Safe Equivalences for Security Properties

Mário S. Alvim, Miguel E. Andrés, Catuscia Palamidessi, Peter van Rossum

Safe Equivalences for Security Properties

Mário S. Alvim[1], Miguel E. Andrés[2], Catuscia Palamidessi[1], and Peter van Rossum[2].

[1]INRIA and LIX, École Polytechnique Palaiseau, France.
[2]Institute for Computing and Information Sciences, The Netherlands.

Abstract. In the field of Security, process equivalences have been used to characterize various information-hiding properties (for instance secrecy, anonymity and non-interference) based on the principle that a protocol P with a variable x satisfies such property if and only if, for every pair of secrets s_1 and s_2, $P[^{s_1}/_x]$ is equivalent to $P[^{s_2}/_x]$. We argue that, in the presence of nondeterminism, the above principle relies on the assumption that the scheduler "works for the benefit of the protocol", and this is usually not a safe assumption. Non-safe equivalences, in this sense, include trace equivalence and bisimulation. We present a formalism in which we can specify admissible schedulers and, correspondingly, safe versions of these equivalences. We then show safe bisimulation is still a congruence. We conclude showing how to use safe equivalences to characterize information-hiding properties.

1 Introduction

One of the fundamental problems in computer security is the protection from information leaks, namely how to make sure that a system does not reveal, by observations that can be made during or after the execution, some information that we wish to maintain confidential or secret.

One way to prevent an attacker to infer the secret from the observables is to create *noise*, namely to make sure that for every execution in which a given secret produces a certain observable, there is at least another execution in which a different secret produces the same observable. In practice this is often done by using randomization, see for instance the DCNet [7] and the Crowds [16] protocols.

In the literature about the foundations of Computer Security, however, the quantitative aspects are often abstracted away, and probabilistic behavior is replaced by nondeterministic behavior. Correspondingly, there have been various approaches in which information-hiding properties are expressed in terms of equivalences based on nondeterminism, especially in a concurrent setting. For instance, [17] defines *anonymity* as follows[1]: A protocol S is anonymous if, for every pair of culprits a and b, $S[^a/_x]$ and $S[^b/_x]$ produce the same observable traces. A similar definition is given in [1] for *secrecy*, with the difference that $S[^a/_x]$ and $S[^b/_x]$ are required to be bisimilar. In [9], an electoral system S preserves the *confidentiality of the vote* if for any voters v and w, the observable behavior of S is the same if we swap the votes of v and w. Namely, $S[^a/_v \,|^b\, /_w] \sim S[^b/_v \,|^a\, /_w]$, where $\sim$ represents bisimilarity.

[1] The actual definition of [17] is more complicated, but the spirit is the same.

These proposals are based on the implicit assumption that *all the nondeterministic executions present in the specification of S will always be possible under every implementation of S*. Or at least, that the adversary will believe so. In concurrency, however, as argued in [5], nondeterminism has a rather different meaning: if a specification S contains some nondeterministic alternatives, typically it is because we want to abstract from specific implementations, such as the scheduling policy. A specification is considered correct, with respect to some property, if every alternative satisfies the property. Correspondingly, an implementation is considered correct if all executions are among those possible in the specification, i.e. if the implementation is a refinement of the specification. There is no expectation that the implementation will actually make possible all the alternatives indicated by the specification.

We argue that the use of nondeterminism in concurrency corresponds to a *demonic* view: the scheduler, i.e. the entity that will decide which alternative to select, may try to choose the worst alternative. Hence we need to make sure that "all alternatives are good", i.e. satisfy the intended property. In the above mentioned approaches to the formalization of security properties, on the contrary, the interpretation of nondeterminism is *angelic*: the scheduler is expected to actually help the protocol to confuse the adversary and thus protect the secret information.

There is another issue, orthogonal to the angelic/demonic dichotomy, but relevant for the achievement of security properties: the scheduler *should not be able to make its choices dependent on the secret*, or else nearly every protocol would be insecure, i.e. the scheduler would always be able to leak the secret to an external observer (for instance by producing different interleavings of the observables, depending on the secret). This remark has been made several times already, and several approaches have been proposed to cope with the problem of the "almighty" scheduler (aka omniscient, clairvoyant, etc.), see for example [3,4,6,5,10].

The risk of a naive use of nondeterminism to specify a security property, is not only that it may rely on an implicit assumption that the scheduler behaves angelically, but also that it is clairvoyant, i.e. that it peeks at the secrets (that it is not supposed to be able to see) to achieve its angelic strategy.

Example 1. Consider the following system, in a CCS-like syntax:

$$S \stackrel{\text{def}}{=} (c, out)(A \parallel H_1 \parallel H_2 \parallel Corr),$$

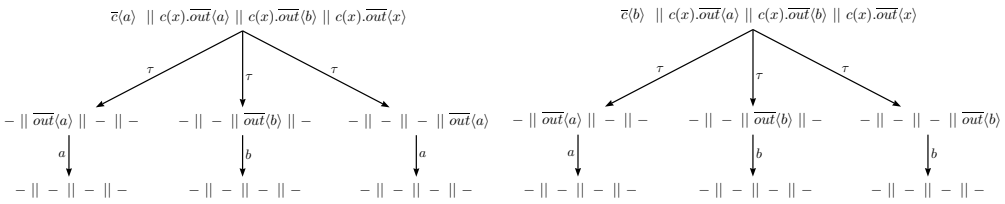
$$A \stackrel{\text{def}}{=} \overline{c}\langle sec\rangle, \quad H_1 \stackrel{\text{def}}{=} c(s).\overline{out}\langle a\rangle, \quad H_2 \stackrel{\text{def}}{=} c(s).\overline{out}\langle b\rangle, \quad Corr \stackrel{\text{def}}{=} c(s).\overline{out}\langle s\rangle$$

where $\parallel$ is the parallel operator, $\overline{c}\langle sec\rangle$ is a process that sends sec on channel c, $c(s).P$ is a process that receives s on channel c and then continues as P, and (c, out) is the restriction operator, enforcing synchronization on c and out. In this example, sec represents a secret information.

Figures 1(a) and 1(b) show the automaton representation of $S\left[{}^{a}/_{sec}\right]$ and $S\left[{}^{b}/_{sec}\right]$ respectively. For the sake of legibility we omit the restriction and we use the symbol '$-$' to denote a component that is stuck.

It is easy to see that we have $S\left[{}^{a}/_{sec}\right] \sim S\left[{}^{b}/_{sec}\right]$. Note that, in order to simulate the third branch in $S\left[{}^{a}/_{sec}\right]$, the process $S\left[{}^{b}/_{sec}\right]$ needs to select its first branch. Viceversa, in order to simulate the third branch in $S\left[{}^{b}/_{sec}\right]$, the process $S\left[{}^{a}/_{sec}\right]$ needs to

select its second branch. This means that, in order to achieve bisimulation, the scheduler needs to know the secret, and change its choice accordingly.



(a) (b)

Fig. 1. Automata representation of $S\left[{}^{a}/_{sec}\right]$ and $S\left[{}^{b}/_{sec}\right]$

This example shows a system that intuitively is not secure, because the third component, $Corr$, reveals whatever secret it receives. However, according to the equivalence-based notions of security discussed above, *it is secure*. But it is secure thanks to a scheduler that angelically helps the system to protect the secret, and it does so by making its choices dependent on the secret! These assumptions on the scheduler seem too strong.

We do not claim, however, that we should rule out the use of angelic nondeterminism in security: on the contrary, angelic nondeterminism can be a powerful specification concept. We only advocate a cautious use of this notion. In particular, it should not be used in a context in which the scheduler may be in collusion with the attacker. The goal of this paper is to define a framework in which we can combine both angelic and demonic nondeterminism in a setting in which also probabilistic behavior may be present, and in a context in which the scheduler is restricted (i.e. not clairvoyant). We define "safe" variant of typical equivalence relations (complete traces and bisimulation), and we show how to use them to characterize information-hiding properties.

1.1 Contribution

The main novelties of our work can be articulated as follows:

- We propose a formalism for concurrent systems which accounts for both probabilistic and nondeterministic behaviour, and in which the latter is of two kinds: *global* and *local*. The first represents the possible interleavings produced by the parallel components, which may be influenced by the attacker. The second is associated to the possible choices internal to each component, which may depend on the secrets or other unknown parameters, not controlled by the attacker. Correspondingly, we split the scheduler in two constituents: global and local. The latter is actually a tuple of local schedulers, one for each component of the system.
- We define, for the above systems, safe versions of trace equivalence and bisimilarity especially tuned for security. This means that we account for the possibility that the global constituent of the scheduler is in collusion with the attacker, and therefore does not necessarily help the system to obfuscate the secret. We show that the latter is still a congruence, like in the classical case.

- We propose a notion of *admissible scheduler* for the above systems, in which the global constituent is not allowed to see the secrets, and each local constituent is not allowed to see any information about the other components. We then generalize the standard definition of strong (probabilistic) information hiding (such as no-interference and strong anonymity) to the case in which also nondeterminism is present, under the assumption that the schedulers are admissible.
- We finally show that our notions of safe trace equivalence and bisimilarity imply strong information hiding in the above sense.

1.2 Related Work

While the proposal of combining angelic and demonic nondeterminism is – we believe– rather new, the problem of the almighty / clairvoyant scheduler has already been extensively investigated in literature. [3] and [4] work in the framework of probabilistic automata and introduce a restriction on the scheduler to the purpose of making them suitable to applications in security protocols. Their approach is based on dividing the actions of each component of the system in equivalence classes (*tasks*). The order of execution of different tasks is decided in advance by a so-called *task scheduler*. The remaining nondeterminism within a task is resolved by a second scheduler, which models the standard *adversarial scheduler* of the cryptographic community. This second entity has limited knowledge about the other components: it sees only the information that they communicate during execution. Hence these works introduce a separation of the scheduler in two parts like we do, however their notion of task scheduler is much more restricted than our notion of global scheduler.

