Simulatable Security and Polynomially Bounded Concurrent Composability

Dennis Hofheinz
CWI, Cryptology and Information Security Group
Amsterdam, The Netherlands
Dennis.Hofheinz@cwi.nl

Dominique Unruh
IAKS, Arbeitsgruppe Systemsicherheit
Universität Karlsruhe, Germany
unruh@ira.uka.de

Abstract

Simulatable security is a security notion for multi-party protocols that implies strong composability features. The main definitional flavours of simulatable security are standard simulatability, universal simulatability, and black-box simulatability. All three come in "computational," "statistical" and "perfect" subflavours indicating the considered adversarial power. Universal and black-box simulatability, in all of their subflavours, are already known to guarantee that the concurrent composition even of a polynomial number of secure protocols stays secure.

We show that computational standard simulatability does not *allow for secure concurrent composition of polynomially many protocols, but we also show that statistical standard simulatability* does*. The first result assumes the existence of an interesting cryptographic tool (namely time-lock puzzles), and its proof employs a cryptographic multi-party computation in an interesting and unconventional way.*

Keywords: *Reactive Simulatability, Universal Composability, concurrent composition.*

1 Introduction

There are several ways to define what it means for a multi-party protocol to be secure. A very elegant and general way is the concept of simulatable security. With simulatable security, one first states what the protocol should do by specifying a single trusted host that completes the protocol task ideally and securely by construction. For instance, a trusted host for tossing a common coin for a set of parties would simply uniformly and randomly sample a bit b and then send b to each party. A simulatably secure protocol for coin toss must now be indistinguishable (in a well-defined sense) from this ideal setting. More specifically, no protocol environment must be able to detect differences between executions with the real protocol and executions with the trusted host in feasible time.

Thus, simulatable security actually establishes a security *relation* that considers a protocol secure *relative* to a suitable idealisation. However, when the idealisation for the considered protocol class is obvious, then a protocol is simply called secure, implicitly meaning security relative to that idealisation. Consequently, simulatable security captures the notion of a secure refinement of one system by another. In particular, it proved useful as a platform to show that a cryptographic implementation of a symbolic protocol is secure against cryptoanalytic attacks (see, e.g., [9, 5, 1, 20]). But simulatable security also helps to analyse the information-theoretic security guarantees of a one-time pad in a nice and convenient manner, cf. [38].

For defining and analysing a large protocol, a divide-and-conquer approach is generally helpful and sometimes even necessary. However, to allow for a modular protocol analysis, it is crucial that the composition of secure protocols stays secure. Secure composition of security properties should not be taken for granted: E.g., [33, 34] shows that several notions of non-interference are not preserved under composition. (This can be rectified, e.g., by deriving properties sufficient for non-interference-preserving composition [42] or adjusting the non-interference notion [31].) Similarly, most definitions of the cryptographic tool of Zero-Knowledge proof systems do not allow for securely composing even only two systems [25]. Since it is a difficult and laborious task to prove the different composability properties anew for each and every security property, it can be of great advantage to simply show that a protocol is simulatably secure. From this, many different security properties can be derived: e.g., preservation of integrity properties [36, 3], non-interference [4, 6], liveness properties [8], or key and message secrecy [7]. One can then make use of the composability guarantees simulatable security gives.

As just hinted, all flavours of simulatable security give certain composition guarantees. Namely, all flavours guarantee that a *constant* number of secure protocols can be composed in an arbitrary, concurrent manner without loss of security. Due to these composability guarantees, simulatable security could be used for defining and analysing protocol constructions for a very large class of protocol tasks in a modular way. Examples include a computationally sound analysis of the Needham-Schroeder-Lowe protocol [5], an electronic payment system [2], and a cryptographic construction for realizing a large class of protocol tasks [21].

However, in some scenarios, it might be desirable to compose more protocols at once. In fact, many commonly deployed cryptographic protocol constructions use a polynomial number of instances of primitives (i.e., subprotocols), e.g. [41, 26, 22]. The analysis of such constructions is generally reduced to analysing only one instance of each used primitive *type* at once. For deriving security of the whole construction, of course secure composability of a polynomial number of *instances* of each primitive type is needed.

So in this contribution, we investigate how simulatable security behaves under composition of a *polynomial* number of secure protocols. The flavours "universal simulatability" and "black-box simulatability" of simulatable security are already known to allow for this type of composition (see [15, 10]). However, whether this also holds for the other main flavour "standard simulatability", which is the default security notion in the Reactive Simulatability framework [37, 11], was explicitly posted as an open question in [10].

We show that computational standard simulatability (in which adversaries are computationally bounded) does not allow for secure composition of a polynomial number of protocols. We also show that statistical and perfect standard simulatability (which capture information-theoretic security and in which adversaries are unbounded) do allow for this type of composition, and we give a general composition theorem for that case. Below, we give a more detailed explanation.

Note that although this shows that the default notion of security in the Reactive Simulatability framework does not imply polynomially bounded concurrent composability, this has no impact on *existing* security proofs in that framework. These all show black-box simulatability, which is known to imply polynomially bounded concurrent composability.

Related Work/Technical Overview. The concept of simulatability has a long history (see, e.g., [40, 27, 26, 12, 35, 13, 36, 14, 37, 15, 11, 18]). In recent years, in particular the simulatability frameworks of Reactive Simulatability [37, 11] and Universal Composability [15, 18] proved useful for analysing security properties of protocols in distributed systems.

In both frameworks, a protocol $\hat{M}_1$ is considered *as secure as* another protocol $\hat{M}_2$ (usually an idealisation of the respective protocol task), if $\hat{M}_1$ is indistinguishable from $\hat{M}_2$ in every protocol context. This should hold also in the presence of attacks, i.e., we should have that every attack on $\hat{M}_1$ can be simulated by an attack on $\hat{M}_2$. (So every weakness of $\hat{M}_1$ must be already present in the ideal specification $\hat{M}_2$.)

A little more formally, this means that for every *adversary* A attacking $\hat{M}_1$, there is an adversary S (usually referred to as the *simulator*) that attacks $\hat{M}_2$, such that from an outside view, both attacks and protocols "look the same." For capturing what "looking the same" means, a designated entity called the *honest user* H is run with protocol $\hat{M}_1$ (together with adversary A) and protocol $\hat{M}_2$ (with simulator S). The honest user H represents a protocol context and may interact with protocol participants and even with the adversary. For security, every possible H must experience indistinguishable views with $\hat{M}_1$ and with $\hat{M}_2$.

One might now choose S in dependence of H; this leads to the *standard simulatability* definition, which is the default in the Reactive Simulatability framework. Alternatively, the user H may be allowed to depend on the respective simulator S; this is called *universal simulatability* and is the default security notion in the Universal Composability model.

Both simulatability variants allow for some form of secure composition of protocols. We can distinguish two important types of composition. First, *simple composability* guarantees that if a protocol $\hat{M}_1$ is as secure as another protocol $\hat{M}_2$, then a protocol $\hat{N}^{\hat{M}_1}$ that uses a single instance of $\hat{M}_1$ is as secure as the protocol $\hat{N}^{\hat{M}_2}$ which uses $\hat{M}_2$ instead. Further, we have *polynomially bounded concurrent composability* which guarantees for every polynomial p, that $\hat{M}_1^p$ is as secure as $\hat{M}_2^p$, where $\hat{M}_1^p$ and $\hat{M}_2^p$ denote the concurrent execution of p instances of $\hat{M}_1$ and $\hat{M}_2$, respectively. One can show that if simple *and* polynomially bounded concurrent composability hold, one can securely substitute a polynomial number of subprotocols at a time in a larger protocol.

