On the Relationships Between Notions of Simulation-Based Security*

Anupam Datta[1], Ralf Küsters[2], John C. Mitchell[1], and Ajith Ramanathan[1]

[1] Computer Science Department
Stanford University
Stanford CA 94305-9045, USA
{danupam,jcm,ajith}@cs.stanford.edu

[2] Institut für Informatik
Christian-Albrechts-Universität zu Kiel
24098 Kiel, Germany
kuesters@ti.informatik.uni-kiel.de

Abstract. Several compositional forms of simulation-based security have been proposed in the literature, including universal composability, black-box simulatability, and variants thereof. These relations between a protocol and an ideal functionality are similar enough that they can be ordered from strongest to weakest according to the logical form of their definitions. However, determining whether two relations are in fact identical depends on some subtle features that have not been brought out in previous studies. We identify the position of a "master process" in the distributed system, and some limitations on transparent message forwarding within computational complexity bounds, as two main factors. Using a general computational framework, we clarify the relationships between the simulation-based security conditions.

1 Introduction

Several current projects use ideal functionality and indistinguishability to state and prove compositional security properties of protocols and related mechanisms. The main projects include work by Canetti and collaborators on an approach called *universal composabiliity* [8, 10–13] and work by Backes, Pfitzmann, and Waidner on a related approach that also uses *black-box simulatability* [22, 7, 4, 5]. Other projects have used the notion of equivalence in process calculus [16, 18,

* This work was partially supported by the DoD University Research Initiative (URI) program administered by the Office of Naval Research under Grant N00014-01-1-0795, by OSD/ONR CIP/SW URI "Trustworthy Infrastructure, Mechanisms, and Experimentation for Diffuse Computing" through ONR Grant N00014-04-1-0725, by NSF CCR-0121403, Computational Logic Tools for Research and Education, and by NSF CyberTrust Grant 0430594, Collaborative research: High-fidelity methods for security protocols. Part of this work was carried out while the second author was at Stanford University supported by the "Deutsche Forschungsgemeinschaft (DFG)".

19], a well-established formal model of concurrent systems. While some process-calculus-based security studies [2, 3, 1] abstract away probability and computational complexity, at least one project [20, 17, 21, 23] has developed a probabilistic polynomial-time process calculus for security purposes. The common theme in each of these approaches is that the security of a real protocol is expressed by comparison with an ideal functionality or ideal protocol. However, there are two main differences between the various approaches: the precise relation between protocol and functionality that is required, and the computational modeling of the entities (protocol, adversary, simulator, and environment). All of the computational models use probabilistic polynomial-time processes, but the ways that processes are combined to model a distributed system vary. We identify two main ways that these computational models vary: one involving the way the next entity to execute is chosen, and the other involving the capacity and computational cost of communication. We then show exactly when the main security notions differ or coincide.

In [8], Canetti introduced universal composability (UC), based on probabilistic polynomial-time interacting Turing machines (PITMs). The UC relation involves a real protocol and ideal functionality to be compared, a real and ideal adversary, and an environment. The real protocol realizes the ideal functionality if, for every attack by a real adversary on the real protocol, there exists an attack by an ideal adversary on the ideal functionality, such that the observable behavior of the real protocol under attack is the same as the observable behavior of the ideal functionality under attack. Each set of observations is performed by the same environment. In other words, the system consisting of the environment, the real adversary, and the real protocol must be indistinguishable from the system consisting of the environment, the ideal adversary, and the ideal functionality. The scheduling of a system of processes (or ITMs) is *sequential* in that only one process is active at a time, completing its computation before another is activated. The default process to be activated, if none is designated by process communication, is the environment. In the present work, we use the term *master process* for the default process in a system that runs when no other process has been activated by explicit communication.

In [22], Pfitzmann and Waidner use a variant of UC and a notion of blackbox simulatability (BB) based on probabilistic polynomial-time IO automata (PIOA). In the BB relation between a protocol and ideal functionality, the UC ideal adversary is replaced by the combination of the real adversary and a simulator that must be chosen independently of the real adversary. Communication and scheduling in the PIOA computational model are sequential as in the PITM model. While the environment is the master process in the PITM studies, the adversary is chosen to be the master process in the Pfitzmann-Waidner version of UC. In the Pfitzmann-Waidner version of BB the master process is the adversary or the simulator [22]. In a later version of the PIOA model (see, e.g., [4]), the environment is also allowed to serve as the master process, subject to the restriction that in any given system it is not possible to designate both the adversary/simulator and the environment as the master process. In proofs in

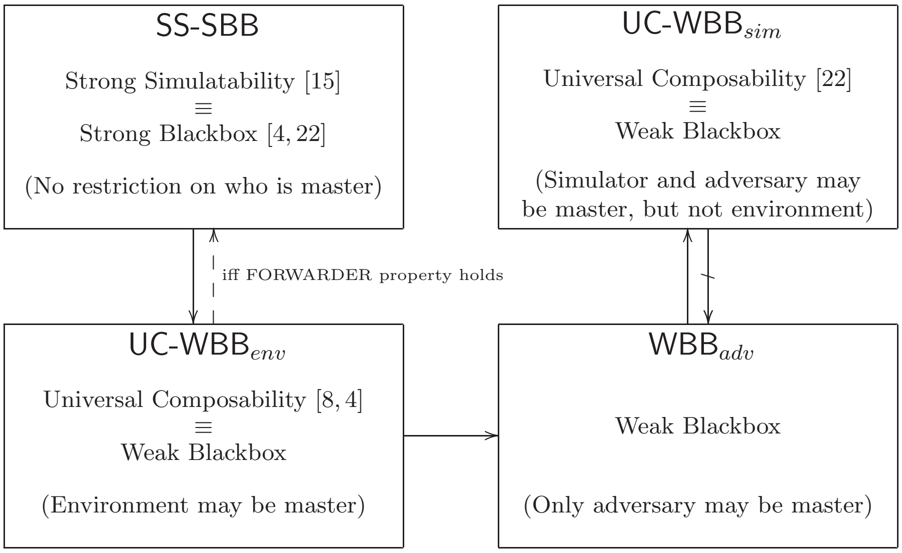

Fig. 1. Equivalences and implications between the security notions in SPPC

cryptography, another variant of BB is often considered in which the simulator may depend on the real adversary or its complexity. We call this variant Weak BB (WBB) and the previous one Strong BB (SBB).

In [17, 21, 23, 24], Mitchell et al. have used a form of process equivalence, where an environment directly interacts with the real and ideal protocol. The computational model in this work is a probabilistic polynomial-time processes calculus (PPC) that allows concurrent (non-sequential) execution of independent processes. The process equivalence relation gives rise to a relation between protocols and ideal functionalities by allowing a simulator to interact with the ideal functionality, resulting in a relation that we call strong simulatability, SS [15]. The difference between SS and SBB is that in SBB, the environment and the adversary are separated while the SS environment also serves as the adversary.

Contribution of the paper. In this paper, we clarify the relationships between UC, SBB, WBB, SS under different placements of the master process and an additional issue involving the ability to define a "forwarding" process that forwards communication from one process to another. While it seems intuitively reasonable that such a forwarder can be placed between two processes without changing the overall behavior of the system, this may violate complexity bounds if a polynomial-time forwarder must be chosen before the sending or receiving process. If the time bound of the sender, for example, exceeds the time bound of the forwarder, then some sent messages may be lost because the time bound of the forwarder has been exhausted. This is relevant to our study because some equivalence proofs require the existence of forwarders that cannot be exhausted.