Another work along these lines is [8], which uses partitions on the state-space to obtain partial-information schedulers. However that work considers a synchronous parallel composition, so the setting is rather different from ours.

In [10] it has been proposed a notion of system and admissible scheduler very similar to our notion of system and admissible global scheduler. The main difference is that in that work the components are deterministic and therefore there is no notion of local scheduler.

The work in [6, 5] is similar to ours in spirit, but in a sense *dual* from a technical point of view. Instead of defining a restriction on the class of schedulers, they provide a way to specify that a choice is transparent to the scheduler. They achieve this by introducing labels in process terms, used to represent both the states of the execution tree and the next action or step to be scheduled. They make two states indistinguishable to schedulers, and hence the choice between them private, by associating to them the same label. Furthermore, their "equivalence classes" (schedulable actions with the same label) can change dynamically, because the same action can be associated to different labels during the execution. So, their framework seems more general than ours, concerning the class of scheduler they can represent. More precisely, we believe that every scheduler in our formalism can be expressed in theirs, too. In [5] they also consider the problem of defining a safe version of bisimulation for expressing security properties. They call it *demonic bisimulation*. The main difference between our notion and theirs is that we consider a combination of angelic and demonic interpretation of nondeterminism, and this affects also the definition of bisimulation. Similarly, our definition

of leakage-freedom reflects this combination. In the approach of [5] the aspect of angelicity is not considered, although they may be able to simulate it with an appropriate labeling.

The fact that almighty / clairvoyant schedulers are unrealistic has also been observed in fields other than security. With the aim to cope with general properties (not only those concerning security), first attempts used restricted schedulers in order to obtain rules for compositional reasoning [8]. The justification for those restricted schedulers is the same as for ours, namely, that not all information is available to all entities in the system. Later on, it was shown that model checking was unfeasible in its general form for the restricted schedulers in [8] (see [12] and, more recently, [11]). Despite of undecidability, not all results concerning such schedulers have been negative as, for instance, the technique of partial-order reduction can be improved by assuming that schedulers can only use partial information [13].

1.3 Plan of the paper

Looking ahead, after reviewing some preliminaries on probabilistic automata (Section 2) we formalize the notions of systems and components (Section 3). In Section 4 we present our proposal for safe equivalences. We then turn our attention to restricting the discerning power of global and local schedulers (Section 5). Finally, we define the notion of information hiding under the novel assumption that nondeterminism is handled partly in a demonic way and partly in an angelical way (Section 6). In Section 7 we conclude and outline some future work.

2 Preliminaries

In this section we gather preliminary notions and results related to probabilistic automata [19, 18].

A function $\mu\colon Q \to [0,1]$ is a *discrete probability distribution* on a set Q if the support of μ is countable and $\sum_{q\in Q} \mu(q) = 1$. The set of all discrete probability distributions on Q is denoted by $\mathcal{D}(Q)$.

A *probabilistic automaton* is a quadruple $M = (Q, \Sigma, \hat{q}, \alpha)$ where

- Q is a countable set of *states*,
- Σ a finite set of *actions*,
- $\hat{q}$ the *initial* state, and
- α a *transition function* $\alpha : Q \to \mathcal{P}(\Sigma \times \mathcal{D}(Q))$.

Here $\mathcal{P}(X)$ is the set of all finite subsets of X.

If $\alpha(q) = \emptyset$ then q is a *terminal* state. We write $q \xrightarrow{a} \mu$ for $(a, \mu) \in \alpha(q)$. Moreover, we write $q \xrightarrow{a} r$ whenever $q \xrightarrow{a} \mu$ and $\mu(r) > 0$. A *fully probabilistic automaton* is a probabilistic automaton satisfying $|\alpha(q)| \leq 1$ for all states. In case $\alpha(q) \neq \emptyset$ in a fully probabilistic automaton, we will overload notation and use $\alpha(q)$ to denote the distribution outgoing from q.

A *path* in a probabilistic automaton is a sequence $\sigma = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots$ where $q_i \in Q$, $a_i \in \Sigma$ and $q_i \xrightarrow{a_{i+1}} q_{i+1}$. A path can be *finite* in which case it ends with a state.

A path is *complete* if it is either infinite or finite ending in a terminal state. Given a path σ, $\mathit{first}(\sigma)$ denotes its first state, and if σ is finite then $\mathit{last}(\sigma)$ denotes its last state. Let $\mathrm{Paths}_q(M)$ denote the set of all paths, $\mathrm{Paths}^\star_q(M)$ the set of all finite paths, and $\mathrm{CPaths}_q(M)$ the set of all complete paths of an automaton M, starting from the state q. We will omit q if $q = \hat{q}$. Paths are ordered by the prefix relation, which we denote by $\leq$. The *trace* of a path is the sequence of actions in $\Sigma^\omega = \Sigma^* \cup \Sigma^\infty$ obtained by removing the states, hence for the above path σ we have $\mathit{trace}(\sigma) = a_1 a_2 \ldots$, we denote by $\mathit{Traces}(M)$ to the set of all traces of M, i.e. $\mathit{Traces}(M) \stackrel{\text{def}}{=} \{\mathit{trace}(\sigma) \mid \sigma \in \mathrm{Paths}(M)\}$. If $\Sigma' \subseteq \Sigma$, then $\mathit{trace}_{\Sigma'}(\sigma)$ is the projection of $\mathit{trace}(\sigma)$ on the elements of Σ'. The *length* of a finite path σ, denoted by $|\sigma|$, is the number of actions in its trace.

Let $M = (Q, \Sigma, \hat{q}, \alpha)$ be a (fully) probabilistic automaton, $q \in Q$ a state, and let $\sigma \in \mathrm{Paths}^\star_q(M)$ be a finite path starting in q. The *cone* generated by σ is the set of complete paths $\langle\sigma\rangle = \{\sigma' \in \mathrm{CPaths}_q(M) \mid \sigma \leq \sigma'\}$. Given a fully probabilistic automaton $M = (Q, \Sigma, \hat{q}, \alpha)$ and a state q, we can calculate the *probability value*, denoted by $\mathbf{P}_q(\sigma)$, of any finite path σ starting in q as follows: $\mathbf{P}_q(q) = 1$ and $\mathbf{P}_q(\sigma \xrightarrow{a} q') = \mathbf{P}_q(\sigma) \cdot \mu(q'),$ where $\mathit{last}(\sigma) \xrightarrow{a} \mu$.

Let $\Omega_q \stackrel{\text{def}}{=} \mathrm{CPaths}_q(M)$ be the sample space, and let $\mathcal{F}_q$ be the smallest σ-algebra generated by the cones. Then $\mathbf{P}_q$ induces a unique *probability measure* on $\mathcal{F}_q$ (which we will also denote by $\mathbf{P}_q$) such that $\mathbf{P}_q(\langle\sigma\rangle) = \mathbf{P}_q(\sigma)$ for every finite path σ starting in q. For $q = \hat{q}$ we write $\mathbf{P}$ instead of $\mathbf{P}_{\hat{q}}$.

A scheduler for a probabilistic automaton M is a function

$$\zeta \colon \mathrm{Paths}^\star(M) \to (\Sigma \times \mathcal{D}(Q) \cup \{\bot\})$$

such that for all finite path σ, if $\alpha(\mathit{last}(\sigma)) \neq \emptyset$ then $\zeta(\sigma) \in \alpha(\mathit{last}(\sigma))$, and $\zeta(\sigma) = \bot$ otherwise.
Hence, a scheduler ζ selects one of the available transitions in each state, and determines therefore a fully probabilistic automaton, obtained by pruning from M the alternatives that are not chosen by ζ. A scheduler is history dependent since it takes into account the path (history) and not only the current state. It may be partial, i.e. it may halt the execution at any time, since in general it gives a sub-probability distribution[2].

3 Systems

In this section we describe the kind of systems we are dealing with. We start by introducing a variant of probabilistic automata, that we call *Tagged Probabilistic Automata* (TPA). These systems are parallel compositions of probabilistic processes, called *components*. Each component is equipped with a unique identifier, called *tag*. Whenever a component (or a pair of components in case of synchronization) makes a step, the corresponding transition will be decorated with the associated tag (or pair of tags).

Similar systems have been already introduced in [10]. The main difference is that here the components are simple probabilistic automata, i.e. they may contain internal

[2] In this paper, however, we will consider only total schedulers, to be more in line with the standard semantics of CCS.

nondetermism, and each transition goes from a node and a label to a distributions over nodes. In [10] the components are fully probabilistic (except for the input guards, that may receive different values), and the secrets can appear only in a probabilistic choice.

3.1 Tagged Probabilistic Automata

We now formalize the notion of TPA.

Definition 1. *A* Tagged Probabilistic Automaton *is a tuple* $(Q, L, \Sigma, \hat{q}, \alpha)$*, where*

- Q *is a set of* states,
- L *is a set of* tags,
- Σ *is a set of* actions,
- $\hat{q} \in Q$ *is the* initial state,
- $\alpha \colon Q \to \mathcal{P}(L \times \Sigma \times \mathcal{D}(Q))$ *is a* transition function.

In the following we write $q \xrightarrow{l:a} \mu$ for $(\ell, a, \mu) \in \alpha(q)$, and we use $Enabled(q)$ to denote the tags of the components that are enabled to make a transition. Namely,

$$Enabled(q) \stackrel{\text{def}}{=} \{\ell \in L \mid \textit{there exists } a \in \Sigma, \mu \in \mathcal{D}(Q) \textit{ such that } q \xrightarrow{l:a} \mu\}$$

In these systems, we can decompose the scheduler in two: a *global scheduler*, which decides which component or pair of components makes the move next, and a *local scheduler*, which solves the internal nondeterminism of the selected component.

We assume that the local scheduler can select only a transition which is enabled, and that the global scheduler can only select a component among those which are enabled. This means that the execution does not stop unless all components are blocked (suspended or terminated). This is in line with the spirit of process algebra, and also with the tradition of Markov Decision Processes, but contrasts with that of the Probabilistic Automata of Lynch and Segala [19]. However, the results in this paper do not depend on this assumption; we could as well allow schedulers which decide to terminate the execution even though there are transitions enabled in the last state.