It is known that standard simulatability implies simple composability, cf. [36, 37]. Also known is that universal and black-box simulatability additionally allow polynomially bounded concurrent composability, see [15, 10]. Furthermore, [32] investigated which further relationships between the notions of standard/universal simulatability and simple composability/polynomially bounded concurrent composability hold and found the interesting fact that simple composability and standard simulatability are equivalent. However, the following was given as open questions in [32]: Does standard simulatability imply polynomially bounded concurrent composability, and do simple and polynomially bounded concurrent composability together already imply universal simulatability? Or do even standard and universal simulatability coincide?

For a modified definition of standard simulatability, this question was answered in [17, 19]. In this definition, the runtime of the honest user H may depend on the length of its non-uniform input, which again may depend on the simulator. They showed that using this modification, standard, universal, and black-box simulatability all coincide. However, this modification of standard simulatability breaks the proof of [32] that standard simulatability and simple composability coincide. So even the modified definition of standard simulatability left open whether simple composability implies universal simulatability.

Further progress was then made by [29] who showed that computational standard simulatability (in the original formulation) does not imply computational universal simulatability. However, their separating counterexample is not only secure w.r.t. standard simulatability, but also composes concurrently even a polynomial number of times, so simple and polynomially bounded concurrent composability together do *not* already imply universal simulatability. Also, [29] show that their result depends on the computational model: while they give separating examples in case of computational and statistical security, they show that in case of perfect security, standard/universal simulatability (and thus also simple/polynomially bounded concurrent composability) coincide. However, the open question of [32] whether standard simulatability is sufficient for polynomially bounded concurrent composability was still left unanswered.

A note concerning the nomenclature: Universal simulatability is also often called UC security [15], standard simulatability is called specialised-simulator UC in [32], the honest user is also known as the environment [15], simple composability as 1-bounded general composability [32], and simple and polynomially bounded concurrent composability together are also called polynomially-bounded general composability [32].

Our Work. In this work, we answer the remaining open questions and provide the missing implications and separations among standard/universal simulatability and the different notions of composability. More specifically, we show that computational standard simulatability does *not* imply polynomially bounded concurrent composability. Further, we show that in contrast, statistical standard simulatability *does* imply polynomially bounded concurrent composability. An overview over the implications and separations is given in Figure 1.

Our results hold both in the Reactive Simulatability and the Universal Composability framework (as in [15]). The main difference between these security notions is that Reactive Simulatability considers uniform machines, while with Universal Composability, the honest user has access to a non-uniform input that is chosen after honest user and simulator. We prove the results using the Reactive Simulatability formalism, but additionally cover the case that the honest user gets such a non-uniform input, so that it is easy to reformulate the proof using Universal Composability.

Finally, we discuss the impact of recent developments in simulatability-based security definitions on our work. Namely, in [19] and in [28], (different) alternative definitions of polynomial-time adversarial entities were introduced. We point out why our separating example does not work with these definitions.

Organization. After recalling the Reactive Security framework in Section 2, we show in Section 3 that computational standard simulatability does not imply polynomially bounded concurrent composability. In Section 4, we prove polynomially bounded concurrent composability for the statistical and perfect case. Section 5 concludes this work.

2 Reactive Simulatability

Here we review the notion of Reactive Simulatability (RS). This introduction only sketches the definitions, and the reader is encouraged to read [11] for more detailed information and formal definitions.

Reactive simulatability (in the "standard" flavour) is a definition of security which defines a protocol $\hat{M}_1$ (the *real protocol*) to be *as secure as* another protocol $\hat{M}_2$ (the *ideal protocol*, the *trusted host*, the *ideal functionality*), iff the following holds: for any adversary A (also called the *real adversary*), and any *honest user* H (that represents a possible protocol environment), there is an adversary S (also called *simulator* or *ideal adversary*), s.t. the view of H is indistinguishable in the following two scenarios:

- The honest user H runs together with the real adversary A and the real protocol $\hat{M}_1$
- The honest user H runs together with the simulator S and the ideal protocol $\hat{M}_2$.

Note that there is a security parameter k common to all machines, so that the notion of indistinguishability makes sense.

This definition allows to specify some trusted host—which *by definition* is a *secure* formalisation of some cryptographic task—as the ideal protocol, and then to consider the question whether a real protocol is as secure as the trusted host (and thus also a secure implementation of that task).

In order to understand the above definitions in more detail, we have to specify what is meant by machines "running together". Consider a set of machines that may send messages through *connections* to each other. Whenever a ma-

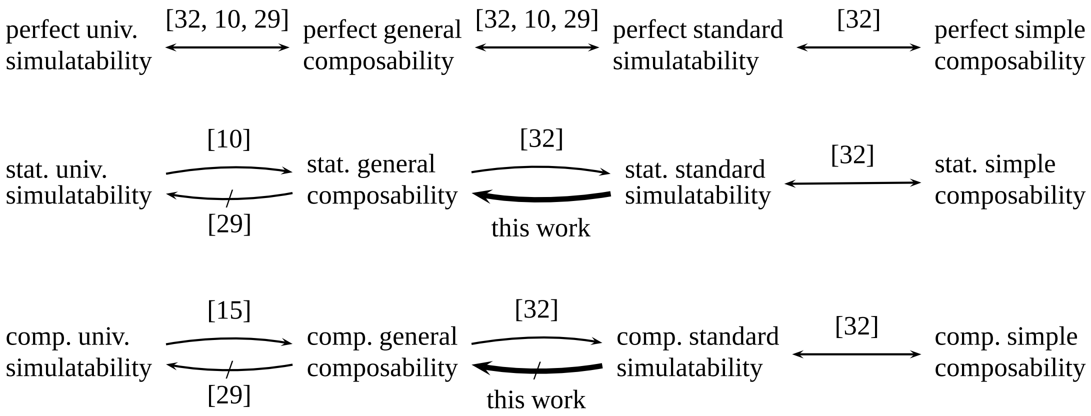
perfect univ. simulatability
[32, 10, 29]
perfect general composability
[32, 10, 29]
perfect standard simulatability
[32]
perfect simple composability
stat. univ. simulatability
[10]
[29]
stat. general composability
[32]
this work
stat. standard simulatability
[32]
stat. simple composability
comp. univ. simulatability
[15]
[29]
comp. general composability
[32]
this work
comp. standard simulatability
[32]
comp. simple composability

Figure 1. Implications and separations between the various security notions. For the presentation of these relations, we adopt the taxonomy of [32] who additionally has the notion of (polynomially bounded) general composability, which means that *both* simple composability and polynomially bounded concurrent composability hold. The references next to the arrows indicate where this was proven. The results from this paper are given by bold arrows.

chine sends a message, the receiving machine is activated with that message.[2]

At the start of a *run* of these machines, a designated machine called the *master scheduler* is activated. This machine is always the honest user or the adversary. Afterwards, the next machine to be activated is determined by the message sent by the current machine as described above. If the current machine decides not to send a message, the master scheduler is activated again. The transcript of all communication and all internal states of the machines in such a run gives us a random variable which we call simply the *run*. By restricting the run to the internal states of and the communication sent or received by a machine H, we get the *view* of the machine H. We write $view_{\hat{M},k}(\mathsf{H})$ for the view of H given security parameter k. The index k can be omitted, then we mean the family consisting of all the random variables $view_{\hat{M},k}(\mathsf{H})$.

A protocol is simply a set of machines (e.g., protocol parties, trusted hosts) together with a specification over which connections an honest user can talk to the protocol. The latter is important, because there usually are connections that reflect the internal communication of the protocol which should not be accessible directly by the honest user. A protocol cannot run by itself, it can only run together with an honest user and an adversary.