Our main results are summarized in Figure 1. Each of the four boxes in this figure stands for a class of equivalent security notions. Specifically, if a real and ideal protocol are related by one notion in this class, then they are also related by all other notions in this class. A solid arrow from one class to another indicates that relations in the first class imply relations in the second class. The implication indicated by the dashed arrow is contingent on whether the aforementioned forwarding property holds for the processes in question.

The proofs of equivalence and implication between security notions are axiomatic, using a relatively small set of clearly stated equivalence principles involving processes and distributed systems. This approach gives us results that carry over to a variety of computational models. Our axiomatic system is proved sound for a specific computational model, a sequential probabilistic polynomial-time process calculus (SPPC), developed for the purpose of this study. SPPC is a sequential model, allowing only one process to run at a time. When one process completes, it sends an output indicating which process will run next. This calculus is close to PIOA and PITM in expressiveness and spirit, while (1) providing a syntax for writing equations between systems of communicating machines and (2) being flexible enough to capture different variants of security notions, including all variants of SS, SBB, WBB, and UC discussed in this paper. Our results about these security notions formulated over SPPC are:

1. Equivalences between security notions.
   (a) The different forms of Strong Simulatability and Strong Blackbox obtained by varying the entity that is the master process are all equivalent. This equivalence class, denoted $\mathsf{SS\text{-}SBB}$, is depicted in the top-left box in Figure 1 and includes placements of the master process as considered for Strong Blackbox in [4, 22]
   (b) All variants of Universal Composability and Weak Blackbox in which the environment may be the master process are equivalent. This equivalence class, denoted $\mathsf{UC\text{-}WBB}_{env}$, is depicted in the bottom-left box in Figure 1 and includes placements of the master process as considered for Universal Composability in [8, 4].
   (c) All variants of Universal Composability and Weak Blackbox in which the simulator and the adversary may be the master process, but not the environment are equivalent. This equivalence class, denoted $\mathsf{UC\text{-}WBB}_{sim}$, is depicted in the top-right box in Figure 1 and includes placements of the master process as considered for Universal Composability in [22].
   (d) All variants of Weak Blackbox where the adversary may be the master process, but neither the environment nor the simulator may play this role are equivalent. This equivalence class, denoted WBB_{adv}, is depicted in the bottom-right box in Figure 1.
2. Implications between the classes.
   (a) $\mathsf{SS\text{-}SBB}$ implies $\mathsf{UC\text{-}WBB}_{env}$. In particular, Strong Blackbox with placements of the master process as considered in [4, 22] implies Universal Composability with placements of the master process as considered in [8, 4].

(b) $\mathsf{UC\text{-}WBB}_{env}$ implies WBB_{adv}.
(c) WBB_{adv} implies $\mathsf{UC\text{-}WBB}_{sim}$. In particular, Strong Blackbox with placements of the master process as considered in [4, 22] and Universal Composability with placements of the master process as considered in [8, 4] implies Universal Composability with placements of the master process as considered in [22].

3. Separations between the classes.
   (a) The security notions in $\mathsf{UC\text{-}WBB}_{env}$ are strictly weaker than those in $\mathsf{SS\text{-}SBB}$ in any computational model where the forwarding property (expressed precisely by the FORWARDER axiom) fails. Since this property fails in the PITM model [8] and the buffered PIOA model [4], it follows that $\mathsf{UC\text{-}WBB}_{env}$ does not imply $\mathsf{SS\text{-}SBB}$ in these models. This contradicts a theorem claimed in [4]. However, the forwarding property holds in SPPC and the buffer-free PIOA model for most protocols of interest. In these cases, $\mathsf{UC\text{-}WBB}_{env}$ implies $\mathsf{SS\text{-}SBB}$.
   (b) The security notions in $\mathsf{UC\text{-}WBB}_{sim}$ are strictly weaker than the notions in WBB_{adv}, and hence, the notions in $\mathsf{UC\text{-}WBB}_{env}$ and $\mathsf{SS\text{-}SBB}$. In particular, the Universal Composability relation with placements of the master process as considered in [22] does neither imply the Strong Blackbox relations with placements of the master process as considered in [4, 22] nor Universal Composability relations with placements of the master process as considered in [8, 4].

These results all show that the relationship between universal composability and black-box simulatability is more subtle than previously described. One consequence is that when proving compositional security properties by a black-box reduction, care must be taken to make sure that the computational model gives appropriate power to the environment. In particular, the composability theorem of Canetti [8] does not imply that blackbox simulatability is a composable security notion, over any computational model in which the forwarding property (expressed by the FORWARDER axiom) is not satisfied.

Outline of the paper. Section 2 defines the sequential polynomial-time process calculus SPPC, with security relations defined precisely in Section 3. The main results are given in Section 4, with consequences for PIOA and PITM models developed in Section 5. In Section 6, we briefly consider a less prominent security notion, called *reactive simulatability* in [5] and *security with respect to specialized simulators* in [9], and relate it to the other notions.

Full definitions, proofs, and further explanations are provided in a technical report [14]. This technical report also proves a composition theorem for SPPC that is similar to the composition theorem for ITMs established by Canetti [8].

2 Sequential Probabilistic Process Calculus

In this section, we introduce Sequential Probabilistic Process Calculus (SPPC) as a language-based computational model for studying security notions (see [14] for

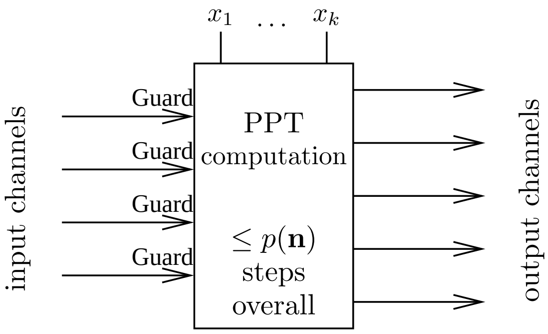

Fig. 2. Probabilistic polynomial-time machines in SPPC

a detailed technical presentation and further explanation). We start by discussing how individual probabilistic polynomial-time machines are modelled in SPPC and then explain how to build and execute systems of interacting machines. Our exposition parallels that of related models [8, 22, 5].

Single probabilistic polynomial-time machines. In SPPC, single machines are of the form as depicted in Figure 2. For the time being, let us ignore the "guards" and the variables $x_1, \ldots, x_k$. Conceptually, a single machine is a black-box with internal state that receives inputs, performs polynomially-bounded computation and then produces outputs. Inputs are received on input channels and outputs are written on output channels. More precisely, single machines are restricted to receiving one input and producing at most one output at a time. While this at first might appear to be a restriction, it is not really a problem since any machine that sends multiple messages can be converted to a machine that stores internally (possibly using internal buffers) the messages it wants to send, and then sends the messages one at a time on request. In fact, this style of communication corresponds exactly to the manner in which communication is defined in other *sequential* models, notably the PIOA and PITM models [8, 22]. Also, just as in these models, the overall runtime of a machine is bounded by a polynomial in the security parameter and does not depend on the number or length of inputs sent to the machine.

The channels of a single machine in SPPC correspond to ports in the PIOA model and to tapes in the PITM model. However, while messages on channels (and ports) are removed when read, this is not the case for tapes. Nevertheless, tapes can be modelled by adding machines, one for each input channel, which simulate the tapes in the obvious way. The "main machine" will then receive its input from the "tape machines". In the PIOA model, buffer machines serve a similar purpose. Note that while in SPPC and the PIOA model, the number of input and output channels/ports is not restricted, in Canetti's PITM model only one pair of input/output and input/output communication tapes is considered.