Definition 2. *Let* $M = (Q, L, \Sigma, \hat{q}, \alpha)$ *be a Tagged Probabilistic Automaton.*

- *A global scheduler for* M *is a function* $\zeta \colon \mathrm{Paths}^\star(M) \to (L \cup \{\bot\})$ *such that for all finite paths* σ*, if* $Enabled(last(\sigma)) \neq \emptyset$ *then* $\zeta(\sigma) \in Enabled(last(\sigma))$*, and* $\zeta(\sigma) = \bot$ *otherwise.*
- *A local scheduler for* M *is a function* $\xi \colon \mathrm{Paths}^\star(M) \to (L \times \Sigma \times \mathcal{D}(Q) \cup \{\bot\})$ *such that, for all finite paths* σ*, if* $\alpha(last(\sigma)) \neq \emptyset$ *then* $\xi(\sigma) \in \alpha(last(\sigma))$*, and* $\xi(\sigma) = \bot$ *otherwise.*
- *A global scheduler* ζ *and a local scheduler* ξ *for* M *are* compatible *if, for all finite paths* σ*,* $\xi(\sigma) = (\ell, a, \mu)$ *implies* $\zeta(\sigma) = \ell$*, and* $\xi(\sigma) = \bot$ *implies* $\zeta(\sigma) = \bot$*.*
- *A scheduler for* M *is a pair* (ζ, ξ) *of compatible global and local schedulers for* M*.*

3.2 Components

We are going to use a simple probabilistic process calculus (a sort of probabilistic version of CCS [14, 15]) to specify the components.

We assume a set of *actions* or *channel names* Σ with elements $a, a_1, a_2, \cdots$, including the special symbol τ denoting a *silent step*. With the exception of τ, each action a has a unique co-action $\bar{a} \in \Sigma$ and we assume $\bar{\bar{a}} = a$.

A component q is a process specified by the following grammar:

Components	$q ::= 0$		termination
		$a.q$	prefix
	$\mid$	$q_1 + q_2$	nondeterministic choice
	$\mid$	$\sum_i p_i : q_i$	probabilistic choice
	$\mid$	$q_1 \mid q_2$	parallel composition
	$\mid$	$(a)q$	restriction
	$\mid$	A	process call

The p_i, in the blind and secret choices, represents the probability of the i-th branch and must satisfy $0 \leq p_i \leq 1$ and $\sum_i p_i = 1$. The process call A is a simple process identifier. For each identifier, we assume a corresponding unique process declaration of the form $A \stackrel{\text{def}}{=} q$. The idea is that, whenever A is executed, it triggers the execution of q. Note that q can contain A or another process identifier, which means that our language allows (mutual) recursion. We will denote by $fn(q)$ the *free channel names* occurring in q, i.e. the channel names not bound by a restriction operator.

Components' semantics: The operational semantics consists of probabilistic transitions of the form $q \stackrel{a}{\rightarrow} \mu$ where $q \in Q$ is a process, $a \in \Sigma$ is an action and $\mu \in \mathcal{D}(Q)$ is a distribution on processes. They are specified by the following rules:

$$\text{PRF}\ \frac{}{a.q \stackrel{a}{\rightarrow} \delta_q} \qquad \text{NDT}\ \frac{q_1 \stackrel{a}{\rightarrow} \mu}{q_1 + q_2 \stackrel{a}{\rightarrow} \mu}$$

$$\text{PRB}\ \frac{}{\sum_i p_i : q_i \stackrel{\tau}{\rightarrow} \sum_i p_i \cdot \delta_{q_i}} \qquad \text{PAR}\ \frac{q_1 \stackrel{a}{\rightarrow} \mu}{q_1 \mid q_2 \stackrel{a}{\rightarrow} \mu \mid q_2}$$

$$\text{CALL}\ \frac{q \stackrel{a}{\rightarrow} \mu}{A \stackrel{a}{\rightarrow} \mu}\ \text{if } A \stackrel{\text{def}}{=} q \qquad \text{COM}\ \frac{q_1 \stackrel{a}{\rightarrow} \delta_{r_1} \quad q_2 \stackrel{\bar{a}}{\rightarrow} \delta_{r_2}}{q_1 \mid q_2 \stackrel{\tau}{\rightarrow} \delta_{r_1 | r_2}}$$

$$\text{RST}\ \frac{q \stackrel{a}{\rightarrow} \mu}{(b)q \stackrel{a}{\rightarrow} (b)\mu}\quad a, \bar{a} \neq b$$

We assume also the symmetric versions of the rules NDT, PAR and COM. The symbol δ_q is the delta of Dirac, which assigns probability 1 to q and 0 to all other processes.

The symbol $\sum\!\!\!\!\!\!\sum_i$ represents summation on distributions. Namely, $\sum\!\!\!\!\!\!\sum_i p_i \cdot \mu_i$ is the distribution μ such that $\mu(x) = \sum_i p_i \cdot \mu_i(x)$. The notation $\mu \mid q$ represents the distribution μ' such that $\mu'(r) = \mu(q')$ if $r = q' \mid q$, and $\mu'(r) = 0$ otherwise. Similarly, $(b)\mu$ represents the distribution μ' such that $\mu'(q) = \mu(q')$ if $q = (b)q'$, and $\mu'(q) = 0$ otherwise.

3.3 Systems

A system is composed by n processes (components) in parallel, and restricted at the top-level on a subset of actions $A \subseteq \Sigma$:

$$(A)\ q_1 \parallel q_2 \parallel \cdots \parallel q_n.$$

The restriction on A enforces synchronization on the channel names belonging to A, in accordance with the CCS spirit.

Systems' semantics The semantics of a system gives rise to a TPA, where the states are terms representing systems during their evolution. A transition now is of the form $q \xrightarrow{\ell:a} \mu$ where $a \in \Sigma$, $\mu \in \mathcal{D}(Q)$, and $\ell \in L$ is either the tag of the component which makes the move, or a (unordered) pair of tags representing the two partners of a synchronization. We will set L to be the indexes of the components, i.e. $L = I \cup I^2$ where $I = \{1, 2, \ldots, n\}$.

Interleaving

$$\frac{q_i \xrightarrow{a} \sum\!\!\!\!\!\!\sum_j p_j \cdot \delta_{q_{ij}}}{(A)\ q_1 \parallel \cdots \parallel q_i \parallel \cdots \parallel q_j \parallel \cdots \parallel q_n \xrightarrow{i:a} \sum\!\!\!\!\!\!\sum_j p_j \cdot \delta_{(A)q_1 \parallel \cdots \parallel q_{ij} \parallel \cdots \parallel q_n}} \quad a \notin A$$

where i is the tag indicating that the component i is making the step.

Note that we assume that the probabilistic choices in the syntax of the components are finite. This implies that every transition $q \xrightarrow{\ell:a} \mu$ can be written as $q \xrightarrow{\ell:a} \sum\!\!\!\!\!\!\sum_i p_i \cdot \delta_{q_i}$, thus justifying the notation used in the interleaving rule.

Synchronization

$$\frac{q_i \xrightarrow{a} \delta_{q_i'} \qquad q_j \xrightarrow{\bar{a}} \delta_{q_j'}}{(A)\ q_1 \parallel \cdots \parallel q_i \parallel \cdots \parallel q_j \parallel \cdots \parallel q_n \xrightarrow{\{i,j\}:\tau} \delta_{(A)q_1 \parallel \cdots \parallel q_i' \parallel \cdots \parallel q_j' \parallel \cdots \parallel q_n}}$$

here $\{i, j\}$ is the tag indicating that the components making the step are i and j. Note that we write the pair (i, j) as $\{i, j\}$ in order to make explicit that the pair is unordered. In addition, sometimes we will write i, j instead of $\{i, j\}$, for simplicity.

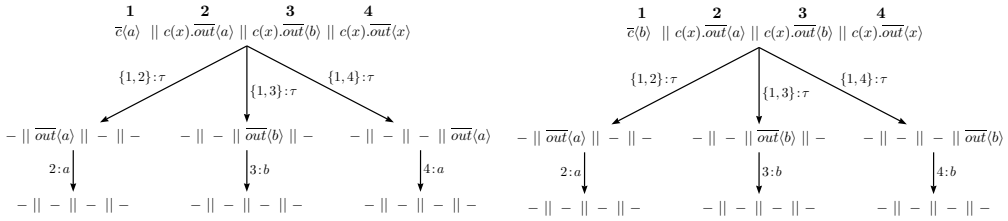
Example 2. We now show the TPA associated to the system S presented in the Introduction. Figure 2(a) shows TPA of $S\,[^a/_{sec}]$. For simplicity we do not write neither the restriction on channels c and out nor the termination symbol 0 of each component; furthermore, we use the symbol '$-$' to denote a component that is stuck. The

corresponding tags are indicated in the figure with numbers above the components, for instance the component $\overline{c}\langle a\rangle$ has tag **1**. Similarly the semantic of $S\left[{}^{b}/_{sec}\right]$ is shown in Figure 2(b).

The set of enabled transitions becomes clear in the figures. For instance, we have $\mathit{Enabled}(S\left[{}^{b}/_{sec}\right]) = \{\{1,2\},\{1,3\},\{1,4\}\}$ and $\mathit{Enabled}(\ -\ ||\ \overline{out}\langle a\rangle\ ||\ -\ ||\ -\ ) = \{2\}$. Finally, the scheduler ζ defined as

$$\zeta(\sigma) \stackrel{\text{def}}{=} \begin{cases} \{1,4\} & if\ \sigma = S\left[{}^{a}/_{sec}\right], \\ 2 & if\ \sigma = S\left[{}^{a}/_{sec}\right] \xrightarrow{1,2:\tau} (\ -\ ||\ \overline{out}\langle a\rangle\ ||\ -\ ||\ -\ ), \\ 3 & if\ \sigma = S\left[{}^{a}/_{sec}\right] \xrightarrow{1,3:\tau} (\ -\ ||\ -\ ||\ \overline{out}\langle b\rangle\ ||\ -\ ), \\ 4 & if\ \sigma = S\left[{}^{a}/_{sec}\right] \xrightarrow{1,4:\tau} (\ -\ ||\ -\ ||\ -\ ||\ \overline{out}\langle a\rangle\ ), \\ \bot & otherwise, \end{cases}$$

is a global scheduler for $S\left[{}^{a}/_{sec}\right]$.