Given the above definitions, we can now state the definition of security more formally: Let $\hat{M}_1$ and $\hat{M}_2$ be protocols. We say that $\hat{M}_1$ is *as secure as* $\hat{M}_2$ if for any adversary A and any honest user H there is a simulator S s.t. $view_{\mathsf{H,A},\hat{M}_1}(\mathsf{H})$ and $view_{\mathsf{H,S},\hat{M}_2}(\mathsf{H})$ are indistinguishable.

The meaning of “indistinguishable” depends on the exact notion of security. For *perfect* security, the views must be identically distributed. For *statistical* security, their statistical distance must be negligible (in the security parameter k). For *computational* security, they must be computationally indistinguishable by polynomial-time algorithms; in that case, also only polynomial-time users and adversaries/simulators are considered.

A further interesting point is the order of quantifiers. In the above definition we have allowed the simulator S to depend on the honest user H. We call this the standard order of quantifiers (because it is the default order in the RS framework) and speak of *standard simulatability*. Another possibility is to choose H after S, i.e., H may depend on S. Since in this case the simulator S has to be universal for all honest users H we speak of *universal simulatability*. Yet another possibility *black-box simulatability*, which demands the existence of an S that is even independent of A (but may use A as a black box).

Composition. A major advantage of a security definition by simulatability is the possibility of *composition*. There are two major flavours of composability, namely *simple com-*

[2] In the model of [11] there is additionally the concept of so-called clock-ports. These allow to model asynchronous communication. We have opted to omit these clock-ports here and to assume that all messages are delivered immediately (or sent to the adversary in case of an insecure connection). This greatly simplifies the presentation and does not principally restrict the expressibility of the model, since asynchronous communication can also be modelled by introducing functionalities for communication which deliver messages only upon request by the adversary. However, all our results can also be stated in the more general setting of [11].

posability and *polynomially bounded concurrent composability* . To sketch simple composability, let $\hat{N}^{\hat{M}_1}$ be an arbitrary large protocol that uses (*one* instance of) another protocol $\hat{M}_1$ as subprotocol. Simple composability means that in any such $\hat{N}^{\hat{M}_1}$, any secure realisation $\hat{M}_2$ of $\hat{M}_1$ can substitute $\hat{M}_1$ *without losing security*. More precisely, if $\hat{M}_2$ is as secure as $\hat{M}_1$, then $\hat{N}^{\hat{M}_2}$ (in which $\hat{M}_1$ has been replaced by $\hat{M}_2$) is as secure as $\hat{N}^{\hat{M}_1}$. Both standard and universal have this property of *simple composability*. Simple composition could be used, e.g., to modularise the proof of protocols for secure message transmission using public-key [37, 15] or secret-key encryption schemes [38].

A natural extension is to consider substituting *multiple* instances of one subprotocol at once. In other words, one can ask for the same property as above even if $\hat{N}^{\hat{M}_1}$ uses several instances of $\hat{M}_1$. This stronger notion has been used, e.g., to modularise the security proof of the general protocol construction [21] for secure function evaluation. Given that simple composition holds, this concept can be reduced to what is known as *polynomially bounded concurrent composability* : roughly, this means that $\hat{M}_2^p$ (i.e., p copies of $\hat{M}_2$ run concurrently) is as secure as $\hat{M}_1^p$ whenever $\hat{M}_2$ is as secure as $\hat{M}_1$. (Commonly, the number p of allowed instances is restricted to be polynomial in the security parameter, since this is usually sufficient for many applications—in particular, for the important class of polynomial-time protocols—and, in particular for statistical security, often the best one can hope for.)

As sketched in the introduction, it is known that universal simulatability already has the feature of polynomially bounded concurrent composability (cf. [15, 10]). In this contribution, we are interested whether this also holds for standard simulatability. Thus, to express polynomially bounded concurrent composability formally, we need a definition for the "concurrent composition" $\hat{M}^p$ of a protocol $\hat{M}$.

Intuitively, when $\hat{M}$ is a protocol and $p = p(k)$ a polynomial in the security parameter, then $\hat{M}^p$ is the protocol where each machine has been replaced by p copies of the original machine. To avoid complicated definitions, instead of p copies we will introduce a single machine that simulates p copies which are accessed by a session ID that precedes each message.[3]

Definition 2.1 (Polynomially Bounded Concurrent Composability)**.** *Let* M *be a machine and* $p = p(k)$ *be a polynomial in the security parameter. Then* M^p *simulates* p *copies* $\mathsf{M}_1, \dots, \mathsf{M}_p$ *of* M. *Upon receiving a message* (sid, m) *with* $1 \leq sid \leq p$, M^p *hands* m *to* M_{sid}. *When a simulated* M_{sid} *sends a message* m, *then* M^p *sends* (sid, m).

For a protocol $\hat{M}$, *the protocol* $\hat{M}^p$ *consists of all machines* M^p *with* $\mathsf{M} \in \hat{M}$.

Given this definition, we can now formulate polynomially bounded concurrent composability: say that $\hat{M}_1$ is as secure as $\hat{M}_2$. Then $\hat{M}_1^p$ should be as secure as $\hat{M}_2^p$ for any given polynomial p in the security parameter.

3 The Computational Case

Consider the case of computational standard simulatability. We give protocols $\hat{M}_1$ and $\hat{M}_2$ such that $\hat{M}_1$ is as secure as $\hat{M}_2$, but the k-fold concurrent composition $\hat{M}_1^k$ is not as secure as $\hat{M}_2^k$. (As usual, k denotes the security parameter.)

3.1 Time-lock puzzles

As a tool for our construction, we need means for a machine to prove its computational strength. Such a tool is provided by time-lock puzzles [39, 29]. Intuitively, solving a time-lock puzzle t of hardness $s \in \mathbb{N}$ is a strong indication that a machine has done computational work polynomial in s.

More precisely, a time-lock puzzle consists of a puzzle generation algorithm $\mathcal{G}$ and a solution verification algorithm $\mathcal{V}$. The solution generation algorithm $\mathcal{G}$ that takes some number s (the *hardness* of the puzzle) as input and then outputs a puzzle q and some auxiliary information a for the solution verification algorithm. The solution verification algorithm $\mathcal{V}$ takes an supposed solution t and decides (possibly using the auxiliary information a) whether the solution is correct.

To ensure that s indeed represents the hardness of the puzzle q, we require the following two properties for $\mathcal{G}$ and $\mathcal{V}$ to represent a system for time-lock puzzles:

- *Hardness condition:* For any algorithm B that is polynomial in the security parameter k, there is a polynomial p, s.t. the algorithm B never solves puzzles of hardness $\geq p(k)$. More concretely, when choosing $s \geq p(k)$ and generating a puzzle q of hardness s, the probability is negligible that B upon input q outputs a solution t that is accepted by $\mathcal{V}$.

- *Easiness condition:* For any polynomial p there is an algorithm C polynomial in the security parameter k, s.t. the algorithm solves all puzzles of hardness $\leq p(k)$. More concretely, when choosing $s \leq p(k)$ and generating a puzzle q of hardness s, the probability is overwhelming that B upon input q outputs a solution t that is accepted by $\mathcal{V}$. (Note that in both cases, B and C do not have access to the auxiliary information a.)

A formal definition and more details can be found in [29] or in the full version of this paper.

[3] A more general methodology can be found in [10], where parametrised families of protocols are used to formulate a variable number of machines. The results given here can also be stated in their formalism.

Time-lock puzzles will allow us to perform "contests of computational strength" between polynomial-time machines, because whichever machine can solve the larger time-lock puzzles is the more powerful. This idea of a "contest" has already been used in [29] to separate universal and standard simulatability in the computational case, and will also be useful to construct our counterexample.

