathsf{hyb}}}, t_{\mathsf{A}})$ for the given system pairs. Now let a configuration $conf^{\#} = (\hat{M}^{\#}, S, \mathsf{H}_{(\hat{M}^{\#},S)}, \mathsf{A}_{(\hat{M}^{\#},S)})$ be given. The concrete security considerations and Equation (1) imply $\Delta_i = \delta^{\mathsf{Dis}^{(i)}}_{conf_i^{\mathsf{hyb}}} \le \delta(k)$, and therefore $\delta^{\mathsf{Dis}}_{conf^{\#}} \le P(k) \cdot \delta(k)$. The function $P \cdot \delta$ is negligible. This proves the desired polynomial indistinguishability of the families of user views in $\mathsf{Conf}^{f^{\#}}(Sys^{\#})$ and $\mathsf{Conf}(Sys^*)$. ■

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 $(\hat{M}_1, S_1) \in Sys_1$ is contained in $\bar{S}_1^c$, 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 $(\hat{M}_1, S_1) \in Sys_1$, we have $\mathsf{p}^{\triangleleft}! \in \bar{S}_1^c \Rightarrow (\mathsf{p}? \in \bar{S}_1^c \vee \mathsf{p}! \in \bar{S}_1^c)$. Then $Sys_1 \ge_{\mathsf{sec}}^{\mathsf{bb},f,y} Sys_2$ iff $Sys_1 \ge_{\mathsf{sec}}^{\mathsf{uni},f,y} Sys_2$ for $y = \mathsf{perf}$ or a function δ and also for $y = \mathsf{poly}$ if Sys_1 is polynomial-time.*

For concrete security, if $\ge_{\mathsf{sec}}^{\mathsf{uni},f}$ is given with a τ-mapping f_{C}, then we obtain $\ge_{\mathsf{sec}}^{\mathsf{bb},f}$ with simulator complexity $\tau(t_{Sys_1})$, and a (t, δ)-distinguisher for the views in the black-box case gives rise to a (t', δ)-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 $conf_1 = (\hat{M}_1, S, \mathsf{H}, \mathsf{A}_1)$ of the sub-system $Sys_{1,k}$ be given. We first derive another configuration $conf_1^{\mathsf{uni}} = (\hat{M}_1, S, \mathsf{H}^{\mathsf{uni}}, \mathsf{A}_1')$ of Sys_1 as follows: We insert a machine $\mathsf{TS}_{P,b,k}$, called *transparent scheduler*, into the connections between A_1 and the simple ports in $\bar{S}_1$. 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 $\mathsf{H}^{\mathsf{uni}} := \mathsf{comb}(\mathsf{H}, \mathsf{A}_1)$, and the new adversary is $\mathsf{A}_1' := \mathsf{TS}_{P,b,k}$. We show that the views of both H and A_1 are identical in the two configurations.
2. We now show that $conf_1^{\mathsf{uni}} \in \mathsf{Conf}^f(Sys_1)$ and apply the precondition $Sys_1 \ge_{\mathsf{sec}}^{\mathsf{uni},f} Sys_2$. This yields an indistinguishable configuration $conf_2^{\mathsf{uni}}$ 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 $\mathsf{A}_1' = \mathsf{TS}_{P,b,k}$. Since $\mathsf{TS}_{P,b,k}$ 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 $conf_2^{\mathsf{uni}}$ by reversing the combination of H and A_1 into $\mathsf{H}^{\mathsf{uni}}$, and by defining the simulator as $\mathsf{Sim} := \mathsf{A}_2$. 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. ∎