In SPPC, machines can preprocess their input using *guards* (see Figure 2) which are deterministic polynomial-time machines that are placed on input channels. Given an input on the channel, a guard may accept or reject the input. If rejected, the process does no computation. If accepted, the process receives the output of the guard. This may be different from the input, e.g., a guard can eliminate unnecessary information or transform data. The computation performed by the guard may depend on the current internal state of the process. Its runtime is polynomially-bounded in the security parameter per invocation and is not factored into the overall runtime of the process using the guard. In particular, a guard can be invoked an unbounded number of times. Since guards allow a process to discard messages without incurring a computation cost, attempts to "exhaust" a process by sending many useless messages to the process can be defeated. Additionally, using guards we can simulate an unbounded number of "virtual" channel names by prefixing each message with a session id and/or party name and then stipulating that the guards accept only those messages with the right header information. Such an ability is required for systems with a polynomial number of machines, e.g., multiparty protocols, or with multiple instances of the same protocol. While mechanisms analogous to guards are absent in other models, notably [22, 8], a newer version of PIOA [6] has a length function that, when set to zero, prevents messages from being received by the machine. This corresponds to a guard which rejects all inputs and so can be used to help avoid exhaustion attacks. However, it does not help in the creation of a mechanism analogous to virtual channels.

As mentioned above, guards can be invoked an unbounded number of times without being exhausted and in every invocation their runtime is bounded by a polynomial in the security parameter—the runtime could even depend on the length of the input. Hence, the runtime of a single machine including the guards is polynomially bounded in the security parameter *and* the number of invocations. However, the overall runtime of a single machine excluding the guards is polynomially bounded in the security parameter alone, and hence, such a machine can produce at most polynomially many output messages overall in the security parameter. Now, since guards can only be triggered by messages sent by single machines, it follows that in a system of polynomially many machines guards are only invoked a polynomial number of times in the security parameter. As shown in [14], from this we can conclude that such systems can be simulated by a probabilistic polynomial time Turing machine.

In SPPC, a machine may have auxiliary input, just like auxiliary input can be given to the interacting Turing machines in Canetti's model. This input is written on specific tapes before a (system of) machines is run. If such auxiliary input is used, it results in a non-uniform computational model. The tapes are represented by $x_1, \ldots, x_k$ (see Figure 2). Just like in Canetti's model, we only allow the environment machine to use auxiliary input. However, whether the environment machine is uniform or not does not affect the results presented in this paper.

Formally, in SPPC a single machine is defined by a *process expression* $\mathcal{P}$. Such an expression corresponds to a description of an interacting Turing machine in the PITM model or an I/O automaton in the PIOA model. A process expression is always parameterized by the security parameter $\mathbf{n}$ and possibly so-called free variables $x_1, \ldots, x_k$, which represent the tapes for the auxiliary input mentioned above. Therefore, we sometimes write $\mathcal{P}(x_1, \ldots, x_k)$ instead of $\mathcal{P}$. A process expression with value i chosen for the security parameter and values $\vec{a}$ (the auxiliary inputs) substituted for its free variables $\vec{x}$ yields a *process* $\mathcal{P}(\vec{a})^{\mathbf{n} \leftarrow i}$. A process corresponds to an interacting Turing machine where the security parameter is written on the security parameter tape and the auxiliary input is written on the input tape. Hence, a process can perform computations as soon as it receives input on the input channels. As an expositional convenience, we will use the terms 'process expression' and 'process' interchangeably. A process expression is called *open* if it has free variables, and *closed* otherwise. Hence, open process expressions correspond to non-uniform machines and closed expressions to uniform ones.

Systems of interacting machines. In SPPC, a system of interacting machines is simply a multiset of single machines where an output channel of one machine connects directly to an identically-named input channel of another machine. The manner in which these machines are wired together is uniquely determined by the channel names since we stipulate that no two machines have the same input and output channel names respectively. After a machine M_1 has sent a message on an output channel, the machine waits to receive input on an input channel. The message sent on the output channel is immediately received by the machine M_2 that has an identically-named input channel. If the guards on the input channel of this machine accepts the message, then M_2 may perform some computation and produce one output message. While M_2 now waits for new input on its input channels, the output message (if any) is processed by the next receiving machine, and so on. If there is no receiving machine, or the guard of the receiving machine rejects the message, or no output message is produced, computation would halt since no machine is triggered. To avoid this, in a system of machines, one machine is always declared to be a master machine, also called *master process*, and this machine is triggered if no other machine is.

In SPPC, given process expressions $\mathcal{P}_1, \ldots, \mathcal{P}_n$, each representing a single machine, the combined system of machines is denoted by the process expression $\mathcal{P}_1 \upharpoonright \cdots \upharpoonright \mathcal{P}_n$. Instead of interpreting $\mathcal{P}_1 \upharpoonright \cdots \upharpoonright \mathcal{P}_n$ as a system of n single machines, one can consider this system as a single machine (consisting of n submachines). This corresponds to the transformation, in the PIOA model, of a system of fixed, finite number of machines into a single machine. However, in SPPC we can apply such transformations to systems containing a polynomial number of machines as well.

With the bounded replication operator $!_{q(\mathbf{n})}\, \mathcal{P}$, where $q(\mathbf{n})$ is some polynomial in the security parameter and $\mathcal{P}$ is a process expression (representing a single machine or a system of machines), systems containing a polynomial number of machines can be described. The process expression $!_{q(\mathbf{n})}\, \mathcal{P}$ stands for a $q(\mathbf{n})$-fold parallel composition $\mathcal{P} \upharpoonright \cdots \upharpoonright \mathcal{P}$. Note that in such a system, different copies of $\mathcal{P}$ have the same input and output channels. However, as discussed earlier, guards allow us to send messages to (virtual) channels of particular copies of a protocol. Bounded replication can be combined with parallel composition to build bigger systems such as $!_{q_1(\mathbf{n})}\ (\mathcal{P}_1 \mid \mathcal{P}_2 \mid !_{q_3(\mathbf{n})}\ \mathcal{P}_3)$.

We note that the details of the communication model, such as a specific activation order of entities, the communication primitives available (such as insecure, authenticated, or secure channels), and specific forms of buffering, are not explicitly modelled in SPPC. The driving philosophy behind the design of SPPC is to move such details into the specification of the protocol rather than explicitly encoding them into the model. This makes SPPC simple and flexible, thereby allowing easy formulation of a variety of security notions.

As described earlier, since our execution model is sequential, computation may not proceed if currently executing machine produces no output, or a receiving machine rejects an input. In order to ensure that computation proceeds even in this case, we identify a master process by using a special input channel `start`. In case no output is produced by a machine, a fixed value is written on `start` thereby triggering the master process. The master process is also the first machine to be activated when execution starts.