3.2 The General Idea

The idea behind our example is as follows. Both protocols $\hat{M}_1$ and $\hat{M}_2$ consist only of one machine M_1 (resp. M_2) that expects to take part in a k-party secure function evaluation (SFE) of a specific function f. Here, k is the security parameter, so the number of parties actually increases for larger security parameters. Such a secure k-party function evaluation is possible under reasonable computational assumptions (namely, the existence of enhanced trapdoor permutations) using the construction of [26, 23, 24]. Since M_1 executes the program of only *one* party of the SFE, all internal messages of the SFE are sent to and expected from *the honest user* H.

The machine M_1 differs from M_2 only in its way of choosing the inputs to the function evaluation. More specifically, M_1 chooses all of its inputs on its own, whereas M_2 chooses only some inputs on its own (in a different way than M_1) and lets the simulator S decide upon the remaining inputs. The specific choice of f ensures that a simulator S that is fixed after the protocol user H is able to deliver inputs to M_2 such that the function output of f is the same in real and ideal model. Using the secrecy property of the function evaluation construction, this means that $\hat{M}_1$ and $\hat{M}_2$ are indistinguishable from the point of view of H, even though H sees the internal messages of the SFE.

However, when considering $\hat{M}_1^k$ and $\hat{M}_2^k$, a suitable protocol user H can simply "intermediate" between the function evaluation parties (i.e., the k copies of M_1, resp. M_2). Thus, in the real model, H forces a secure function evaluation with k copies of M_1, and in the ideal model, it forces a secure function evaluation with k copies of M_2. Because there are now k different function evaluation parties that give all different inputs in the real, resp. the ideal model, the choice of f guarantees that now the simulator S is unable to enforce indistinguishable function outputs in the real, resp. ideal model.

3.3 The Evaluated Function

Of course, the choice of the function f is crucial, so we will begin by presenting f. The function f proceeds in two rounds. In the first round, f expects input (b_i, s_i) with $b_i \in \{\mathtt{real}, \mathtt{ideal}\}$, $s_i \in \mathbb{N}$ from each party $i = 1, \ldots, k$ (we will call these the *first inputs*). Then time-lock puzzles q_i of hardness s_i are (independently) chosen, and the output to party i is q_i (we call these the *first outputs*). The information for checking the solution is stored. In the second round, f expects a solution t_i to the puzzle q_i from each party i. The final output out of f (which is the same for all parties) is then calculated as follows:

1. Sort all s_i with $b_i = \mathtt{ideal}$ in order of ascending s_i into a list $s_{i_1}, s_{i_2}, \ldots, s_{i_n}$ such that $s_{i_j} \le s_{i_{j+1}}$ for all j.
2. Let $out := \mathtt{true}$ if the predicate

   $$\forall j = 1, \ldots, n : s_{i_j} \ge 2^j \text{ and } t_{i_j} \text{ is a correct solution for } q_{i_j}$$

   holds, and let $out := \mathtt{false}$ otherwise.

Obviously, only the set of values (s_i, t_i) with $b_i = \mathtt{ideal}$ is relevant for the output of f. In particular, $out = \mathtt{true}$ implies that a time-lock puzzle has been solved that has a hardness that is exponential in the number of inputs with $b_i = \mathtt{ideal}$. Or, put differently, no polynomial machine can give inputs such that $b_i = \mathtt{ideal}$ for all i and hope to achieve an evaluation to $out = \mathtt{true}$ with non-negligible probability.

3.4 The Protocols

Using the construction of [26, 23, 24], denote by $P_1, \ldots P_k$ parties that securely evaluate f in a k-party function evaluation. That is, each P_i takes as local first input a tuple (b_i, s_i) as above and eventually—after having communicated with $k - 1$ other parties—outputs a time-lock puzzle q_i as specified by f. Then P_i expects a second input t_i and finally—after further communication—outputs out as prescribed by f.

Using the programs of these parties, we define the protocol machines M_1 and M_2 which make up the protocols $\hat{M}_1$, resp. $\hat{M}_2$.[4] Namely, let M_1's program be as follows:

1. Ask the protocol user H for a party index $i \in \{1, \ldots, k\}$.
2. Run the program P_i internally, where
   - P_i's first inputs are set to $b_i := \mathtt{real}$ and $s_i := 0$, and the second input is $t_i := \varepsilon$ (where ε denotes the empty word). The first output of P_i is simply ignored.
   - All outgoing messages are sent to H (prefixed with the recipient party index or indicated as a broadcast).

[4] In our example, each protocol consists of only one machine.

- Messages coming from H that are prefixed with a party index $j \neq i$ are forwarded to the internal P_i as if coming from P_j.

3. As soon as P_i generates its final output out, forward this output to H and halt.

In other words, M_1 asks H for a party index i and then expects to take part in an evaluation of f in the role of P_i. Here, P_i's local inputs are fixed to $b_i := \texttt{real}$, $s_i := 0$, and $t_i := \varepsilon$, and all network communication is relayed over H. The evaluated function output is eventually forwarded to H.

As mentioned earlier, the protocol $\hat{M}_1$ then consists only of this single machine M_1. On the other hand, protocol $\hat{M}_2$ consists of only one machine M_2 that is defined—very similarly—as follows:

1. Ask the protocol user H for a party index $i \in \{1, \dots, k\}$.

2. Run the program P_i internally, where

   - P_i's first inputs are set to $b_i := \texttt{ideal}$, and the simulator is asked for the value of s_i. When the first output q_i has been generated, it is sent to the simulator, and a second input t_i is expected.
   - All outgoing messages are sent to H (prefixed with the recipient party index or indicated as a broadcast).
   - Messages coming from H that are prefixed with a party index $j \neq i$ are forwarded to the internal P_i as if coming from P_j.

3. As soon as P_i generates its final output out, forward this output to H and halt.

The only difference between M_1 and M_2 lies in the way the local inputs to P_i are determined: M_1 fixes these inputs as above, and M_2 only sets $b_i := \texttt{ideal}$ and lets the simulator determine the inputs s_i and t_i.

3.5 Security of the Single Protocol

We show that $\hat{M}_1$ is as secure as $\hat{M}_2$ (with respect to computational standard simulatability. For this, we may assume a given protocol user H and adversary A and need to construct a simulator S such that H cannot distinguish running with $\hat{M}_1$ and A from running with $\hat{M}_2$ and S. Intuitively, H can distinguish only if the respective function evaluation outputs in $\hat{M}_1$ and $\hat{M}_2$ differ. So S must only ensure that the function outputs in $\hat{M}_2$ are as they would have been in $\hat{M}_1$ (where the inputs of M_1 are different from those of the ideal-model machine M_2).

More specifically, S runs A as a black box, so that communication between A and H is the same in the real and in the ideal model. The only thing that S needs to do on its own is to answer M_2's question for the strength s_i and the solution t_i. When asked for these inputs, S chooses and solves a puzzle of hardness s_i more than twice as large as the largest hardness H could solve.[5] (The time-lock puzzle definition guarantees that such an S exists for fixed H.) The situation is depicted in Figure 2.

This way, S solves a puzzle of such large hardness s_i that when evaluating f, this puzzle appears in the last position in the sorted list $(s_{i_1}, s_{i_2}, \dots, s_{i_n})$ (cf. the definition of f in Section 3.3) and is at least twice as hard as the preceding puzzle $s_{i_{n-1}}$ (or there is an invalid solution t_{i_j} with overwhelming probability). Thus, if $s_{i_n} = s_i < 2^n$, then already $s_{i_{n-1}} < 2^{n-1}$. So intuitively, it is never the "fault" of M_2 when f evaluates to $\texttt{false}$; the same would have happened in the real model with a machine M_1. Conversely, if already one of the s_{i_j} $(j < n)$ is smaller than 2^j or does not have a valid solution, then f will return $\texttt{false}$ independently of s_{i_n}. So it is never the "fault" of M_2 when f evaluates to $\texttt{true}$, either.