(a) (b)

Fig. 2. Automata $S\left[{}^{a}/_{sec}\right]$ and $S\left[{}^{b}/_{sec}\right]$

4 Safe equivalences

In this section we revise some of the main equivalence notions used in literature so to guarantee their safe use in security.

4.1 Safe Traces

We define here a safe version of complete-trace semantics. The idea is that we want to compare two processes on the $\mathrm{CPaths}(M)$ basis not only of their traces, but also of the choices that the global scheduler makes at every step. One way of doing this is by recording explicitly the tags in the traces.

Definition 3.

- *Given a TPA $M = (Q, L, \Sigma, \hat{q}, \alpha)$ the (complete) safe traces of M, denoted here by $\mathit{Traces}_s(M)$, are defined as the probabilities of sequences of tags and actions corresponding to all possible complete executions, i.e.*

$$\mathit{Traces}_s(M) = \{\ f : (L \times \Sigma)^\omega \to [0,1]\ \mid \\ \text{there exists a scheduler } (\zeta,\xi) \text{ for } M \text{ s.t., for all } t \in (L\times\Sigma)^\omega, \\ f(t) = \mathbf{P}_{M,\zeta,\xi}(\{\sigma \in \mathrm{CPaths}(M) \mid \mathit{trace}_s(\sigma) = t\})\quad \}$$

where $\mathbf{P}_{M,\zeta,\xi}$ *is the probability measure in* M *under* (ζ, ξ)*, and* $trace_s$ *extracts from a path the sequence of tags and actions, i.e.* $trace_s(\epsilon) = \epsilon$ *(on the empty path* $trace_s$ *gives the empty string) and* $trace_s(q \xrightarrow{\ell:a} \sigma) = \ell : a \cdot trace_s(\sigma)$.

- *Given a system* q*, we will denote by* $Traces_s(q)$ *the safe traces of the automaton associated to* q.
- *Two systems* q_1 *and* q_2 *are safe-trace equivalent, denoted by* $q_1 \simeq_s q_2$*, if and only if* $Traces_s(q_1) = Traces_s(q_2)$.

It is clear that safe-trace equivalence is at least as discriminating as the standard (complete-) trace equivalence, denoted here by $\simeq$, which compares only the sequences of actions. In fact, the latter is obtained from the former by abstracting from the tags. The following example points out the converse does not hold.

Example 3. Consider the system S given in the introduction. There we have

$$Traces(S\,[^{a}/_{sec}]) = \{a, b\} = Traces(S\,[^{b}/_{sec}]).$$

On the other hand we have $Traces_s(S\,[^{a}/_{sec}]) = \{f_1, f_2, f_3\}$ where $f_1(\{1,2\} : \tau \cdot 2 : a) = f_2(\{1,3\} : \tau \cdot 3 : a) = f_3(\{1,4\} : \tau \cdot 4 : a\}) = 1$, and $f_i(t) = 0$ otherwise (for $i \in \{1,2,3\}$), while $Traces_s(S\,[^{b}/_{sec}]) = \{f_1, f_2, f_4\}$ with f_1, f_2 are as above, and $f_4(\{1,4\} : \tau \cdot 4 : b) = 1$, $f_4(t) = 0$ otherwise.

4.2 Safe Bisimilarity

In this section we propose a security-safe version of bisimulation, that we call *safe bisimulation*. This is an equivalence relation stricter than safe-trace equivalence, with the advantage of being a congruence.

We start with some notation. Given a TPA $M = (Q, L, \Sigma, \hat{q}, \alpha)$, and a global scheduler ζ, we denote by α_ζ the restriction of α to ζ, i.e. for every $q \in Q$,

$$\begin{aligned}\alpha_\zeta(q) \;=\; \{(a,\mu) \mid\; & \text{there exists } \ell \in L \text{ and } \sigma \in \mathrm{Paths}^\star_q(M) \text{ such that} \\ & (\ell, a, \mu) \in \alpha(q),\; q = last(\sigma),\; \text{and } \zeta(\sigma) = \ell \quad \}.\end{aligned}$$

We will also write $q \xrightarrow{a}_\zeta \mu$ for $(a,\mu) \in \alpha_\zeta(q)$, and M_ζ for the automaton obtained by pruning M from all the choices not compatible with ζ, i.e. $M_\zeta = (Q, L, \Sigma, \hat{q}, \alpha_\zeta)$. Note that M_ζ still contains nondeterminism, since there may be μ_1, μ_2, with $\mu_1 \neq \mu_2$, such that $(a_1, \mu_1), (a_2, \mu_2) \in \alpha_\zeta$ (with either $a_1 = a_2$ or $a_1 \neq a_2$).

We now define the notion of safe bisimulation. The idea is that, if q and q' are bisimilar states, then every move from q should be mimicked by a move from q' *using the same scheduler*.

Definition 4. *Given a TPA* $M = (Q, L, \Sigma, \hat{q}, \alpha)$*, we say that a relation* $\mathcal{R} \subseteq Q \times Q$ *is a safe bisimulation if, whenever* $q_1\,\mathcal{R}\,q_2$*, then:*

- $Enabled(q_1) = Enabled(q_2)$*, and*
- *for all global schedulers* ζ *for* M *such that* $\zeta(\sigma_1) = \zeta(\sigma_2)$ *whenever* $last(\sigma_1) = q_1$ *and* $last(\sigma_2) = q_2$*:*

- *if* $q_1 \xrightarrow{a}_\zeta \mu_1$, *then there exists* μ_2 *such that* $q_2 \xrightarrow{a}_\zeta \mu_2$ *and* $\mu_1 \mathcal{R} \mu_2$, *and*
- *if* $q_2 \xrightarrow{a}_\zeta \mu_2$, *then there exists* μ_1 *such that* $q_1 \xrightarrow{a}_\zeta \mu_1$ *and* $\mu_1 \mathcal{R} \mu_2$,

where $\mu_1 \mathcal{R} \mu_2$ *means that for all equivalence classes* $X \in Q_{\hat{\mathcal{R}}}$, *we have* $\mu_1(X) = \mu_2(X)$, *where* $\hat{\mathcal{R}}$ *is the smallest equivalence class induced by* $\mathcal{R}$.

The following result is immediate:

Proposition 1. *The union of all the safe bisimulations for* M *is still a safe bisimulation for* M.

Therefore the largest safe bisimulation exists, and coincides with the union of all safe bisimulations. We call it *safe bisimilarity*, and we denote it by $\sim_s$.

Given two TPA's on the same L and Σ, $M_1 = (Q_1, L, \Sigma, \hat{q}_1, \alpha_1)$ and $M_2 = (Q_2, L, \Sigma, \hat{q}_2, \alpha_2)$, we can define bisimulation and bisimilarity across their states, i.e. as relations on $(Q_1 \cup Q_2)$, in the obvious way, by constructing the TPA M with a new initial state $\hat{q}$ and two transitions to $\delta_{\hat{q}_1}$ and to $\delta_{\hat{q}_2}$, respectively.

Given two components or systems, q_1 and q_2, we will say that q_1 and q_2 are safely bisimilar, denoted by $q_1 \sim_s q_2$, if the initial states of the corresponding TPA's are safely bisimilar. Note that $q_1 \sim_s q_2$ is possible only if q_1 and q_2 have the same number of active components, where "active", for a component, means that during the execution of the system it will make at least one step.

Note that in the case of components, or of systems constituted by one component only, safe bisimulation and safe bisimilarity coincide with standard bisimulation and bisimilarity, respectively. For systems, safe bisimulation is at least as strong as standard bisimulation (denoted by $\sim$):

Remark 1. Given two systems q_1 and q_2, if $q_1 \sim_s q_2$ then $q_1 \sim q_2$.

The converse does not hold in general, as shown by the following example.

Example 4. We consider again the system S presented in the Introduction. It is easy to see that $S\left[{}^{a}/_{sec}\right] \sim S\left[{}^{b}/_{sec}\right]$. In order to show that $S\left[{}^{a}/_{sec}\right] \not\sim_s S\left[{}^{b}/_{sec}\right]$ we construct a new TPA (as described before) with initial state $\hat{q}$ such that $\hat{q} \xrightarrow{\ell:\tau} S\left[{}^{a}/_{sec}\right]$ and $\hat{q} \xrightarrow{\ell:\tau} S\left[{}^{b}/_{sec}\right]$. Figures 2(a) and 2(b) show the TPAs associated to $S\left[{}^{a}/_{sec}\right]$ and $S\left[{}^{b}/_{sec}\right]$ respectively. Now consider the scheduler ζ such that

$$\zeta(\sigma) \stackrel{\text{def}}{=} \begin{cases} \ell & if\ \sigma = \hat{q}, \\ \{1,4\} & if\ \sigma = \hat{q} \xrightarrow{\ell:\tau} S\left[{}^{a}/_{sec}\right], \\ 2 & if\ \sigma = \hat{q} \xrightarrow{\ell:\tau} S\left[{}^{a}/_{sec}\right] \xrightarrow{1,2:\tau} (\ - \ || \ \overline{out}\langle a\rangle \ || \ - \ || - \ ), \\ 3 & if\ \sigma = \hat{q} \xrightarrow{\ell:\tau} S\left[{}^{a}/_{sec}\right] \xrightarrow{1,3:\tau} (\ - \ || \ - \ || \ \overline{out}\langle b\rangle \ || - \ ), \\ 4 & if\ \sigma = \hat{q} \xrightarrow{\ell:\tau} S\left[{}^{a}/_{sec}\right] \xrightarrow{1,4:\tau} (\ - \ || \ - \ || \ - \ || \ \overline{out}\langle a\rangle \ ), \\ \{1,4\} & if\ \sigma = \hat{q} \xrightarrow{\ell:\tau} S\left[{}^{b}/_{sec}\right], \\ 2 & if\ \sigma = \hat{q} \xrightarrow{\ell:\tau} S\left[{}^{b}/_{sec}\right] \xrightarrow{1,2:\tau} (\ - \ || \ \overline{out}\langle a\rangle \ || \ - \ || - \ ), \\ 3 & if\ \sigma = \hat{q} \xrightarrow{\ell:\tau} S\left[{}^{b}/_{sec}\right] \xrightarrow{1,3:\tau} (\ - \ || \ - \ || \ \overline{out}\langle b\rangle \ || - \ ), \\ 4 & if\ \sigma = \hat{q} \xrightarrow{\ell:\tau} S\left[{}^{b}/_{sec}\right] \xrightarrow{1,4:\tau} (\ - \ || \ - \ || \ - \ || \ \overline{out}\langle b\rangle \ ), \\ \bot & otherwiase. \end{cases}$$