Additionally, in studying security notions, it will be useful to define the output of a system. We do so by writing a bit, the output, onto an output channel named `decision`. The machine containing this channel is called the *decision process*. Given a process expression $\mathcal{R}(\overrightarrow{x})$ with free variables $\overrightarrow{x}$, we denote by $\texttt{Prob}[\mathcal{R}(\overrightarrow{a})^{\mathbf{n}\leftarrow i} \rightsquigarrow 1]$ the probability that $\mathcal{R}$ with security parameter i and substitution of values $\overrightarrow{a}$ for its variables $\overrightarrow{x}$ outputs a 1 on `decision`. Recall that $\mathcal{R}(\overrightarrow{a})^{\mathbf{n}\leftarrow i}$ denotes the process obtained from the process expression $\mathcal{R}$ by replacing the security parameter $\mathbf{n}$ by a value i and replacing the variables $\overrightarrow{x}$ by values $\overrightarrow{a}$. Two process expressions $\mathcal{P}(\overrightarrow{x})$ and $\mathcal{Q}(\overrightarrow{x})$ are called *equivalent* or *indistinguishable*, written $\mathcal{P}(\overrightarrow{x}) \equiv \mathcal{Q}(\overrightarrow{x})$, iff for every polynomial $p(\mathbf{n})$ there exists i_0 such that $|\texttt{Prob}[\mathcal{P}(\overrightarrow{a})^{\mathbf{n}\leftarrow i} \rightsquigarrow 1]-\texttt{Prob}[\mathcal{Q}(\overrightarrow{a})^{\mathbf{n}\leftarrow i} \rightsquigarrow 1]| \leq 1/p(i)$ for every $i \geq i_0$ and every tuple $\overrightarrow{a}$ of bit strings.

We call machines which are neither master nor decision processes *regular*. A machine which is both master and decision is called a *master decision process*. In what follows, by **R**, **M**, **D**, and **MD** we denote the set of all closed regular processes, closed master processes, open or closed decision processes, and open or closed master decision processes, respectively.

3 The Security Notions and Their Variants

In this section, we formulate the security notions and their variants. In order to do so, we must first define which SPPC expressions constitute well-formed systems of interacting machines. We will do so by specifying how machines are connected together.

We start by defining the communication interfaces of individual processes. A process uses directional external channels—*input* and *output external channels*—to communicate with other machines. These channels are partitioned into two types: *network channels* and *IO channels*. The channels `start` and `decision` are respectively defined to be input and output IO channels. Input channels connect to identically-named output channels of the same type.

If $\mathcal{P}$ represents a protocol, then an adversary $\mathcal{A}$ connects to the network channels of $\mathcal{P}$ while an environment $\mathcal{E}$ connects to the IO channels of $\mathcal{P}$ and $\mathcal{A}$. Formally, two processes, $\mathcal{P}$ and $\mathcal{Q}$, are *compatible* if they have the same set of external channels. We say that two processes are *IO-compatible* if they have the same set of IO channels and disjoint sets of network channels. A process expression $\mathcal{Q}$ is *connectible* for $\mathcal{P}$ if each common external channel of $\mathcal{P}$ and $\mathcal{Q}$ has the same type in both and complementary directions. A process expression $\mathcal{A}$ is *adversarially connectible* for $\mathcal{P}$ if $\mathcal{A}$ is connectible for $\mathcal{P}$ and the set of external channels of $\mathcal{A}$ is disjoint from the set of IO channels of $\mathcal{P}$. Thus an adversary can only connect on the network channels of a protocol. Similarly, $\mathcal{E}$ is *environmentally connectible* for $\mathcal{P}$ if it can only connect on the IO channels of $\mathcal{P}$.

We can now define what it means for a process to be an adversary, environment, or simulator. We do so in a parametric fashion so that we can succinctly represent the variants of a security notion. Given a set of processes $\mathbf{C}$ we define: a) $Env_{\mathbf{C}}(\mathcal{P})$ to be the set of all processes in $\mathbf{C}$ that are environmentally connectible for $\mathcal{P}$, b) $Adv_{\mathbf{C}}(\mathcal{P})$ to be the set of all processes in $\mathbf{C}$ that are adversarially connectible for $\mathcal{P}$, c) $Sim_{\mathbf{C}}(\mathcal{P}, \mathcal{F})$ to be the set of all processes $\mathcal{S}$ in $\mathbf{C}$ that are adversarially connectible for $\mathcal{F}$ and such that $\mathcal{S} \rceil \mathcal{F}$ is compatible with $\mathcal{P}$, d) $Con_{\mathbf{C}}(\mathcal{P})$ to be the set of all processes in $\mathbf{C}$ that are connectible for $\mathcal{P}$.

Definition 1. *Let* $\mathbf{A}$ *(real adversaries),* $\mathbf{I}$ *(ideal adversaries),* $\mathbf{E}$ *(environments), and* $\mathbf{S}$ *(simulators) be sets of process expressions, and* $\mathcal{P}$ *(the real protocol) and* $\mathcal{F}$ *(the ideal functionality/protocol) be IO-compatible process expressions.*

Strong Simulatability (SS): $\mathtt{SS}_{(\mathbf{S},\mathbf{E})}(\mathcal{P}, \mathcal{F})$ *iff* $\exists \mathcal{S} \in Sim_{\mathbf{S}}(\mathcal{P}, \mathcal{F}) \forall \mathcal{E} \in Con_{\mathbf{E}}(\mathcal{P}) : \mathcal{E} \rceil \mathcal{P} \equiv \mathcal{E} \rceil \mathcal{S} \rceil \mathcal{F}$*, i.e., there exists a simulator such that no environment can distinguish whether it is interacting with the real protocol or the ideal functionality-simulator combination.*

Strong Blackbox Simulatability (SBB): $\mathtt{SBB}_{(\mathbf{A},\mathbf{S},\mathbf{E})}(\mathcal{P}, \mathcal{F})$ *iff* $\exists \mathcal{S} \in Sim_{\mathbf{S}}(\mathcal{P}, \mathcal{F}) \forall \mathcal{A} \in Adv_{\mathbf{A}}(\mathcal{P}) \forall \mathcal{E} \in Env_{\mathbf{E}}(\mathcal{A} \rceil \mathcal{P}) : \mathcal{E} \rceil \mathcal{A} \rceil \mathcal{P} \equiv \mathcal{E} \rceil \mathcal{A} \rceil \mathcal{S} \rceil \mathcal{F}$*, i.e., there exists a simulator such that for all adversaries, no environment can distinguish whether it is interacting with the real protocol-adversary combination or the ideal functionality-simulator-adversary combination.*

Weak Blackbox Simulatability (WBB): $\mathtt{WBB}_{(\mathbf{A},\mathbf{S},\mathbf{E})}(\mathcal{P}, \mathcal{F})$ *iff* $\forall \mathcal{A} \in Adv_{\mathbf{A}}(\mathcal{P}) \exists \mathcal{S} \in Sim_{\mathbf{S}}(\mathcal{P}, \mathcal{F}) \forall \mathcal{E} \in Env_{\mathbf{E}}(\mathcal{A} \rceil \mathcal{P}) : \mathcal{E} \rceil \mathcal{A} \rceil \mathcal{P} \equiv \mathcal{E} \rceil \mathcal{A} \rceil \mathcal{S} \rceil \mathcal{F}$*, i.e., for each adversary there exists a simulator such that no environment can distinguish whether it is interacting with the real protocol-adversary combination or the ideal functionality-simulator-adversary combination.*

Universal Composability (UC): $\mathtt{UC}_{(\mathbf{A},\mathbf{I},\mathbf{E})}(\mathcal{P}, \mathcal{F})$ *iff* $\forall \mathcal{A} \in Adv_{\mathbf{A}}(\mathcal{P})\ \exists \mathcal{I} \in Sim_{\mathbf{I}}(\mathcal{A} \rceil \mathcal{P}, \mathcal{F})\ \forall \mathcal{E} \in Env_{\mathbf{E}}(\mathcal{A} \rceil \mathcal{P}) : \mathcal{E} \rceil \mathcal{A} \rceil \mathcal{P} \equiv \mathcal{E} \rceil \mathcal{I} \rceil \mathcal{F}$*, i.e., for each real adversary there exists an ideal adversary such that no environment can distinguish whether it is interacting with the real protocol-real adversary combination or the ideal functionality-ideal adversary combination.*