In other words, the output of the SFE of f has the same distribution, regardless of whether H runs with $\hat{M}_1$ and A, or with $\hat{M}_2$ and S. Due to the secrecy of the SFE, this implies that the internal messages of the SFE and therefore the views of H are also indistinguishable in these two scenarios.

So we get the following lemma:

Lemma 3.1. *Assume enhanced trapdoor permutations and systems for time-lock puzzles exist. Then protocol $\hat{M}_1$ from above is as secure as protocol $\hat{M}_2$ from above with respect to computational standard simulatability.*

This also holds when the honest user has access to an auxiliary input (that may even be chosen after the simulator).

The complete proof will be given in the full version of this paper.

3.6 Insecurity under k-fold Concurrent Composition

Lemma 3.2. *Assume that systems for time-lock puzzles exist. Then for the protocols $\hat{M}_1$ and $\hat{M}_2$ from above, we have that $\hat{M}_1^k$ is not as secure as $\hat{M}_2^k$ with respect to computational standard simulatability.*

This does not depend on whether the honest user has auxiliary input or not.

Proof. We show that $\hat{M}_1^k$ is not as secure as $\hat{M}_2^k$. For this, we give a special adversary A and protocol user H such that no simulator S can mimic A in the ideal model.

[5] In the formal proof, we need a larger, yet still polynomial bound for technical reasons.

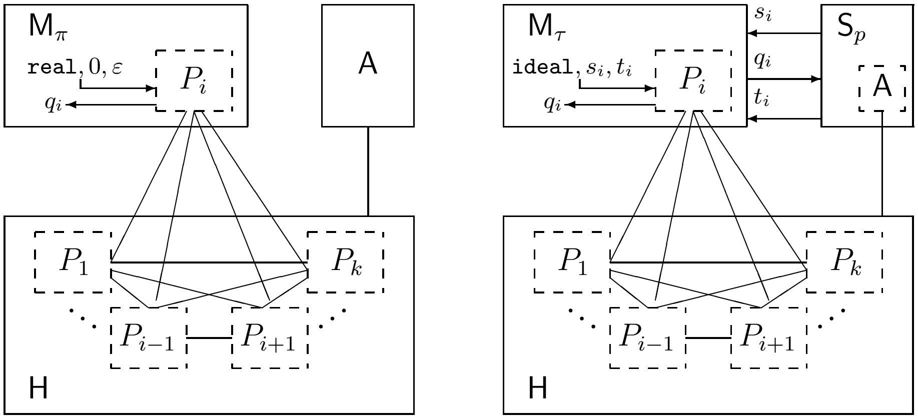

Figure 2. Left-hand side: a single execution of protocol $\hat{M}_1$ with adversary A and user H; right-hand side: a single execution of protocol $\hat{M}_2$ with simulator S and user H. The simulated parties $P_1, \dots, P_k$ perform a secure function evaluation protocol both in $\hat{M}_1$ and in $\hat{M}_2$.

Let A be a machine that does nothing at all (note that since $\hat{M}_1$ is a one-party-protocol, the adversary does not need to deliver any messages). Let H be such that, when running with k protocol machines (either k copies of M_1 or k copies of M_2), it behaves as follows:

1. FOR $i := 1$ TO k: Tell the i-th protocol machine (i.e., the i-th copy of either M_1 or M_2) to take the role of P_i. END FOR

2. Whenever the i-th protocol machine wants to send a message to the j-th protocol machine, relay this message. (When the i-th protocol machine wants to broadcast a message, deliver that message to all protocol machines.)

3. As soon as the first protocol machine generates output, halt.

By definition of f, in the real model, running with A and k copies of M_1, this honest user H will experience a function evaluation output $out = \texttt{true}$ (i.e., at least one copy of M_1 will output $\texttt{true}$ to the honest user H). Thus, a successful simulator S has to achieve a function evaluation output $out = \texttt{true}$ as well with overwhelming probability. By definition of f and the ideal machines M_2, this means that it has to supply valid solutions t_i to puzzles of hardness s_i where at least one satisfies $s_i \geq 2^k$ (since all $b_i = \texttt{ideal}$). However, this directly contradicts the hardness requirement in the time-lock puzzle definition, since S has to be polynomial-time. Therefore no such simulator exists and H can always distinguish $\hat{M}_1$ and $\hat{M}_2$. □

Combining Lemmas 3.1 and 3.2, we can summarize:

Theorem 3.3. *Assume that enhanced trapdoor permutations and systems for time-lock puzzles exist. Then computational standard simulatability does not guarantee polynomially bounded concurrent composability. That is, there are protocols $\hat{M}_1$ and $\hat{M}_2$, such that with respect to computational standard simulatability, $\hat{M}_1$ is as secure as $\hat{M}_2$, but the composed protocol $\hat{M}_1^k$ is not as secure as $\hat{M}_2^k$.*

This holds regardless of whether the honest user has access to an auxiliary input or not.

4 The Statistical Case

In contrast to the case of computational security, we will show that for statistical (i.e., information theoretical) security a concurrent composition theorem indeed holds.

First some investigation of the actual definition of statistical security is necessary. The definition of statistical security for the RS framework in [11] requires the following: *Polynomial prefixes* of the views of the honest user in the ideal and real model shall be statistically indistinguishable. However, in [30] it was shown that this notion is problematic. It was shown that due to the restriction to polynomial prefixes of views not even the simple composability holds, even in the case of universal security. Further it was shown in [30] that the natural correction of the problem, namely removing the restriction to polynomial prefixes, fixes the (simple) composition theorem.

Therefore, we will adapt the following definition of statistical standard security:

Definition 4.1 (Strict statistical security (as in [30], slightly simplified))**.** *Let $\hat{M}_\pi$ and $\hat{M}_\tau$ be protocols. We say that $\hat{M}_\pi$* is as secure as *$\hat{M}_\tau$* with respect to standard statistical security *iff the following holds:*

For every honest user H *and real adversary* A*, there is a simulator* S*, s.t. the statistical distance between the following families of views is negligible in* k*:*

$$\{view_{\mathsf{H},\mathsf{A},\hat{M}_\pi,k}(\mathsf{H})\}_k, \qquad \{view_{\mathsf{H},\mathsf{S},\hat{M}_\tau,k}(\mathsf{H})\}_k$$

(Here $view_X(\mathsf{H})$ *denotes the view of* H *in a run of* X*.)*

When the simulator S *does not depend on the adversary* A*, we speak of* statistical universal security.

When referring to Def. 4.1 we will simply speak of *statistical security* for brevity.

4.1 Proving Polynomially Bounded Concurrent Composability

Here, we first review the idea of how to show concurrent composability in the case of universal security, and argue why the proof idea doesn't apply to standard security.

When investigating proofs of concurrent composability (in the case of universal security, for more details see e.g., [15, 10]), we see that the main proof idea is approximately the following: Consider as real adversary only a dummy adversary, i.e., an adversary that simply follows all instructions received from the honest user.[6] To prove $\hat{M}_1^p$ as secure as $\hat{M}_2^p$ assuming $\hat{M}_1$ is as secure as $\hat{M}_2$, let a simulator S for that dummy adversary attacking the single protocol be given. Note that since we assume universal security, S does not depend on the honest user.