It is easy to see that $S\left[{}^{b}/_{sec}\right]$ cannot mimic the transition $4 : a$ produced by $S\left[{}^{a}/_{sec}\right]$ using the same scheduler ζ.

It turns out that safe bisimulation is a congruence with respect to all the operators of our language, as expressed by the following theorem. The reader should note that statements $2(a)$ and $2(b)$ are just the standard compositionality result for probabilistic bisimulation.

Theorem 1.

1. *$\sim_s$ is an equivalence relation.*
2. *Let $a \in \Sigma$ and $A, B, B' \subseteq \Sigma$. Let $p_1, \ldots, p_n$ be probability values, and let $q, q_1, q_2, \ldots, q_n, q'_1, q'_2, \ldots, q'_n$ be components.*
   (a) *If $q_1 \sim_s q_2$, then $a.q_1 \sim_s a.q_2$, $q_1 + q \sim_s q_2 + q$, $(a)q_1 \sim_s (a)q_2$, and $q_1 \mid q \sim_s q_2 \mid q$.*
   (b) *If $q_1 \sim_s q'_1, \ldots, q_n \sim_s q'_n$, then $\sum_i p_i : q_i \sim_s \sum_i p_i : q'_i$.*
   (c) *If $(B)\ q_1 \parallel \ldots \parallel q_n \sim_s (B')\ q'_1 \parallel \ldots \parallel q'_n$, and $fn(q) \notin B \cup B'$, then*

$$(A \cup B)\ q_1 \parallel \ldots \parallel q \parallel \ldots \parallel q_n \sim_s (A \cup B')\ q'_1 \parallel \ldots \parallel q \parallel \ldots \parallel q'_n.$$

The following property shows that bisimulation is stronger than safe-trace equivalence, like in the standard case.

Proposition 2. *For every pair of components or systems, q_1 and q_2, if $q_1 \sim_s q_2$ then $q_1 \simeq_s q_2$.*

Like in the standard case, the vice-versa does not hold, and safe-trace equivalence is not a congruence.

5 Admissible schedulers

In this section we restrict the discerning power of the global and local schedulers in order to avoid the problem of the information leakage induced in security by clairvoyant schedulers. We impose two kinds of restrictions: For the global scheduler, following the framework proposed in [10], we assume that it can only see, and keep memory of, the observable actions and the components that are enabled. It cannot see the secret actions and the internal choices of the various components. As for the local scheduler, we assume that the local nondeterminism of each component is solved on the basis of the local view of the history (local to that component), i.e. the projection of the history of the system on that component. In other words, each component has to make decisions based only on the history of its own execution; it cannot see anything of the other components.

5.1 Restricting Global Schedulers

We assume that the set of actions Σ is divided in two parts, the *secret actions* $\mathcal{S}$ and the *observable actions* $\mathcal{O}$. The secret actions are supposed to be invisible to the global scheduler. Formally, this can be achieved using a function $sift$ defined as:

$$sift(a) = \begin{cases} \tau \text{ if } a \in \mathcal{S} \\ a \text{ otherwise} \end{cases}$$

Then, we restrict the power of the global scheduler by forcing it to make the same decisions on paths he cannot tell apart, as formalized in the next definition.

Definition 5. *Given a TPA M, a global scheduler ζ for M is admissible if for all paths σ_1 and σ_2 we have $t(\sigma_1) = t(\sigma_2)$ implies $\zeta(\sigma_1) = \zeta(\sigma_2)$ where*

$$t\left(\hat{q} \xrightarrow{l_1:a_1} q_1 \xrightarrow{l_2:a_2} \cdots \xrightarrow{l_n:a_n} q_{n+1}\right) \stackrel{\text{def}}{=} \begin{array}{l} (Enabled(\hat{q}), sift(a_1), l_1) \\ (Enabled(q_1), sift(a_2), l_2) \\ \qquad\qquad \vdots \\ (Enabled(q_n), sift(a_n), l_n) \end{array}$$

The idea is that t sifts the information of the path that the scheduler can see. This way, since $sift$ "hides" the secrets to the scheduler, the scheduler cannot take different decisions based on secret information.

5.2 Restricting Local Schedulers

The restriction on the local scheduler is based on the idea that a step of the component i of a system can only be based on the view that i has of the history, i.e. its own history. In order to formalize this restriction, it is convenient to introduce the concept of i-view of a path σ, or *projection* of σ on i, which we will denote by $\sigma_{\restriction i}$. We define it inductively:

$$(\sigma \xrightarrow{\ell:a} \mu)_{\restriction i} = \begin{cases} \sigma_{\restriction i} \xrightarrow{i:b} \delta_{q_i} & \text{if } \ell = \{i,j\} \text{ and } \mu = \delta_{(A)\ q_1 \| \ldots \| q_i \| \ldots \| q_j \| \ldots \| q_n} \\ \sigma_{\restriction i} \xrightarrow{i:a} \mu & \text{if } \ell = i \\ \sigma_{\restriction i} & \text{otherwise} \end{cases}$$

In the above definition, the first line represents the case of a synchronization step involving the component i, where we assume that the premise for i is of the form $q_i' \xrightarrow{b} \delta_{q_i}$. The second line represents an interleaving step in which i is the active component. The third line represents step in which the component i is idle.

The restriction to the local scheduler can now be expressed as follows:

Definition 6. *Given a TPA M and a local scheduler ξ for M, we say that ξ is admissible if for all paths σ and σ', if $\xi(\sigma) = (\ell, a, \mu)$, and $\xi(\sigma') = (\ell', a', \mu')$ we have:*

- *if $\ell = \ell' = i$ and $\sigma_{\restriction i} = \sigma'_{\restriction i}$, then $\xi(\sigma) = \xi(\sigma')$,*
- *if $\ell = \ell' = \{i, j\}$, $\sigma_{\restriction i} = \sigma'_{\restriction i}$, and $\sigma_{\restriction j} = \sigma'_{\restriction j}$ then $\xi(\sigma) = \xi(\sigma')$.*

A pair of compatible schedulers (ζ, ξ) for M is called *admissible* if both ζ and ξ are admissible.

6 Safe Nondeterministic Information Hiding

In this section we define the notion of information hiding under the most general hypothesis that the nondeterminism is handled partly in a demonic way and partly in an angelic way. We assume that the demonic part is in the realm of the global scheduler, while the angelic part is controlled by the local scheduler. The motivation is that in a protocol the local components can be thought of as programs running locally in a single machine, and locally predictable and controllable, while the network can be subject to attacks that make the interactions unpredictable.

We recall that, in a purely probabilistic setting, the absence of leakage, such as no-interference and strong anonymity, is expressed as follows (see for instance [2]). Given a purely probabilistic automaton M, and a sequence $\tilde{a} = a_1 a_2 \ldots a_n$, let $\mathbf{P}_M([\tilde{a}])$ represent the probability measure of all complete paths with trace $\tilde{a}$ in M. Let S be a protocol containing a variable action $secr$, and let s be secret actions. Let M_s be the automaton corresponding to $S[^s/_{secr}]$. Define $Pr(\tilde{a} \mid s)$ as $\mathbf{P}_{M_s}([\tilde{a}])$. Then S is leakage-free if for every observable trace $\tilde{a}$, and for every secret s_1 and s_2, we have:

$$Pr(\tilde{a} \mid s_1) = Pr(\tilde{a} \mid s_2).$$

In a purely nondeterministic setting, on the other hand, the absence of leakage has been characterized in the literature by the following property:

$$S[^{s_1}/_{secr}] \cong S[^{s_2}/_{secr}]$$

where $\cong$ is an equivalence relation like trace equivalence, or bisimulation. As we have argued in the introduction, this definition assumes an angelic interpretation of nondeterminism.

We want to combine the above notions so to come with the case in which we have both probability and nondeterminism. Furthermore, we want to extend it to the case in which part of the nondeterminism is interpreted demonically. Let us first introduce some notation.

Let S be a system containing a variable action $secr$. Let s be a secret action. Let M_s be the TPA associated to $S[^s/_{secr}]$ and let (ζ, ξ) be a compatible pair of global and local schedulers for M_s. The probability of an observable trace $\tilde{a}$ given s is defined as

$$Pr_{\zeta,\xi}(\tilde{a} \mid s) = \mathbf{P}_{M_s,\zeta,\xi}([\tilde{a}]).$$

The global nondeterminism is interpreted demonically, and therefore we need to ensure that the conditional of an observable, given the two secrets, are calculated with respect to the same global scheduler. On the other hand, the local scheduler is interpreted angelically, and therefore we can compare the conditional probabilities generated by the two secrets as sets under different schedulers. In other words, we have the freedom to match conditional probability from the first set with one of the other set, without requiring the local scheduler to be the same.