4 Relationships Between the Security Notions

In this section, we examine the relationships between the security notions introduced in the previous section. The instances of each security notion are obtained by assigning roles (decision, master, regular) to the various entities (environment, real and ideal adversary, simulator, real and ideal protocol). The environment is always the decision process, and the real and ideal protocols are always regular processes. So, the variants of each security notion differ only wrt the entity that assumes the role of the master process. Formally, the variants are obtained by defining the sets $\mathbf{A}$, $\mathbf{I}$, $\mathbf{S}$, and $\mathbf{E}$ to be one of the sets $\mathbf{R}$, $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{MD}$. For example, the security notion considered in [8] is $\mathtt{UC}_{(\mathbf{R},\mathbf{R},\mathbf{MD})}(\mathcal{P},\mathcal{F})$. Here, $\mathcal{P}$ and $\mathcal{F}$ are UC with the environment as the master decision process and the real and ideal adversary as regular processes. Another variant of UC considered in [4] is $\mathtt{UC}_{(\mathbf{M},\mathbf{M},\mathbf{MD})}(\mathcal{P},\mathcal{F})$. Here, the environment is the master decision process and the real and ideal adversaries are the master processes. Notice that although both the environment $\mathcal{E}$ and the real/ideal adversary $\mathcal{A}/\mathcal{I}$ *may* play the role of the master process, in any specific setting, according to our definitions, exactly one of $\mathcal{E}$ or $\mathcal{A}/\mathcal{I}$ will actually be the master process. If the real adversary is the master process, then the ideal adversary must be master as well; furthermore, the environment cannot be master (as it would not be environmentally valid if it also had the `start` channel). Conversely, if the adversary is not the master (i.e., does not have the `start` channel), then the environment may be the master. Note that combinations without a master process or a decision process do not make sense since no computation can take place in their absence or no decision can be generated, i.e., no process can write on the channel `decision`. Henceforth, we omit such combinations. We will consider combinations where we *require* a certain entity to play the role of the master by saying that this entity is a process expression in $\mathbf{M} \setminus \mathbf{R}$. In variants of these notions where the simulator $\mathcal{S}$ plays the role of the master process, we allow the simulator to hand over control to $\mathcal{A}$ or $\mathcal{E}$ via a channel $\mathtt{start}'$, which replaces `start` in $\mathcal{A}$ and $\mathcal{E}$. Additionally, we also consider a variant of WBB where the simulator $\mathcal{S}$ only depends on the complexity of $\mathcal{A}$ rather than on $\mathcal{A}$ in its entirety. We can easily show that these two variants are equivalent [14], and so we will not distinguish between them in what follows.

In the following theorems, the security notions are considered to be binary relations over the set of regular processes $\mathbf{R}$.

Theorem 1. *All variants of Strong Simulatability and Strong Blackbox obtained by varying the entity that is the master process are equivalent, i.e., the following identities hold:*

$$\begin{aligned}&\mathsf{SS}_{(\mathbf{R},\mathbf{MD})} = \mathsf{SS}_{(\mathbf{M},\mathbf{MD})} = \mathsf{SBB}_{(\mathbf{R},\mathbf{R},\mathbf{MD})} = \mathsf{SBB}_{(\mathbf{M},\mathbf{R},\mathbf{MD})} = \\ &\mathsf{SBB}_{(\mathbf{M},\mathbf{R},\mathbf{D})} = \mathsf{SBB}_{(\mathbf{M}\setminus\mathbf{R},\mathbf{R},\mathbf{D})} = \mathsf{SBB}_{(\mathbf{M}\setminus\mathbf{R},\mathbf{R},\mathbf{MD})} = \mathsf{SBB}_{(\mathbf{M},\mathbf{M},\mathbf{MD})} = \\ &\mathsf{SBB}_{(\mathbf{R},\mathbf{M},\mathbf{MD})} = \mathsf{SBB}_{(\mathbf{M},\mathbf{M},\mathbf{D})}.\end{aligned}$$

We call this class of security notions $\mathsf{SS\text{-}SBB}$. It includes placements of the master process as considered for Strong Blackbox in [4] and [22]. In [4], the environment, the adversary, and the simulator may play the role of the master process, and hence, this corresponds to the notion $\mathtt{SBB}_{(\mathbf{M},\mathbf{M},\mathbf{MD})}$. In [22], only the adversary and the simulator may be the master process, but not the environment, and hence, this corresponds to the notion $\mathtt{SBB}_{(\mathbf{M},\mathbf{M},\mathbf{D})}$.

Recall that the difference between SS and SBB is that, in the latter notion, the environment and the adversary are separate entities, while in the former they are combined into one. Since the adversary and the environment can communicate freely, it is perhaps expected that the two notions should be equivalent. Theorem 1 bears out this intuition, and shows that the equivalences among the notions are independent of which entity plays the role of the master process. Consequently, there appears to be no technical benefit from treating the adversary and environment as two separate entities as in the SBB setting. We point out that in order to prove Theorem 1, it is important that situations in which the simulator is a master process do not differentiate between SS and SBB. This is true because such situations yield degenerate relations; for instance, $\mathtt{SS}_{(\mathbf{M}\setminus\mathbf{R},\mathbf{MD})}$ is an empty relation. The reason is that the environment can exhaust a master simulator since whenever execution defaults to the simulator, it triggers the environment immediately. Hence the environment can repeatedly "ping" the simulator until its time-bound is exhausted. In a model in which the runtime of the simulator may depend on how often it is invoked by the environment, such exhaustion attacks would not be viable. In such a model, the notions SS and SBB may differ depending on whether or not the simulator plays the role of the master. Since such extensions are not studied in published work, we defer a detailed study to future work.

Theorem 2. *All variants of Universal Composability and Weak Blackbox in which the environment may be the master process are equivalent, i.e., the following identities hold:*

$$\begin{aligned}\mathtt{UC}_{(\mathbf{R},\mathbf{R},\mathbf{MD})} &= \mathtt{UC}_{(\mathbf{M},\mathbf{M},\mathbf{MD})} = \mathtt{WBB}_{(\mathbf{R},\mathbf{R},\mathbf{MD})} = \\ \mathtt{WBB}_{(\mathbf{M},\mathbf{R},\mathbf{MD})} &= \mathtt{WBB}_{(\mathbf{M},\mathbf{M},\mathbf{MD})} = \mathtt{WBB}_{(\mathbf{R},\mathbf{M},\mathbf{MD})}.\end{aligned}$$

We call this class of security notions $\mathsf{UC\text{-}WBB}_{env}$. It includes placements of the master process as considered for Universal Composability in [8] and [4]. While in [8] only the environment may play the role of the master process, corresponding to the notion $\mathtt{UC}_{(\mathbf{R},\mathbf{R},\mathbf{MD})}$, in [4] the adversary may play this role as well, corresponding to $\mathtt{UC}_{(\mathbf{M},\mathbf{M},\mathbf{MD})}$.

The fact that WBB implies UC follows simply by combining the simulator and real adversary to produce an ideal adversary. To go in the reverse direction, we consider what happens when instantiating the real adversary with a process that simply forwards messages between the protocol and the environment. The corresponding ideal adversary then serves as the simulator in the definition of WBB. We note that it is important that the runtime of the simulator be allowed to depend on the complexity of the real adversary. Note that as in the class $\mathsf{SS\text{-}SBB}$, equivalence among the security notions in $\mathsf{UC\text{-}WBB}_{env}$ holds independently of whether or not the simulator may be the master process.