It might be reasonable to expect that a "parallelised version" S^p of the simulator S (so that S^p internally keeps p simulations of S, one for each protocol instance) is a good simulator for the dummy adversary that attacks the composed protocol M_1^p. To support this intuition, we reduce honest users of the composed protocol to honest users of the single protocol. (Note that since we can restrict to the dummy adversary as real adversary, this is all we need for showing our claim.)

Namely, for each honest user H^* of the composed protocol $\hat{M}_1^p$, we construct a new honest user H_p of a single copy $\hat{M}_1$ as follows (cf. also Figure 3): H_p simulates H^*. For each copy of the protocol that H^* expects, H_p does one of the following: (i) the real protocol and real (dummy) adversary are simulated (we will call this a "real copy"), (ii) the ideal protocol and simulator S are simulated (we call this an "ideal copy"), or (iii) communication from H^* is rerouted to the outside of H_p, where either one copy of the real or of the ideal protocol resides (we speak of an "external copy"). The number of "real" and "ideal copies" is chosen randomly (and there will be exactly one "external copy"). H_p choosing to simulate l "real copies" and running with the real protocol is equivalent to H_p choosing to simulate $l+1$ "real copies" and running with the ideal protocol and simulator. This again is indistinguishable (by assumption of the security of a single protocol copy) from H_p choosing to simulate $l+1$ copies and running with the real protocol. So we get a chain of polynomial length of indistinguishable views[7] from H_p choosing to simulate 0 "real copies" to H_p choosing to simulate p "real copies", so these two settings are again indistinguishable (by the simulated H^*). These two scenarios again correspond to H^* running with only ideal copies of the protocol (and copies of the 'dummy adversary) and H^* running with only real copies (and copies of S for each protocol-copy), so H^* can indeed not distinguish between real and ideal model.

But when we consider standard security, the following problem occurs: We have relied on the fact that the simulator S is a "good" simulator for H_p. But for standard security, such a "good" S would depend on H_p, which in turn depends on S. It is not clear that this mutual dependency should have a fixpoint (and in fact, it does not have such a fixpoint in the counterexample presented in Section 3 for the computational case).

While it is unknown whether such a fixpoint exists in the case of statistical standard security, a variation of the above construction yields a proof. We first state the theorem:

Theorem 4.2 (Polynomially Bounded Concurrent Composition Theorem)**.** *Let* $\hat{M}_1$ *and* $\hat{M}_2$ *be protocols s.t.* $\hat{M}_1$ *is as secure as* $\hat{M}_2$ *(with respect to standard statistical security as in Def. 4.1). Let further* p *be a polynomial.*

Then $\hat{M}_1^p$ *is as secure as* $\hat{M}_2^p$ *(where* $\hat{M}_i^p$ *denotes the polynomially bounded concurrent composability as in Def. 2.1).*

This also holds when the honest user has access to an auxiliary input.

Note that the limitation to a polynomial number is not a limitation of our proof, indeed, it can easily be seen that the concurrent composition of a superpolynomial number of protocol instances can be insecure, even if the single instance is secure. This condition is usually not explicitly stated in the computational case: Since with polynomial-time machines only a polynomial number of protocol instances can be created, the condition is automatically fulfilled.

We will now give a proof sketch for Theorem 4.2. The full proof will be presented in the full version of this paper.

Proof sketch. Like in the approach sketched above, given an honest user H^* for the composed protocol $\hat{M}_1^p$, we construct honest users H_i for the single protocol $\hat{M}_1$. These choose a random number l and then simulate $l-1$ "ideal copies" with session IDs $1, \dots, l-1$, have one "external

[6] Maybe somewhat surprisingly, this dummy adversary is the "worst possible adversary" in the sense that it suffices to give a simulator for the real dummy adversary to show security, cf. [15].

[7] With a *common* negligible bound on the statistical distance.

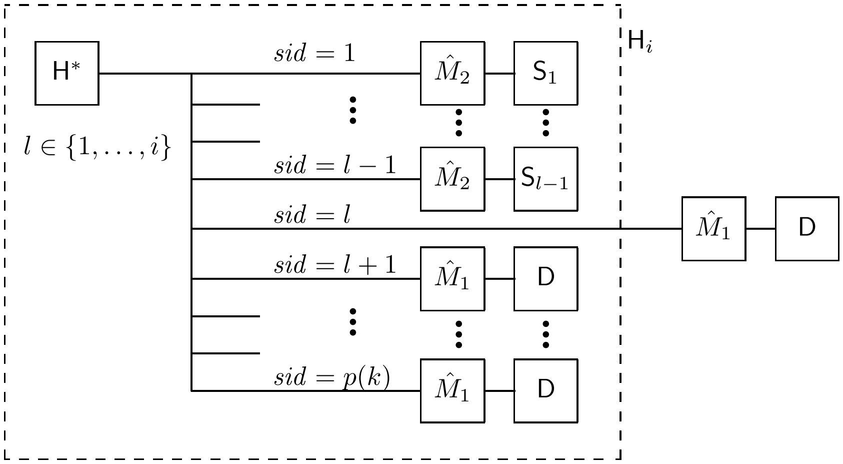

Figure 3. Construction of the honest user H_i (the dashed box). The variable l is drawn from the set $\{1,\ldots,i\}$. Messages from and to H^* are rerouted according to their session ID sid as depicted. (The fact that the adversaries/simulators are only connected to M_i is only for graphical reasons, in reality, they are of course connected to H^* as well.) The machines shown outside H_i are only exemplary, H_i might of course be connected to other machines, e.g. M_2 and S_i.

copy" with session ID l and simulate $p(k) - l - 1$ "real copies" with session IDs $l+1,\ldots,p(k)$ (cf. also Figure 3).

There are however some noteworthy differences to the construction of H_p in the approach above:

- Instead of having a single honest user H_p which chooses a random $l \in \{1,\ldots,p(k)\}$, H_i chooses $l \in \{1,\ldots,i-1\}$.
- The number l is chosen randomly with a fixed distribution s.t. any number l has a probability to be chosen whose inverse is polynomial in l. Then, if $l \geq i$, the honest user H_i aborts. Therefore, effectively a number $l \in \{1,\ldots,i\}$ is chosen, but in a way that any H_i with $i > l$ chooses l with the same probability. This gives us a kind of "compatibility" between the different honest users which will prove necessary to construct a common simulator for all these H_i.
- Most importantly, the "ideal copies" do not all contain the same simulator, since in the case of standard security there is no universal simulator to be used here. Instead, in the "ideal copy" with session ID sid has a simulator S_{sid}, where S_{sid} is defined in dependence of H_{sid} (see below). This of course seems to be a cyclic definition. However, closer inspection reveals that H_i only invokes "ideal copies" with session IDs $sid < i$, so H_i only depends on S_{sid} with $sid < i$. Therefore we have a mutually recursive definition of the H_i and the S_i.

The simulator S_i is defined to be a simulator for H_i. However, we require the simulator to be near-optimal in the following sense: For any security parameter k and any simulator S', the statistical distance among the real and ideal view of H_i when running with simulator S_i is by at most 2^{-k} larger than the statistical distance between those views when running with S'. The existence of such near-optimal simulators can easily be shown when unbounded simulators are allowed.

Further, we define H_∞ to be constructed like H_i with the exception that the number l is chosen without any limit. And, as before, S_∞ is a near-optimal simulator for H_∞.