Either angelic or demonic, we want to avoid the almightily / clairvoyant schedulers, i.e. a scheduler should not be able to use the secret information to achieve its goals. For this purpose, we require both the global and the local scheduler to be admissible.

Definition 7. *A system S is leakage-free if, for every pair of secrets s_1 and s_2, every admissible global scheduler ζ, and every observable trace $\tilde{a}$,*

$$\{Pr_{\zeta,\xi}(\tilde{a} \mid s_1) \mid \xi \textit{ admissible and compatible with } \zeta\} = \{Pr_{\zeta,\xi}(\tilde{a} \mid s_2) \mid \xi \textit{ admissible and compatible with } \zeta\}.$$

It turns out that the safe equivalences defined in Section 4 imply the absence of leakage:

Theorem 2. *Let S be a system with a variable action $secr$ and assume that $S[^{s_1}/_{secr}] \simeq_s S[^{s_2}/_{secr}]$ for every pair of secrets s_1 and s_2. Then S is leakage-free.*

From the above theorem and from Proposition 2, we also have the following corollary (with the same premises as the previous theorem):

Corollary 1. *If $S[^{s_1}/_{secr}] \sim_s S[^{s_2}/_{secr}]$ for every pair of secrets s_1 and s_2, then S is leakage-free.*

It actually turns out that we can restrict our definitions of safe equivalences to the case in which the global schedulers are admissible, as shown by the following theorem. This result is interesting because the resulting variants of safe trace equivalence and bisimilarity the number of transitions that we have to check is smaller, and therefore in principle it could be more efficient to prove that these equivalences hold.

Theorem 3. *Let $\simeq'_s$, $\sim'_s$ be the versions of $\simeq_s$, $\sim_s$ respectively, obtained by considering only admissible global schedulers. Let S be a system with a variable action $secr$ and assume that $S[^{s_1}/_{secr}] \simeq'_s S[^{s_2}/_{secr}]$ for every pair of secrets s_1 and s_2. Then S is leakage-free. The same holds if in the above we replace $\simeq'_s$ by $\sim'_s$.*

7 Conclusion and Future work

We have observed that some definitions of security properties based on process equivalences may be too naive, in that they assume the scheduler to be angelic, and, worse yet, to achieve its angelic strategy by peeking at the secrets. We have presented a formalism allowing us to specify a demonic constituent of the scheduler, possibly in collusion with the attacker, and an angelic one, under the control of the system. Correspondingly, we have defined "safe" equivalences. In particular we have defined the notions of safe trace equivalence and safe bisimilarity, and we have shown that the latter is still a congruence. We have also considered restrictions on the schedulers to limit the power of what they can see, and extended to our nondeterministic framework the (probabilistic) information-hiding properties like non interference and strong anonymity. Finally, we have shown that the safe equivalences can be used to prove these information-hiding properties.

For the future, we plan to extend our framework to consider quantitative notions of information leakage, possibly based on information theory. We also plan to implement model checking techniques to verify information hiding properties for systems in our framework. A natural candidate for the implementation would be PRISM. Of course, we would need to restrict the class of schedulers in PRISM so to meet the admissibility criteria.

Acknowledgement. The authors wish to thank Pedro D'Argenio for helpful discussion on the use of process equivalences in security.

References

1. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. and Comp.*, 148(1):1–70, 10Jan. 1999.
2. M. Bhargava and C. Palamidessi. Probabilistic anonymity. In *Proc. of CONCUR*, volume 3653 of *LNCS*, pages 171–185. Springer, 2005.
3. R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Task-structured probabilistic i/o automata. In *Proc. of WODES*, 2006.
4. R. Canetti, L. Cheung, D. K. Kaynar, M. Liskov, N. A. Lynch, O. Pereira, and R. Segala. Time-bounded task-PIOAs: A framework for analyzing security protocols. In *Proc. of DISC*, volume 4167 of *LNCS*, pages 238–253. Springer, 2006.
5. K. Chatzikokolakis, G. Norman, and D. Parker. Bisimulation for demonic schedulers. In *Proc. of FOSSACS*, volume 5504 of *LNCS*, pages 318–332. Springer, March 2009 2009.
6. K. Chatzikokolakis and C. Palamidessi. Making random choices invisible to the scheduler. In *Proc. of CONCUR*, volume 4703 of *LNCS*, pages 42–58. Springer, 2007.
7. D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.
8. L. de Alfaro, T. A. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In *Proc. of CONCUR*, volume 2154 of *LNCS*. Springer, 2001.
9. S. Delaune, S. Kremer, and M. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.
10. M. E.Andrés, C. Palamidessi, P. van Rossum, and A. Sokolova. Information hiding in probabilistic concurrent systems. Technical report, Radboud University Nijmegen, 2010. `http://www.cs.ru.nl/M.Andres/downloads/SAuN.pdf`.
11. S. Giro. Undecidability results for distributed probabilistic systems. In *Proc. of SBMF*, pages 220–235, 2009.
12. S. Giro and P. R. D'Argenio. Quantitative model checking revisited: Neither decidable nor approximable. In *Proc. of FORMATS*, pages 179–194, 2007.
13. S. Giro, P. R. D'Argenio, and L. M. F. Fioriti. Partial order reduction for probabilistic systems: A revision for distributed schedulers. In *Proc. of CONCUR*, pages 338–353, 2009.
14. R. Milner. *Communication and Concurrency*. Int. Series in Computer Science. Prentice Hall, 1989.
15. R. Milner. *Communicating and mobile systems: the π-calculus*. Cambridge University Press, 1999.
16. M. K. Reiter and A. D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
17. S. Schneider and A. Sidiropoulos. CSP and anonymity. In *Proc. of ESORICS*, volume 1146 of *LNCS*, pages 198–218. Springer, 1996.
18. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, June 1995. Tech. Rep. MIT/LCS/TR-676.
19. R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

Appendix

In this appendix we give the proofs of the results in the paper. We first need a lemma to justify the notation that we will use in some of these proofs.

Lemma 1. *Let M be a TPA, and let ζ be a global scheduler for M. Assume that for every pair of states q_1 and q_2, and paths σ_1 and σ_2, if $last(\sigma_1) = q_1$, and $last(\sigma_2) = q_2$, then $\zeta(\sigma_1) = \zeta(\sigma_2)$. Then ζ is history-independent, i.e. it depends only on the last state of a path σ.*

Proof. It is easy to see that the relation of having the same last state is an equivalence relation on paths, and therefore it determines a partition on the set of paths. Since the above q_1 and q_2 may be identical, the scheduler must give the same value on equivalent paths. □

The above lemma justifies restricting to history-independent schedulers in the definition of safe bisimulation (Definition 4), hence in the following results about safe bisimulation we will usually write $\zeta(q)$ where q is a state. Note however that this does not mean that computations of safely bisimilar systems are restricted by history-independent schedulers: at each step of the computation we may change scheduler, and therefore we may change alternative when we pass by the same state q at a later time.

Proposition 1. *The union of all the safe bisimulations for M is still a safe bisimulation for M.*

Proof. We will use $\sim_s$ to denote the union of all safe bisimulations, according to the notation introduced in the paper. Assume now that $q_1 \sim_s q_2$. Then $q_1 \mathcal{R} q_2$ holds, for some safe bisimulation $\mathcal{R}$. Hence we have $Enabled(q_1) = Enabled(q_2)$, and for every global scheduler ζ, if $\zeta(q_1) = \zeta(q_2)$, and $q_1 \xrightarrow{a}_\zeta \mu_1$, then there exists μ_2 such that $q_2 \xrightarrow{a}_\zeta \mu_2$, and $\mu_1 \mathcal{R} \mu_2$. This implies that $\mu_1 \sim_s \mu_2$. In fact $\hat{\mathcal{R}}$ is a finer relation than $\hat{\sim}_s$, i.e. $q_1 \hat{\mathcal{R}} q_2$ implies $q_1 \hat{\sim}_s q_2$. Furthermore $\hat{\mathcal{R}}$ is an equivalence relation, and therefore it induces a partition on each of the equivalence classes $X \in Q_{\hat{\sim}_s}$. Hence we have, for each $X \in Q_{\hat{\sim}_s}$, $\mu_1(X) = \sum_{Y \in X_{\hat{\mathcal{R}}}} \mu_1(Y) = \sum_{Y \in X_{\hat{\mathcal{R}}}} \mu_2(Y) = \mu_2(X)$.

We proceed analogously to show that, if $q_2 \xrightarrow{a}_\zeta \mu_2$, then there exists μ_1 such that $q_1 \xrightarrow{a}_\zeta \mu_1$ and $\mu_1 \sim_s \mu_2$. □

Theorem 1.

1. *$\sim_s$ is an equivalence relation.*
2. *Let $a \in \Sigma$ and $A, B, B' \subseteq \Sigma$. Let $p_1, \ldots, p_n$ be probability values, and let $q, q_1, q_2, \ldots, q_n, q'_1, q'_2, \ldots, q'_n$ be components.*
   (a) If $q_1 \sim_s q_2$, then $a.q_1 \sim_s a.q_2$, $q_1 + q \sim_s q_2 + q$, $(a)q_1 \sim_s (a)q_2$, and $q_1 \mid q \sim_s q_2 \mid q$.
   (b) If $q_1 \sim_s q'_1, \ldots, q_n \sim_s q'_n$, then $\sum_i p_i : q_i \sim_s \sum_i p_i : q'_i$.

(c) *If* $(B)\ q_1 \parallel \ldots \parallel q_n \ \sim_s \ (B')\ q'_1 \parallel \ldots \parallel q'_n$, *and* $fn(q) \notin B \cup B'$, *then*

$$(A \cup B)\ q_1 \parallel \ldots \parallel q \parallel \ldots \parallel q_n \ \sim_s \ (A \cup B')\ q'_1 \parallel \ldots \parallel q \parallel \ldots \parallel q'_n.$$

Proof.