Theorem 3. *All variants of Universal Composability and Weak Blackbox in which the simulator and the adversary may be the master process and the environment is not the master process are equivalent, i.e., the following identities hold:*

$$\mathtt{UC}_{(\mathbf{M},\mathbf{M},\mathbf{D})} = \mathtt{UC}_{(\mathbf{M}\setminus\mathbf{R},\mathbf{M}\setminus\mathbf{R},\mathbf{D})} = \mathtt{UC}_{(\mathbf{M}\setminus\mathbf{R},\mathbf{M}\setminus\mathbf{R},\mathbf{MD})} = \mathtt{WBB}_{(\mathbf{M},\mathbf{M},\mathbf{D})} = \mathtt{WBB}_{(\mathbf{M}\setminus\mathbf{R},\mathbf{M},\mathbf{D})}.$$

We call this class of security notions $\mathsf{UC\text{-}WBB}_{sim}$. It includes the placements of the master process as considered for Universal Composability in [22], which corresponds to the notion $\mathtt{UC}_{(\mathbf{M},\mathbf{M},\mathbf{D})}$.

Equivalence among the notions in $\mathsf{UC\text{-}WBB}_{sim}$ is established similarly to the class $\mathsf{UC\text{-}WBB}_{env}$. Note that $\mathsf{UC\text{-}WBB}_{sim}$ does not contain a version of WBB where the simulator is restricted to be regular. As we will see, restricting the simulator in this way, yields a strictly stronger notion.

Theorem 4. *All variants of Weak Blackbox where the adversary is the master process and neither the environment nor the simulator is the master process are equivalent, i.e., the following identities hold:* $\mathtt{WBB}_{(\mathbf{M},\mathbf{R},\mathbf{D})} = \mathtt{WBB}_{(\mathbf{M}\setminus\mathbf{R},\mathbf{R},\mathbf{D})}$.

We call this class of security notions WBB_{adv}. We now study the relationships between the four classes. We say that a class of equivalent security notions $\mathcal{C}$ implies another class $\mathcal{C}'$ of equivalent security notions ($\mathcal{C} \Rightarrow \mathcal{C}'$), if a notion in $\mathcal{C}$ (and hence, every notion in $\mathcal{C}$) implies a notion in $\mathcal{C}'$ (and hence, every notion in $\mathcal{C}'$).

Theorem 5. *$\mathsf{SS\text{-}SBB} \Rightarrow \mathsf{UC\text{-}WBB}_{env} \Rightarrow \mathsf{WBB}_{adv} \Rightarrow \mathsf{UC\text{-}WBB}_{sim}$, but the class $\mathsf{UC\text{-}WBB}_{sim}$ does not imply the other classes, i.e., $\mathsf{UC\text{-}WBB}_{sim} \not\Rightarrow \mathsf{WBB}_{adv}$, and hence, $\mathsf{UC\text{-}WBB}_{sim} \not\Rightarrow \mathsf{UC\text{-}WBB}_{env}$ and $\mathsf{UC\text{-}WBB}_{sim} \not\Rightarrow \mathsf{SS\text{-}SBB}$*

In particular, we have that the Strong Blackbox relation with the placements of the master process as considered in [22, 4] implies the Universal Composability relation with the placements of the master process as considered in [8, 22, 4]. Also, the Universal Composability relation with the placement of the master process as considered in [22] is strictly weaker than the Universal Composability relation with the placements of the master process as considered in [8] and [4].

The argument from $\mathsf{SS\text{-}SBB}$ to $\mathsf{UC\text{-}WBB}_{env}$ relies on the order of quantification over the entities. The fact that $\mathsf{UC\text{-}WBB}_{env}$ implies WBB_{adv} relies on the observation that making the environment the master intuitively gives the environment more discriminatory power. The final implication follows from the fact that the set of simulators considered in WBB_{adv} is a subset of the set of simulators considered in $\mathsf{UC\text{-}WBB}_{sim}$.

To show that $\mathsf{UC\text{-}WBB}_{sim} \not\Rightarrow \mathsf{WBB}_{adv}$ we provide a concrete example. Consider a protocol $\mathcal{P}$ that receives a bit on an IO channel and forwards it on a network channel. The ideal functionality $\mathcal{F}$ does the same but only forwards the bit if it is 1. We can show that $\mathtt{UC}_{(\mathbf{M},\mathbf{M},\mathbf{D})}(\mathcal{P},\mathcal{F})$ but not $\mathtt{WBB}_{(\mathbf{M},\mathbf{R},\mathbf{D})}(\mathcal{P},\mathcal{F})$. It is open whether WBB_{adv} implies $\mathsf{UC\text{-}WBB}_{env}$.

The following theorem identifies a necessary and sufficient condition—the forwarder axiom (defined below)—for the equivalence of $\mathsf{UC\text{-}WBB}_{env}$ and $\mathsf{SS\text{-}SBB}$.

As a consequence, we can see that the strongest variant of UC (where the environment is the master process) implies SBB just when the forwarder axiom holds.

Theorem 6. *Let* $\mathbf{C}$ *be a class of regular processes closed under channel-renaming. Then, restricting the relations to* $\mathbf{C}$*, we obtain that*

$$(\mathsf{UC\text{-}WBB}_{env} \implies \mathsf{SS\text{-}SBB}) \textit{ iff } (\textit{FORWARDER holds for all processes in } \mathbf{C}).$$

In particular, the Universal Composability relations with the placements of the master process as considered in [8] and [4] are strictly weaker than the Strong Blackbox relations with the placements of the master process as considered in [4] and [22] in any computational model in which the forwarding property given by the FORWARDER axiom does not hold. The axiom FORWARDER($\mathbf{C}$) for a class of regular processes $\mathbf{C}$ is stated as follows:

FORWARDER($\mathbf{C}$) Given any process $\mathcal{P} \in \mathbf{C}$ with network channels net, there exists a process $\mathcal{D}$ with network channels $net \cup (net' = \{c' \mid c \in net\})$ such that for all $\mathcal{E}$ whose only shared channels with $\mathcal{P}$ are external channels of $\mathcal{P}$:

$$\mathcal{E} \upharpoonleft \mathcal{P} \equiv \mathcal{E} \upharpoonleft \mathcal{D} \upharpoonleft [net'/net]\mathcal{P}$$

where $[net'/net]\mathcal{P}$ denotes the process obtained from $\mathcal{P}$ by replacing the channels in net by those in net'.

Intuitively, this axiom allows us to invisibly plug a communication medium $\mathcal{D}$ between two entities connected over network channels.

While the forwarder property appears believable, it turns out that for arbitrary protocols, the forwarder property does not hold. The problem lies in the fact that the forwarder is chosen independently of the environment $\mathcal{E}$. As a result its runtime is fixed a priori and the environment can exhaust the forwarder by sending it many useless messages. Then, the presence of the forwarder can be easily detected by the environment. All is not lost. For a class of protocols including those commonly studied in the literature (c.f., [22, 8, 5]) the forwarder property holds in SPPC. We shall refer to these protocols as *standard* protocols. The specific way in which the exhaustion problem is avoided for standard protocols involves using guards to reject the spurious messages. We cannot do this for every protocol because it is important that the forwarder knows the communication structure of the protocol (see [14] for details). The following corollary to Theorem 6 is now immediate.

Corollary 1. $\mathsf{SS\text{-}SBB} \Leftrightarrow \mathsf{UC\text{-}WBB}_{env}$ *for the class of standard protocols.*

In particular, the Strong Blackbox relation with the placement of the master process as considered in [22, 4] and the Universal Composability relation with the placement of the master process as considered in [8, 4] are equivalent for standard protocols in SPPC (see also Section 5.1).