Since we have constructed all the simulators to be "compatible" in the sense that any $l \leq i$ will be chosen by H_i with a probability not depending on i, we can argue as follows: When we ignore the protocol runs in which l is chosen as $l > i$, the view of the simulated H^* in H_i and H_∞ is the same (independent of further machines involved). S_∞ is a simulator for H_∞ that achieves that the statistical distance between H_∞'s real and ideal view is bounded by some negligible function, say ε. By ignoring runs with $l > i$, the distance of the views cannot increase. Therefore also the views of H_i have a distance of at most ε when running with S_∞. Since S_i was a near-optimal simulator, the statistical distance when running with S_i is bounded by $\varepsilon + 2^k$. Therefore we have a uniform bound for the distance of views for all pairs of honest user H_i and simulator S_i.

Now, if we modify H_i to always choose $l = i$ (and call the result $\tilde{\mathsf{H}}_i$), the statistical distance of views of this honest user (with simulator S_i) increases by a factor of at most the inverse probability that $l = i$ is chosen. Since this probability was polynomial in l (and independent of i), the statistical distance of the views of these modified $\tilde{\mathsf{H}}_i$ is bounded by a function $\varepsilon_i(k)$ negligible in i and k.

Finally, fix a security parameter k. By construction, $\tilde{\mathsf{H}}_{i+1}$ simulates i "ideal" and $p(k) - i - 1$ "real copies". So when running with $\hat{M}_1$ and D as the "external copy", this is equivalent to having $\tilde{\mathsf{H}}_i$ run with $\hat{M}_2$ and S_i. This again has only a statistical distance of $\varepsilon_i(k)$ (in the view of H^*) from the $\tilde{\mathsf{H}}_i$ running with $\hat{M}_1$ and D. So by repeatedly applying that equation, we see that between $\tilde{\mathsf{H}}_0$ and $\tilde{\mathsf{H}}_{p(k)+1}$ there is a distance of at most $\sum_{i=0}^{p(k)} \varepsilon_i(k) =: \nu(k)$, which is negligible in k. But $\tilde{\mathsf{H}}_0$ just simulates H^* together with $p(k)$ "real copies", which corresponds exactly to H^* running with the composed real protocol $\hat{M}_1^p$ (and the dummy adversary). Similarly, $\tilde{\mathsf{H}}_{p(k)+1}$ simulates H^* with $p(k)$ "ideal copies", corresponding to H^* running with the composed ideal protocol $\hat{M}_2^p$ and a simulator S resulting from combining all the individual simulators S_i. So the statistical distance between the views of H^* bounded by $\nu(k)$.

Since k was chosen arbitrarily, this holds for any k, i.e., the views of H^* in real and ideal composed protocol have a distance of at most ν which is negligible. Since the proof was done for arbitrary H^*, it follows that $\hat{M}_1^p$ is as secure as $\hat{M}_2^p$. □

4.2 The Perfect Case

The above proof can easily be modified to show concurrent composition in the case of perfect security (i.e., the views of the honest user must be identical and not only statistically close). However, there is a simpler argument using the results of [29]. They show that in the perfect case, standard and universal security coincide. Since for universal security, secure polynomially bounded concurrent composability is possible [15, 10], we immediately get

Theorem 4.3 (Polynomially Bounded Concurrent Composition Theorem, perfect case)**.** *The Polynomially Bounded Concurrent Composition Theorem 4.2 also holds in the case of perfect standard security.*

5 Conclusions

Composability properties of notions of simulatable security are of great importance when designing and analysing protocols modularly. Here, already some results are known, but the practically very significant question of polynomially bounded concurrent composability has not been answered in the case of standard simulatability. In this work, we have answered this open question for all flavours of standard simulatability. This clarifies all previously unknown relations among the different flavours of simulatability and compositional properties as depicted in Figure 1.

More specifically, we have shown that computational standard simulatability does not imply polynomially bounded concurrent composability. This does not only settle an open problem from [10]. It also has practical implications: many cryptographic protocol constructions in the spirit of [41, 26] make use of a polynomial number of subprotocols. Our results show that due to the lack of polynomially bounded concurrent composability, computational standard security is not well suited to analyse such constructions modularly. Hence, computational universal or black-box security should be preferred over computational standard security wherever possible, especially since all practical protocol constructions known to the authors are already proven secure with respect to these stronger notions.

On the other hand, we showed that in the statistical case, polynomially bounded concurrent composability is indeed guaranteed by standard simulatability. However, we still recommend the use of universal or black-box simulatability even in the statistical case, since the simulator constructed in our proof needs much more computational power than the simulator for the uncomposed protocol. In contrast to this, universal and black-box simulatability guarantee the existence of a simulator whose complexity is polynomial in the complexity of the real adversary.

Acknowledgements We are indebted to Michael Backes for many helpful comments and improvements. We also thank Ran Canetti and Jörn Müller-Quade for valuable discussions. This work was partially funded by the EC project ProSecCo under IST-2001-39227. Most of this work was done while the first author was with the Institut für Algorithmen und Kognitive Systeme, Arbeitsgruppe Systemsicherheit, Prof. Dr. Th. Beth, Universität Karlsruhe.

References

[1] Michael Backes. A cryptographically sound Dolev-Yao proof of the Otway-Rees protocol. In Pierangela Samarati, Peter Y.A. Ryan, Dieter Gollmann, and Refik Molva, editors, *Computer Security, Proceedings of ESORICS 2004*, number 3193 in Lecture Notes in Computer Science, pages 89–108. Springer-Verlag, 2004. Online available at `http://www.infsec.cs.uni-sb.de/~backes/papers/Back_04OtwayRees.ps`.

[2] Michael Backes and Markus Dürmuth. A cryptographically sound Dolev-Yao security proof of an electronic payment system. In *18th IEEE Computer Security Foundations Workshop, Proceedings of CSFW 2005*, pages 78–93. IEEE Computer Society, 2005. Extended version online available at `http://www.zurich.ibm.`

com/security/publications/2004/BaDu2004PaymentCL.pdf.

[3] Michael Backes and Christian Jacobi. Cryptographically sound and machine-assisted verification of security protocols. In Helmut Alt and Michel Habib, editors, *20th Annual Symposium on Theoretical Aspects of Computer Science, Proceedings of STACS 2003*, number 2607 in Lecture Notes in Computer Science, pages 675–686. Springer-Verlag, 2003.

[4] Michael Backes and Birgit Pfitzmann. Computational probabilistic non-interference. In Dieter Gollmann, Günter Karjoth, and Michael Waidner, editors, *Computer Security, Proceedings of ESORICS 2002*, number 2502 in Lecture Notes in Computer Science, pages 1–23. Springer-Verlag, 2002. Online available at http://www.infsec.cs.uni-sb.de/~backes/papers/BaPf_02ESORICS.ps.

[5] Michael Backes and Birgit Pfitzmann. A cryptographically sound security proof of the Needham-Schroeder-Lowe public-key protocol. In Paritosh K. Pandya and Jaikumar Radhakrishnan, editors, *Foundations of Software Technology and Theoretical Computer Science, Proceedings of FSTTCS 2003*, number 2914 in Lecture Notes in Computer Science, pages 1–12. Springer-Verlag, 2003. Extended version online available at http://eprint.iacr.org/2003/121.ps.

[6] Michael Backes and Birgit Pfitzmann. Intransitive non-interference for cryptographic purposes. In *IEEE Symposium on Security and Privacy, Proceedings of SSP 2003*, pages 140–152. IEEE Computer Society, 2003. Online available at http://www.zurich.ibm.com/~mbc/papers/BaPf_03Oakland.ps.

[7] Michael Backes and Birgit Pfitzmann. Relating symbolic and cryptographic secrecy. In *IEEE Symposium on Security and Privacy, Proceedings of SSP 2005*, pages 171–182. IEEE Computer Society, 2005. Extended version online available at http://eprint.iacr.org/2004/300.ps.