1. Although safe bisimulations are not equivalence relations in general, their union, i.e. safe bisimilarity, is an equivalence. In fact:
   - It is easy to see that, if $\mathcal{R}$ is a safe bisimulation, then the smallest equivalence that includes $\mathcal{R}$, namely $\hat{\mathcal{R}}$, is also a safe bisimulation.
   - From Proposition 1 we know that $\sim_s$ is a safe bisimulation.
   - Hence we derive that $\hat{\sim}_s$ is a safe bisimulation, and therefore $\hat{\sim}_s \subseteq \sim_s$. But since obviously $\sim_s \subseteq \hat{\sim}_s$, we conclude that $\sim_s = \hat{\sim}_s$, which means that $\sim_s$ is already an equivalence relation.

2. Assume that $a, A, B, B', p_1, \ldots, p_n, q, q_1, q_2, \ldots, q_n, q'_1, q'_2, \ldots, q'_n$ are of the type prescribed by the hypothesis of the theorem.
   (a) Assume $q_1 \sim_s q_2$.
   - Let
   $$\mathcal{R} = \{(a.q_1, a.q_2)\} \cup \sim_s .$$
   We show that $\mathcal{R}$ is a safe bisimulation, which is sufficient to prove that $a.q_1 \sim_s a.q_2$. Note that, since there is only one component in each of those states, and it is enabled, we have $Enabled(a.q_1) = Enabled(a.q_2) = \{1\}$, and $\zeta(a.q_1) = \zeta(a.q_2) = 1$ for any global scheduler ζ. Given a global scheduler ζ, there is exactly one transitions from each of $a.q_1$ and $a.q_2$: these are $a.q_1 \stackrel{a}{\rightarrow}_\zeta \delta_{q_1}$ and $a.q_2 \stackrel{a}{\rightarrow}_\zeta \delta_{q_2}$ respectively, which mimic each other in the action a. Finally, since $q_1 \sim_s q_2$, we have $\delta_{q_1} \sim_s \delta_{q_2}$ and therefore $\delta_{q_1} \mathcal{R} \delta_{q_2}$.

   - Let
   $$\mathcal{R} = \{(q_1 + q, q_2 + q)\} \cup \sim_s .$$
   We show that $\mathcal{R}$ is a safe bisimulation, which is sufficient to prove that $q_1 + q \sim_s q_2 + q$. We have that $Enabled(q_1 + q) = Enabled(q_1) \cup Enabled(q) = Enabled(q_2) \cup Enabled(q) = Enabled(q_2 + q)$, in fact $Enabled(q_1) = Enabled(q_2)$ since $q_1 \sim_s q_2$. Correspondingly, given a global scheduler ζ, we have either $\zeta(q_1 + q) = \zeta(q_2 + q) = 1$ or $\zeta(q_1 + q) = \zeta(q_2 + q) = \perp$, since there is only one component. Assume $q_1 + q \stackrel{a}{\rightarrow}_\zeta \mu_1$. We have two cases: either $q_1 \stackrel{a}{\rightarrow}_\zeta \mu_1$, or $q \stackrel{a}{\rightarrow}_\zeta \mu_1$. The second case is obvious. In the first case, since $q_1 \sim_s q_2$, we have that also $q_2 \stackrel{a}{\rightarrow}_\zeta \mu_2$, with $\mu_1 \sim_s \mu_2$. We derive that $\mu_1 \mathcal{R} \mu_2$. For the transitions from $q_2 + q$ we proceed in the analogous way.

   - Let
   $$\mathcal{R} = \{((a)q_1, (a)q_2) \mid q_1 \sim_s q_2\}.$$
   We show that $\mathcal{R}$ is a safe bisimulation, which is sufficient to prove that, if $q_1 \sim_s q_2$, then $(a)q_1 \sim_s (a)q_2$. First observe that $Enabled((a)q_1) =$

$Enabled(q_1) = \{1\}$ if q_1 can make a transition with a label different from a, otherwise $Enabled((a)q_1) = \emptyset$. The same holds for $(a)q_2$. Since $q_1 \sim_s q_2$, we derive that $Enabled((a)q_1) = Enabled((a)q_2)$. Accordingly, given a global scheduler ζ, we have that either $\zeta((a)q_1) = \zeta((a)q_2) = 1$, or $\zeta((a)q_1) = \zeta((a)q_2) = \perp$. Assume $(a)q_1 \stackrel{b}{\rightarrow}_\zeta \mu_1$. Then we must have $b \neq a$ and $\mu_1 = (a)\mu_1'$, where $q_1 \stackrel{b}{\rightarrow}_\zeta \mu_1'$. Since $q_1 \sim_s q_2$, we have also $q_2 \stackrel{b}{\rightarrow}_\zeta \mu_2'$, with $\mu_1' \sim_s \mu_2'$. We derive that $(a)q_2 \stackrel{b}{\rightarrow}_\zeta (a)\mu_2'$, and $(a)\mu_1' \,\mathcal{R}\, (a)\mu_2'$.
We proceed in an analogous way for the transitions from $(a)q_2$.

- The case of the parallel operator in components is similar to the case of the parallel operator on systems (see the last item of this proof).

(b) Assume $q_1 \sim_s q_1', \ldots, q_n \sim_s q_n'$. Let

$$\mathcal{R} = \{(\sum_i p_i : q_i, \sum_i p_i : q_i')\} \cup \sim_s .$$

We show that $\mathcal{R}$ is a safe bisimulation, which is sufficient to prove that $\sum_i p_i : q_i \sim_s \sum_i p_i : q_i'$. Observe that both $\sum_i p_i : q_i$ and $\sum_i p_i : q_i'$ are enabled, and, since there is only one component, $Enabled(\sum_i p_i : q_i) = Enabled(\sum_i p_i : q_i') = \{1\}$. Accordingly, if ζ is a global scheduler, we have $Enabled(\sum_i p_i : q_i) = Enabled(\sum_i p_i : q_i') = 1$. Given a global scheduler ζ, the only transitions from $\sum_i p_i : q_i$ and $\sum_i p_i : q_i'$ are $\sum_i p_i : q_i \stackrel{\tau}{\rightarrow}_\zeta \sum_i p_i \cdot \delta_{q_i}$ and $\sum_i p_i : q_i' \stackrel{\tau}{\rightarrow}_\zeta \sum_i p_i \cdot \delta_{q_i'}$ respectively, which mimic each other in the action τ. It is easy to see that we have $(\sum_i p_i : q_i) \sim_s (\sum_i p_i : q_i')$, and therefore $(\sum_i p_i : q_i) \,\mathcal{R}\, (\sum_i p_i : q_i')$.

(c) Let

$$\mathcal{R} = \left\{ \begin{array}{r} ((A \cup B)\; q_1 \parallel \ldots \parallel q \parallel \ldots \parallel q_n \;,\; (A \cup B')\; q_1' \parallel \ldots \parallel q \parallel \ldots \parallel q_n') \mid \\ (B)\; q_1 \parallel \ldots \parallel q_n \;\sim_s\; (B')\; q_1' \parallel \ldots \parallel q_n' \end{array} \right\}$$

We show that $\mathcal{R}$ is a safe bisimulation, which is sufficient to prove that, if

$$(B)\; q_1 \parallel \ldots \parallel q_n \;\sim_s\; (B')\; q_1' \parallel \ldots \parallel q_n' \;,$$

then

$$(A \cup B)\; q_1 \parallel \ldots \parallel q \parallel \ldots \parallel q_n \;\sim_s\; (A \cup B')\; q_1' \parallel \ldots \parallel q \parallel \ldots \parallel q_n' \;.$$

Observe first that

$$Enabled((A \cup B)\; q_1 \parallel \ldots \parallel q \parallel \ldots \parallel q_n) = Enabled((A \cup B')\; q_1' \parallel \ldots \parallel q \parallel \ldots \parallel q_n').$$

In fact the enabled components are the same as those of $(B)\; q_1 \parallel \ldots \parallel q_n$ and of $(B')\; q_1' \parallel \ldots \parallel q_n'$ (modulo the index shift), which are equal by the

bisimilarity hypothesis, plus possibly the component q, plus possibly the synchronizations with q, which again are equal by the bisimilarity hypothesis, minus the transitions with labels in A. Note that the hypothesis $fn(q) \notin B \cup B'$ is essential here to guarantee that the component q is enabled (or disabled) in both sides.
Let us consider the synchronization case; the interleaving case is just a simplified variant. Given a global scheduler ζ, assume

$$\zeta((A \cup B)\ q_1 \parallel \ldots \parallel q \parallel \ldots \parallel q_n) = \zeta((A \cup B')\ q'_1 \parallel \ldots \parallel q \parallel \ldots \parallel q'_n).$$

Consider a move from the system in the left-hand side:

$$(A \cup B)\ q_1 \parallel \cdots \parallel q_i \parallel \cdots \parallel q_j \parallel \cdots \parallel q_n \xrightarrow{i,j:\tau} \delta_{(A)q_1\parallel\cdots\parallel r_i\parallel\cdots\parallel r_j\parallel\cdots\parallel q_n}.$$

Then we must have

$$q_i \xrightarrow{a} \delta_{r_i} \quad , \quad q_j \xrightarrow{\bar{a}} \delta_{r_j} \quad ,$$

where one of the q_i, q_j could be q, and

$$\zeta((A \cup B)\ q_1 \parallel \cdots \parallel q_i \parallel \cdots \parallel q_j \parallel \cdots \parallel q_n) = \{i, j\}.$$

Since $q_i \sim_s q'_i$ and $q_j \sim_s q'_j$ (in case $q_i = q$ then $q'_i = q$ and therefore $q_i \sim_s q'_i$ because $\sim_q$ is reflexive, and analogously for q_j), we must have

$$q'_i \xrightarrow{a} \delta_{r'_i} \quad , \quad q'_j \xrightarrow{\bar{a}} \delta_{r'_j} \quad ,$$

for some r'_i, r'_j such that $\delta_{r_i} \sim_s \delta_{r'_i}$ and $\delta_{r_j} \sim_s \delta_{r'_j}$. We derive that

$$(A \cup B)\ q'_1 \parallel \cdots \parallel q'_i \parallel \cdots \parallel q'_j \parallel \cdots \parallel q'_n \xrightarrow{i,j:\tau} \delta_{(A)q'_1\parallel\cdots\parallel r'_i\parallel\cdots\parallel r'_j\parallel\cdots\parallel q'_n}\ ,$$

and, since $\delta_{r_i} \sim_s \delta_{r'_i}$, $\delta_{r_j} \sim_s \delta_{r'_j}$ imply $r_i \sim_s r'_i$, $r_j \sim_s r'_j$, and by the definition of $\mathcal{R}$, we conclude

$$(\delta_{(A)q_1\parallel\cdots\parallel r_i\parallel\cdots\parallel r_j\parallel\cdots\parallel q_n})\ \mathcal{R}\ (\delta_{(A)q'_1\parallel\cdots\parallel r'_i\parallel\cdots\parallel r'_j\parallel\cdots\parallel q'_n}).$$

We proceed in an analogous way for the transitions from the right-hand side.