Although in this extended abstract, we have only given intuitions behind some of the proofs, we emphasize that the actual proofs are carried out using

$\mathcal{P} \upharpoonleft \mathcal{Q} \equiv \mathcal{Q} \upharpoonleft \mathcal{P}$	COM
$\mathcal{P} \upharpoonleft (\mathcal{Q} \upharpoonleft \mathcal{R}) \equiv (\mathcal{P} \upharpoonleft \mathcal{Q}) \upharpoonleft \mathcal{R}$	ASC
$\mathcal{P} \equiv \mathcal{Q}, \mathcal{Q} \equiv \mathcal{R} \implies \mathcal{P} \equiv \mathcal{R}$	TRN
$\mathcal{P} \equiv \mathcal{Q} \implies \mathcal{Q} \equiv \mathcal{P}$	SYM
$\mathcal{P} \equiv [d/c]\mathcal{P}$ where $c, d \notin \{\texttt{start}, \texttt{decision}\}$, $d \notin \mathit{Channels}(\mathcal{P})$	RENAME

Fig. 3. A representative fragment of SPPC's reasoning system

an equational reasoning system for SPPC. A small representative fragment of the axiom system is given below. These axioms capture simple structural properties like commutativity, associativity, transitivity, and symmetry of process equivalence. It also allows structural operations such as channel-renaming.

These axioms can also serve as an abstract specification of a "reasonable" computational model for simulation-based security.

5 Implications for Other Models

We now study the relationships of the security notions for the PIOA model [22, 5, 4] and the PITM model [8, 9]. The simplicity of SPPC's axiom system enables us to carry over our results to these other computational models. Furthermore, the counter-examples used to demonstrate that certain notions are strictly stronger than others are quite simple and easily translate into the related models.

5.1 The PIOA Model

Most of the axioms used to prove the relationships among the security notions also hold in the different versions of the PIOA model. Therefore, the relationships given in the previous section mostly carry over to PIOA. In particular, we obtain that all the security notions in SS-SBB, UC-WBB$_{sim}$, and WBB$_{adv}$ are equivalent, respectively, and their relationships are as depicted in Figure 1.

However, an axiom used to prove that the security notions in UC-WBB$_{env}$ are equivalent does not hold in PIOA. This axiom captures a property similar to the forwarder property discussed in Section 4. This axiom essentially states that there exists a forwarder $\mathcal{D}$ that is allowed to depend on the complexity of the protocol $\mathcal{P}$ *and* the adversary $\mathcal{A}$ such that $\mathcal{E} \upharpoonleft \mathcal{A} \upharpoonleft \mathcal{P} \equiv \mathcal{E} \upharpoonleft \mathcal{A}' \upharpoonleft \mathcal{D} \upharpoonleft \mathcal{P}$ (where $\mathcal{A}'$ is $\mathcal{A}$ with some renamed network channels). The axiom fails in PIOA because machines always have to communicate through buffers and buffers are triggered by machines other than the one writing into the buffer. In fact, we show that UC does *not* imply WBB in PIOA when the environment is master. This failure of equivalence seems counterintuitive. The problem vanishes if the PIOA model is modified so that machines always trigger their own buffers. In effect, this is equivalent to not having buffers at all, which is why we call this fragment of the PIOA model the *buffer-free PIOA model (BFPIOA)*. This fragment is essentially as expressive as PIOA and this fragment can be embedded into SPPC (see [14]).

In particular, *all* axioms (except the forwarder property of the previous section) are satisfied in BFPIOA and the examples used to prove separation results can also be expressed in BFPIOA. As mentioned in Section 2, starting from the work [6] PIOA (and thus, BFPIOA) has a restricted form of guards. Similar to SPPC, this mechanism suffices to satisfy the forwarder property for standard protocols, but just as in SPPC, there are protocols expressible in BFPIOA which do not satisfy this property. In summary, we obtain for BFPIOA exactly the same relationships as for SPPC (see Figure 1).

In [22], the security notions $\mathtt{UC}_{(\mathbf{M},\mathbf{M},\mathbf{D})}(\mathcal{P},\mathcal{F})$ and $\mathtt{SBB}_{(\mathbf{M},\mathbf{M},\mathbf{D})}(\mathcal{P},\mathcal{F})$ were introduced for the PIOA model, while in [4] the notions $\mathtt{UC}_{(\mathbf{M},\mathbf{M},\mathbf{MD})}(\mathcal{P},\mathcal{F})$ and $\mathtt{SBB}_{(\mathbf{M},\mathbf{M},\mathbf{MD})}(\mathcal{P},\mathcal{F})$ were considered. Our results clarify the relationships between these security notions: while the two variants of SBB are equivalent (they both belong to the class SS-SBB), these notions are different from the two variants of UC. Also, the two variants of UC are not equivalent. Our results contradict the claim in [4] that $\mathtt{SBB}_{(\mathbf{M},\mathbf{M},\mathbf{MD})}(\mathcal{P},\mathcal{F})$ and $\mathtt{UC}_{(\mathbf{M},\mathbf{M},\mathbf{MD})}(\mathcal{P},\mathcal{F})$ are equivalent.

5.2 The PITM model

The PITM model [8] is tailored towards defining UC where the environment is a master process and the adversaries are regular processes i.e., $\mathtt{UC}_{(\mathbf{R},\mathbf{R},\mathbf{MD})}(\mathcal{P},\mathcal{F})$. Depending on which entities are involved, different computational models are defined: the real model (involving the environment, the real adversary, and the real protocol), the ideal model (involving the environment, the ideal adversary, and the ideal functionality together with dummy parties), and the hybrid model which is a combination of the previous two models.

Therefore, it is not immediately clear how the security notions SS, SBB, and WBB, which involve a simulator, would be defined in PITM. Different variants are possible, and as we have seen, differences in the definitions may affect the relationships between the security notions. It is out of the scope of this paper, to extend PITM in order to define SS, SBB, and WBB. However, some general points can be made. The version of PITM in [8] does not have a mechanism, like the guards of SPPC, that will enable the forwarder property to be satisfied. Without this property, UC is a strictly weaker notion than SBB. In ongoing work [9], Canetti allows PITMs to depend on the number of invocations as well as the length of messages on the IO tapes. This mechanism could enable PITM to satisfy the forwarder property whence UC would imply SBB. However, this is speculative since the details of the model are still being developed.

We finally note that in [8], Canetti introduces a special case of UC where the adversary merely forwards messages between the environment and the parties. Canetti proves UC and this notion equivalent. This notion can easily be formulated in SPPC and proved equivalent to UC along the lines of the proof which shows that $\mathtt{UC}_{(\mathbf{R},\mathbf{R},\mathbf{MD})}(\mathcal{P},\mathcal{F})$ implies $\mathtt{WBB}_{(\mathbf{M},\mathbf{R},\mathbf{MD})}(\mathcal{P},\mathcal{F})$.

6 Reactive Simulatability and Extensions of SPPC

In this section, we consider another security notion, called *reactive simulatability* in [5] and *security with respect to specialized simulators* in [9]. This notion has not drawn as much attention as the others studied in the present work because, to our best knowledge, a general composition theorem along the lines of [22, 4, 8, 14], has not been proved for reactive simulatability (see, however, [9]). Therefore, in the previous sections, we have concentrated on the other security notions and only very briefly cover reactive simulatability here. In our terminology, reactive simulatability is defined as follows:

Reactive Simulatability: $\mathtt{RS}_{(\mathbf{A},\mathbf{I},\mathbf{E})}(\mathcal{P},\mathcal{F})$ iff $\forall \mathcal{A} \in Adv_{\mathbf{A}}(\mathcal{P}) \forall \mathcal{E} \in Env_{\mathbf{E}}(\mathcal{A} \upharpoonright \mathcal{P})$ $\exists \mathcal{I} \in Adv_{\mathbf{I}}(\mathcal{F}) : \mathcal{E} \upharpoonright \mathcal{A} \upharpoonright \mathcal{P} \equiv \mathcal{E} \upharpoonright \mathcal{I} \upharpoonright \mathcal{F}$.