Acknowledgments

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

References

1. M. Abadi and L. Lamport. Conjoining specifications. *ACM Transactions on Programming Languages and Systems*, 17(3):507–534, 1995.
2. M. Backes and C. Jacobi. Cryptographically sound and machine-assisted verification of security protocols. In *Proc. 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2607 of *Lecture Notes in Computer Science*, pages 675–686. Springer, 2003.
3. M. Backes and B. Pfitzmann. Computational probabilistic non-interference. In *Proc. 7th European Symposium on Research in Computer Security (ESORICS)*, volume 2502 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2002.
4. M. Backes and B. Pfitzmann. Intransitive non-interference for cryptographic purposes. In *Proc. 24th IEEE Symposium on Security & Privacy*, pages 140–152, 2003.
5. M. Backes, B. Pfitzmann, M. Steiner, and M. Waidner. Polynomial fairness and liveness. In *Proc. 15th IEEE Computer Security Foundations Workshop (CSFW)*, pages 160–174, 2002.
6. D. Beaver. Secure multiparty protocols and zero knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, 1991.
7. M. Bellare, J. Killian, and P. Rogaway. The security of cipherblock chaining. In *Advances in Cryptology: CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 341–358. Springer, 1994.
8. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology: EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994.
9. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 3(1):143–202, 2000.
10. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001. Extended version in Cryptology ePrint Archive, Report 2000/67, `http://eprint.iacr.org/`.
11. A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Secure protocol composition (extended abstract). In *Proc. 1st ACM Workshop on Formal Methods in Security Engineering (FMSE)*, pages 11–23, 2003.
12. O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

13. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game – or – a completeness theorem for protocols with honest majority. In *Proc. 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.
14. S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In *Advances in Cryptology: CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 77–93. Springer, 1990.
15. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
16. M. Hirt and U. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, 2000.
17. C. A. R. Hoare. *Communicating Sequential Processes*. International Series in Computer Science, Prentice Hall, Hemel Hempstead, 1985.
18. D. M. Johnson and F. Javier Thayer. Security and the composition of machines. In *Proc. 1st IEEE Computer Security Foundations Workshop (CSFW)*, pages 72–89, 1988.
19. P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proc. 5th ACM Conference on Computer and Communications Security*, pages 112–121, 1998.
20. P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. Probabilistic polynomial-time equivalence and security analysis. In *Proc. 8th Symposium on Formal Methods Europe (FME 1999)*, volume 1708 of *Lecture Notes in Computer Science*, pages 776–793. Springer, 1999.
21. N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, San Francisco, 1996.
22. H. Mantel. On the composition of secure systems. In *Proc. 23rd IEEE Symposium on Security & Privacy*, pages 88–101, 2002.
23. D. McCullough. Specifications for multi-level security and a hook-up property. In *Proc. 8th IEEE Symposium on Security & Privacy*, pages 161–166, 1987.
24. D. McCullough. A hookup theorem for multilevel security. *IEEE Transactions on Software Engineering*, 16(6):563–568, 1990.
25. J. McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proc. 15th IEEE Symposium on Security & Privacy*, pages 79–93, 1994.
26. J. McLean. A general theory of composition for a class of "possibilistic" security properties. *IEEE Transactions on Software Engineering*, 22(1):53–67, 1996.
27. S. Micali and P. Rogaway. Secure computation. In *Advances in Cryptology: CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 392–404. Springer, 1991.
28. J. Misra and K. M. Chandy. Proofs of network of processes. *IEEE Transactions on Software Engineering*, 7(4):417–426, 1981.
29. B. Pfitzmann and M. Waidner. A general framework for formal notions of "secure" systems. Research Report 11/94, University of Hildesheim, Apr. 1994. `http://www.semper.org/sirene/lit/abstr94.html#PfWa_94`.
30. B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. 7th ACM Conference on Computer and Communications Security*, pages 245–254, 2000. Extended version (with Matthias Schunter) IBM Research Report RZ 3206, May 2000, `http://www.semper.org/sirene/publ/PfSW1_00ReactSimulIBM.ps.gz`.
31. B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proc. 22nd IEEE Symposium on Security & Privacy*, pages 184–200, 2001. Extended version in Cryptology ePrint Archive, Report 2000/066, `http://eprint.iacr.org/`.
32. J. Widom, D. Gries, and F. B. Schneider. Trace-based network proof systems: Expressiveness and completeness. *ACM Transactions on Programming Languages and Systems*, 14(3):396–416, 1992.

33. A. C. Yao. Protocols for secure computations. In *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1982.
34. A. C. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.