□

Proposition 2. *For every pair of components or systems, q_1 and q_2, if $q_1 \sim_s q_2$ then $q_1 \simeq_s q_2$.*

Proof. For this proof, it is convenient to consider a coinductive approximation of safe-trace equivalence. We start with a coinductive characterization of the safe traces. This in itself is not a key notion of the proof, but will help understanding the definition of the approximation.

Given a TPA $M = (Q, L, \Sigma, \hat{q}, \alpha)$, consider the operator

$$\mathcal{T}_{\mathrm{Tr}} : (Q \to \mathcal{P}(\mathrm{CPaths}(M) \to [0,1])) \to (Q \to \mathcal{P}(\mathrm{CPaths}(M) \to [0,1]))$$

defined as:

$$\begin{aligned}\mathcal{T}_{\mathrm{Tr}}(F)(q) = \{\ f : (L \times \Sigma)^\omega \to [0,1] \mid \\ \text{if } q \not\rightarrow \text{ then } f(\epsilon) = 1, \text{ else } f(\epsilon) = 0 \text{ and, for all } \ell \in L, a \in \Sigma, \\ \bullet \text{ if there exists } \mu \text{ s.t. } q \xrightarrow{\ell:a} \mu, \text{ then for each } q' \in Q \text{ there exists } f'_{q'} \in F(q') \text{ s.t.} \\ \text{for every } t \in (L \times \Sigma)^\omega, f(\ell : a \cdot t) = \textstyle\sum_{q'} \mu(q') f'_{q'}(t) \\ \bullet \text{ if } q \xrightarrow{\ell:a}, \text{ then } f(q)(\ell : a \cdot t) = 0 \quad \}\end{aligned}$$

where $q \not\rightarrow$ means that for all $\ell \in L, a \in \Sigma$, we have $q \xrightarrow{\ell:a}$.

Consider the ordering $\sqsubseteq$ on $Q \to \mathcal{P}(\mathrm{CPaths}(M) \to [0,1])$ given by

$$F \sqsubseteq F' \qquad \text{if and only if} \qquad \text{for all } q \in Q, F(q) \subseteq F'(q)$$

Clearly $(\mathrm{CPaths}(M) \to [0,1]), \sqsubseteq)$ is a complete lattice and $\mathcal{T}_{\mathrm{Tr}}$ is monotonic, so by the theorem of Knaster-Tarski it has a greatest fixed point, which coincides with Traces_s.

Following the definition of $\mathcal{T}_{\mathrm{Tr}}$, we now give a coinductive approximation of the equivalence relation induced by Traces_s. Given a TPA $M = (Q, L, \Sigma, \hat{q}, \alpha)$, consider the operator

$$\mathcal{T}_{\mathrm{Treq}} : (\mathrm{CPaths}(M) \to Q \times Q) \to (\mathrm{CPaths}(M) \to Q \times Q)$$

defined as:

$$q_1\ \mathcal{T}_{\mathrm{Treq}}(\mathcal{R})(\epsilon)\ q_2 \quad \overset{\text{def}}{\Leftrightarrow} \quad (q_1 \not\rightarrow \Leftrightarrow\ q_2 \not\rightarrow)$$

and

$$q_1\ \mathcal{T}_{\mathrm{Treq}}(\mathcal{R})(\ell : a \cdot t)\ q_2 \quad \overset{\text{def}}{\Leftrightarrow} \quad \begin{pmatrix} q_1 \xrightarrow{\ell:a} \mu_1 \Rightarrow \exists \mu_2 . (q_2 \xrightarrow{\ell:a} \mu_2\ \wedge\ \mu_1\ \mathcal{R}(t)\ \mu_2) \\ \wedge \\ q_2 \xrightarrow{\ell:a} \mu_2 \Rightarrow \exists \mu_1 . (q_1 \xrightarrow{\ell:a} \mu_1\ \wedge\ \mu_1\ \mathcal{R}(t)\ \mu_2) \end{pmatrix}$$

Consider the ordering $\preceq$ on $\mathrm{CPaths}(M) \to Q \times Q$ given by

$$\mathcal{R} \preceq \mathcal{R}' \qquad \text{if and only if} \qquad \text{for all } t \in \mathrm{CPaths}(M),\ \mathcal{R}(t) \subseteq \mathcal{R}'(t)$$

Clearly $(\mathrm{CPaths}(M) \to Q \times Q, \preceq)$ is a complete lattice and $\mathcal{T}_{\mathrm{Treq}}$ is monotonic, hence by the Knaster-Tarski theorem it has a greatest fixed point, which also coincides with the greatest pre-fixed point, i.e. the greatest relation $\mathcal{R}$ such that $\mathcal{R} \preceq \mathcal{T}_{\mathrm{Treq}}(\mathcal{R})$. Using the definition of $\mathcal{T}_{\mathrm{Tr}}$ it is easy to see that, if $\mathcal{R}$ is a pre-fixed point, and $q_1\ \mathcal{R}(t)\ q_2$ for all $t \in \mathrm{CPaths}(M)$, then $\mathit{Traces}_s(q_1) = \mathit{Traces}_s(q_2)$, i.e. $q_1 \simeq q_2$. In fact, if $F(q_1) = F(q_2)$, and $q_1\ \mathcal{R}(t)\ q_2$ for all $t \in \mathrm{CPaths}(M)$, and $\mathcal{R}$ is a pre-fixed point of $\mathcal{T}_{\mathrm{Treq}}$, then $\mathcal{T}_{\mathrm{Tr}}(F)(q_1) = \mathcal{T}_{\mathrm{Tr}}(F)(q_2)$[3]. Consider now a safe bisimulation $\mathcal{R}$, and let us lift it to a constant function $\mathcal{R} : \mathrm{CPaths}(M) \to Q \times Q$ defined as $\mathcal{R}(t) = \mathcal{R}$. It

[3] Note that the condition is only sufficient, because $\sum_{q'} \mu_1(q') f'_{q'1}(t) = \sum_{q'} \mu_2(q') f'_{q'2}(t)$ may hold even if μ_1 and μ_2 assign some different probability to some equivalence class of $\hat{\mathcal{R}(t)}$.

is easy to see that $\mathcal{R}$ is a pre-fixed point of $\mathcal{T}_{\mathrm{Treq}}$[4]. Assume now $q_1 \ \mathcal{R} \ q_2$. We trivially derive that $q_1 \ \mathcal{R}(t) \ q_2$ for all $t \in \mathrm{CPaths}(M)$, from which we conclude $q_1 \simeq q_2$. □

Theorem 2. *Let S be a system with a variable action $secr$ and assume that $S[^{s_1}/_{secr}] \simeq_s S[^{s_2}/_{secr}]$ for every pair of secrets s_1 and s_2. Then S is leakage-free.*

Proof. Consider the abstraction operator β from safe traces to pairs of the form (tagged observable trace, probability) defined as:

$$(\tilde{a}, p) \in \beta(F) \quad \stackrel{\mathrm{def}}{\Leftrightarrow} \quad p \ = \sum_{\substack{f \in F \\ t_{\restriction L \times \mathcal{O}} = \tilde{a}}} f(t)$$

It is easy to see that β is an abstraction, i.e. if $F_1 = F_2$ then $\beta(F_1) = \beta(F_2)$. Therefore, $S[^{s_1}/_{secr}] \simeq_s S[^{s_2}/_{secr}]$ implies $\beta(\mathit{Traces}_s(S[^{s_1}/_{secr}]) = \beta(\mathit{Traces}_s(S[^{s_2}/_{secr}])$. Finally, the latter holds (for every pair of secrets s_1, s_2) if and only if S is leakage-free. □

Theorem 3. *Let $\simeq'_s$, $\sim'_s$ be the versions of $\simeq_s$, $\sim_s$ respectively, obtained by considering only admissible global schedulers. Let S be a system with a variable action $secr$ and assume that $S[^{s_1}/_{secr}] \simeq'_s S[^{s_2}/_{secr}]$ for every pair of secrets s_1 and s_2. Then S is leakage-free. The same holds if in the above we replace $\simeq'_s$ by $\sim'_s$.*

Proof. The restriction to safe schedulers makes less strict the definitions of traces and bisimulation, hence in general we have $\simeq_s \subseteq \simeq'_s$ and $\sim_s \subseteq \sim'_s$. However their properties are preserved, i.e. Lemma 1, Proposition 1, Theorem 1, Proposition 2, Theorem 2 and Corollary 1 still hold; their proofs are essentially the same. □

[4] Note that the converse does not hold, i.e. $\mathcal{R}$ could be a pre-fixpoint of $\mathcal{T}_{\mathrm{Treq}}$ even if $\mathcal{R}$ is not a bisimulation. This is because $\mathcal{R}$ is sensitive to the (nondeterministic) branching structure, while $\mathcal{R}$ is not.