The only difference between reactive simulatability and universal composability (UC) is that in the former the ideal adversary is allowed to depend on the environment. It has been pointed out by Canetti [9] that reactive simulatability is equivalent to UC if the runtime of the environment may depend on the length of the message on its input tape.[3] In such a model, the notion of indistinguishability has to be slightly modified. The idea of the proof of equivalence is that one can define a *universal environment* which interprets part of its input as an encoding of another environment. The ideal adversary corresponding to this environment, in effect, works for all environments. It is straightforward to extend SPPC in a way that the runtime of *open* processes (recall that the environment is modeled as an open process), may depend on the length of the messages substituted for the free variables. Thus, the same proof also works in SPPC. We note that the argument goes through regardless of whether the environment may or may not play the role of the master process. In the former case, reactive simulatability is equivalent to the notions in the class $\mathsf{UC\text{-}WBB}_{env}$, and in the latter case, it is equivalent to the notions in $\mathsf{UC\text{-}WBB}_{sim}$. This result also carries over to an appropriate extension of BFPIOA.

7 Conclusion

We have carried out a thorough study of the relationships among various notions of simulation-based security, identifying two properties of the computational model that determine equivalence between these notions. Our main results are that all variants of SS (strong simulatability) and SBB (strong black box simulatability) are equivalent, regardless of the selection of the master process, and they imply UC (universal composability) and WBB (weak black box simulatability). Conditions UC and WBB are equivalent as long as the role (master process or not) of the environment is the same in both. However, the variant of UC in

[3] In his new model, Canetti allows every interacting Turing machine to depend on the number of invocations on input tapes and the length of the messages on input tapes. However, to prove the equivalence, it suffices to require this only for the environment.

which the environment may be a master process (as in [8, 4]) is strictly stronger than the variants in which the environment must not assume this role (as in [22]). In addition, the weaker forms of WBB do not imply SS/SBB. Finally, we prove a necessary and sufficient condition for UC/WBB to be equivalent to SS/SBB, based on the ability to define forwarders. These results all show that the relationship between universal composability and black-box simulatability is more subtle than previously described. In particular, the composability theorem of Canetti [8] does not necessarily imply that blackbox simulatability is a composable security notion over any computational model in which the forwarding property is not satisfied. Another technical observation is that making the environment the master process typically yields a stronger security notion. Hence, we recommend that in subsequent developments of the various models, the environment is always assigned the role of the master process.

Since our proofs are carried out axiomatically using the equational reasoning system developed for SPPC, we are able to apply the same arguments to suitably modified versions of the alternative computational models. We emphasize that the our suggested modifications to the other systems are motivated by the failure, in those systems, of simple equational principles. In particular, it seems reasonable to adopt a buffer-free variant of PIOA.

While our study concentrates on models where the runtime of processes is bounded by a polynomial in the security parameter, it would be interesting to consider those models where the runtime may depend on the number of invocations and the length of inputs (e.g., [9]). We believe that most of our results carry over also to these models as they seem to satisfy the axioms that we use in our proofs. However, the issue remains open since the details of these models have not yet been fixed.

Acknowledgments: We thank Michael Backes, Ran Canetti, Birgit Pfitzmann, Andre Scedrov and Vitaly Shmatikov for helpful discussions.

References

1. Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *POPL 2001*, pages 104–115, 2001.
2. Martín Abadi and Andrew D. Gordon. A bisimulation method for cryptographic protocol. In *Proc. ESOP 98*, Lecture notes in Computer Science. Springer, 1998.
3. Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 143:1–70, 1999. Expanded version available as SRC Research Report 149 (January 1998).
4. M. Backes, B. Pfitzmann, and M. Waidner. A General Composition Theorem for Secure Reactive Systems. In *TCC 2004*, volume 2951 of *LNCS*, pages 336–354. Springer, 2004.
5. M. Backes, B. Pfitzmann, and M. Waidner. Secure asynchronous reactive systems. Technical Report 082, Eprint, 2004.
6. Michael Backes, Birgit Pfitzmann, Michael Steiner, and Michael Waidner. Polynomial fairness and liveness. In *CSFW-15 2002*, pages 160–174, 2002.

7. Michael Backes, Birgit Pfitzmann, and Michael Waidner. Reactively secure signature schemes. In *Proceedings of 6th Information Security Conference*, volume 2851 of *LNCS*, pages 84–95. Springer, 2003.
8. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS 2001*. IEEE, 2001. Full version available at `http://eprint.iacr.org/2000/067/`.
9. Ran Canetti. Personal communication, 2004.
10. Ran Canetti and Marc Fischlin. Universally composable commitments. In *Proc. CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40, 2001. Springer.
11. Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 337–351. Springer, 2002.
12. Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 68–86. Springer, 2003.
13. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC 2002*, pages 494–503, 2002.
14. Anupam Datta, Ralf Küsters, John C. Mitchell, and Ajith Ramanathan. Sequential probabilisitic process calculus and simulation-based security. 2004. An extended version is available as a technical report at `http://www.ti.informatik.uni-kiel.de/~kuesters/publications_html/DattaKuestersMitchellRamanathan-TR-SPPC-2004.ps.gz`.
15. Anupam Datta, Ralf Küsters, John C. Mitchell, Ajith Ramanathan, and Vitaly Shmatikov. Unifying equivalence-based definitions of protocol security. In *ACM SIGPLAN and IFIP WG 1.7, 4th Workshop on Issues in the Theory of Security*, 2004.
16. C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
17. Patrick D. Lincoln, John C. Mitchell, Mark Mitchell, and Andre Scedrov. Probabilistic polynomial-time equivalence and security protocols. In *Formal Methods World Congress, vol. I*, number 1708 in LNCS, pages 776–793, 1999. Springer.
18. Robin Milner. *A Calculus of Communicating Systems*. Springer, 1980.
19. Robin Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.
20. John C. Mitchell, Mark Mitchell, and Andre Scedrov. A linguistic characterization of bounded oracle computation and probabilistic polynomial time. In *FOCS 1998*, pages 725–733, 1998. IEEE.
21. John C. Mitchell, Ajith Ramanathan, Andre Scedrov, and Vanessa Teague. A probabilistic polynomial-time calculus for the analysis of cryptographic protocols (preliminary report). In *17th Annual Conference on the Mathematical Foundations of Programming Semantics, 2001*, volume 45. ENTCS, 2001.
22. B. Pfitzmann and M. Waidner. A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In *IEEE Symposium on Security and Privacy*, pages 184–200. IEEE Computer Society Press, 2001.
23. Ajith Ramanathan, John C. Mitchell, Andre Scedrov, and Vanessa Teague. Probabilistic bisimulation and equivalence for security analysis of network protocols. Unpublished, see `http://www-cs-students.stanford.edu/~ajith/`, 2004.
24. Ajith Ramanathan, John C. Mitchell, Andre Scedrov, and Vanessa Teague. Probabilistic bisimulation and equivalence for security analysis of network protocols. In *FOSSACS 2004*, 2004. Summarizes results in [23].