[8] Michael Backes, Birgit Pfitzmann, Michael Steiner, and Michael Waidner. Polynomial fairness and liveness. In *15th IEEE Computer Security Foundations Workshop, Proceedings of CSFW 2002*, pages 160–174. IEEE Computer Society, 2002. Online available at http://www.zurich.ibm.com/~mbc/papers/BPSW_02Liveness.ps.

[9] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A composable cryptographic library with nested operations. In *10th ACM Conference on Computer and Communications Security, Proceedings of CCS 2003*, pages 220–230. ACM Press, 2003. Extended abstract, extended version online available at http://eprint.iacr.org/2003/015.ps.

[10] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A general composition theorem for secure reactive systems. In Moni Naor, editor, *Theory of Cryptography, Proceedings of TCC 2004*, number 2951 in Lecture Notes in Computer Science, pages 336–354. Springer-Verlag, 2004. Online available at http://www.zurich.ibm.com/security/publications/2004/BaPfWa2004MoreGeneralComposition.pdf.

[11] Michael Backes, Birgit Pfitzmann, and Michael Waidner. Secure asynchronous reactive systems. IACR ePrint Archive, March 2004. Online available at http://eprint.iacr.org/2004/082.ps.

[12] Donald Beaver. Foundations of secure interactive computing. In Joan Feigenbaum, editor, *Advances in Cryptology, Proceedings of CRYPTO '91*, number 576 in Lecture Notes in Computer Science, pages 377–391. Springer-Verlag, 1992.

[13] Ran Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Weizmann Institute of Science, 1995. Online available at http://www.wisdom.weizmann.ac.il/~oded/PS/ran-phd.ps.

[14] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 3(1):143–202, 2000. Full version online available at http://eprint.iacr.org/1998/018.ps.

[15] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2001*, pages 136–145. IEEE Computer Society, 2001. Full version online available at http://www.eccc.uni-trier.de/eccc-reports/2001/TR01-016/revisn01.ps.

[16] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2001*, pages 136–145. IEEE Computer Society, 2001.

[17] Ran Canetti. Personal communication with one of the authors at TCC 2004, February 2004.

[18] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. IACR ePrint Archive, January 2005. Full and revised version of [16], online available at http://eprint.iacr.org/2000/067.ps.

[19] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. IACR ePrint Archive, January 2005. Online available at http://eprint.iacr.org/2000/067.ps.

[20] Ran Canetti and Jonathan Herzog. Universally composable symbolic analysis of cryptographic protocols. IACR ePrint Archive, September 2005. Online available at http://eprint.iacr.org/2004/334.ps.

[21] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing, Proceedings of STOC 2002*, pages 494–503. ACM Press, 2002. Extended abstract, full version online available at http://eprint.iacr.org/2002/140.ps.

[22] Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In Don Coppersmith, editor, *Advances in Cryptology, Proceedings of CRYPTO '95*, number 963 in Lecture Notes in Computer Science, pages 110–123. Springer-Verlag, 1995. Online available at http://www.cs.mcgill.ca/~crepeau/PS/CGT95.ps.

[23] Oded Goldreich. Secure multi-party computation. Unpublished, online available at http://www.wisdom.weizmann.ac.il/~oded/PS/prot.ps, October 2002.

[24] Oded Goldreich. *Foundations of Cryptography – Volume 2 (Basic Applications)*. Cambridge University Press, May 2004. Previous version online available at http://www.wisdom.weizmann.ac.il/~oded/frag.html.

[25] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. In Mike Paterson, editor, *Automata, Languages and Programming, 17th International Colloquium, Proceedings of ICALP 90*, number 443 in Lecture Notes in Computer Science, pages 268–282. Springer-Verlag, 1990. Extended version online available at http://www.wisdom.weizmann.ac.il/~oded/PS/zk-comp.ps.

[26] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game—a completeness theorem for protocols with honest majority. In *Nineteenth Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1987*, pages 218–229. ACM Press, 1987. Extended abstract.

[27] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[28] Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. Polynomial runtime in simulatability definitions. In *18th IEEE Computer Security Foundations Workshop, Proceedings of CSFW 2005*, pages 156–169. IEEE Computer Society, 2005. Online available at http://iaks-www.ira.uka.de/home/unruh/publications/continuously_polynomial.ps.

[29] Dennis Hofheinz and Dominique Unruh. Comparing two notions of simulatability. In Joe Kilian, editor, *Theory of Cryptography, Proceedings of TCC 2005*, number 3378 in Lecture Notes in Computer Science, pages 86–103. Springer-Verlag, 2005.

[30] Dennis Hofheinz and Dominique Unruh. On the notion of statistical security in simulatability definitions. In Jianying Zhou and Javier Lopez, editors, *Information Security, Proceedings of ISC 2005*, number 3650 in Lecture Notes in Computer Science, pages 118–133. Springer-Verlag, 2005. Online available at http://eprint.iacr.org/2005/032.ps.

[31] Jan Jürjens. Secure information flow for concurrent processes. In Catuscia Palamidessi, editor, *Concurrency Theory, Proceedings of CONCUR 2000*, number 1877 in Lecture Notes in Computer Science, pages 395–409. Springer-Verlag, 2000. Online available at http://wwwbroy.in.tum.de/~juerjens/papers/J00eWeb.ps.gz.

[32] Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *44th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2003*, pages 394–403. IEEE Computer Society, 2003. Full version online available at http://eprint.iacr.org/2003/141.ps.

[33] Daryl McCullough. Specifications for multi-level security and a hook-up property. In *IEEE Symposium on Security and Privacy, Proceedings of SSP '87*, pages 161–166. IEEE Computer Society, 1987.

[34] Daryl McCullough. Noninterference and the composability of security properties. In *IEEE Symposium on Security and Privacy, Proceedings of SSP '88*, pages 177–186. IEEE Computer Society, 1988.

[35] Silvio Micali and Phillip Rogaway. Secure computation. In Joan Feigenbaum, editor, *Advances in Cryptology, Proceedings of CRYPTO '91*, number 576 in Lecture Notes in Computer Science, pages 392–404. Springer-Verlag, 1992. Abstract.

[36] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *7th ACM Conference on Computer and Communications Security, Proceedings of CCS 2000*, pages 245–254. ACM Press, 2000. Extended version online available at `http://www.semper.org/sirene/publ/PfWa_00CompInt.ps.gz`.

[37] Birgit Pfitzmann and Michael Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *IEEE Symposium on Security and Privacy, Proceedings of SSP 2001*, pages 184–200. IEEE Computer Society, 2001. Full version online available at `http://eprint.iacr.org/2000/066.ps`.

[38] Dominik Raub, Jörn Müller-Quade, and Rainer Steinwandt. On the security and composability of the one time pad. In Peter Vojtás, Mária Bieliková, Bernadette Charron-Bost, and Ondrej Sýkora, editors, *Theory and Practice of Computer Science, Proceedings of SOFSEM 2005*, number 3381 in Lecture Notes in Computer Science, pages 288–297. Springer-Verlag, 2005. Extended version online available at `http://eprint.iacr.org/2004/113.ps`.

[39] Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, Massachusetts Institute of Technology, February 1996. Online available at `http://theory.lcs.mit.edu/~rivest/RivestShamirWagner-timelock.ps`.

[40] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions. In *23th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 1982*, pages 80–91. IEEE Computer Society, 1982.

[41] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 1986*, pages 162–167. IEEE Computer Society, 1986. Extended abstract.

[42] Aris Zakinthinos and E. Stewart Lee. The composability of non-interference. In *8th IEEE Computer Security Foundations Workshop, Proceedings of CSFW 1995*, pages 2–8. IEEE Computer Society, 1995.