Security Analysis in Probabilistic Distributed Protocols via Bounded Reachability[⋆]

Silvia S. Pelozo and Pedro R. D'Argenio

CONICET - FaMAF - Universidad Nacional de Córdoba
{spelozo,dargenio}@famaf.unc.edu.ar

Abstract. We present a framework to analyze security properties in distributed protocols. The framework is constructed on top of the so called (strongly) distributed schedulers where secrecy is also considered. Secrecy is presented as an equivalence class on actions to those components that do not have access to such secrets; however these actions can be distinguished by those with appropriate clearance. We also present an algorithm to solve bounded reachability analysis on this kind of models. The algorithm appropriately encodes the nondeterministic model by interpreting the decisions of the schedulers as parameters. The problem is then reduced to a polynomial optimization problem.

1 Introduction

Model-based verification has proven very useful in the verification of a handful of systems. It is particularly fit for the verification of distributed systems, in which the model of the system is obtained by composing simpler models describing the behaviour of each component. A particular class of these systems are distributed algorithms that aim to provide information hiding, i.e., they try to prevent an adversary to infer confidential information from the observables [2]. Many of these algorithms propose a solution by adding randomization to their decisions (e.g. [8,15]). In addition, the security property is best understood quantitatively by contrasting the likelihood of producing a secret w.r.t. the likelihood of guessing such secret after reading the observables (see e.g. [3,16,2,1]). Notice that, due to the nature of distributed systems, the model needs to consider both probabilistic and nondeterministic behaviour: probabilities allow to model randomization (including the likelihood of secrets) while nondeterminism expresses the interleaving of different processes, abstraction of decision mechanisms, and model underspecification among other things.

In this paper, we focus in quantitative reachability properties, i.e. those that assert about the probability of reaching some states of particular interest, be it because it is desirable or undesirable. In particular, we are interested in accurately verifying security properties of distributed system where private actions and interactions between some components are effectively hidden from others. In this setting, we consider a system secure if the probability of reaching states

[⋆] Supported by Project ANPCYT PAE-PICT 02272, SeCyT-UNC, and EU 7FP grant agreement 295261 (MEALS).

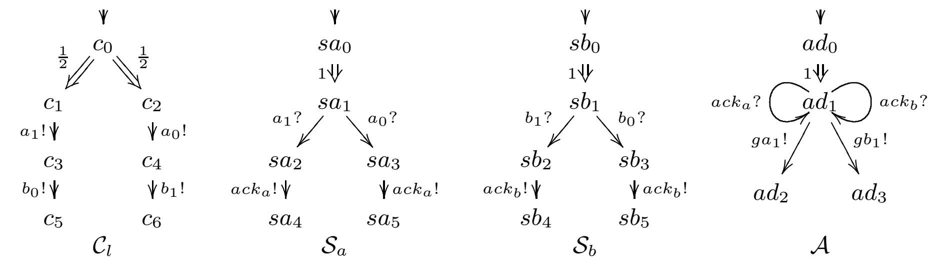

Fig. 1. Output actions are suffixed with "!" and input actions are suffixed with "?". The set of actions of each component are only those depicted in the component. To ensure input enabledness, all states are assumed to have self-loops labeled with the inputs others than those already depicted leaving the state. For every final state s we also omitted the transition $s \Rightarrow \mu$ with $\mu(s) = 1$.

where these properties are violated is between certain known bounds. For example, in an anonymous message exchanging protocol, we would like to verify that the probability of guessing the anonymous sender of a message is not greater than the probability of guessing the sender by chance alone.

Probabilistic model checking is –in principle– adequate to achieve this. However, traditional analysis techniques do not necessarily provide results reflecting the security guarantees of the modeled systems: they only provide pessimistic over-approximations for the actual probability of the property. This is due to the fact that traditional techniques consider *all* possible resolutions of nondeterminism [4]. To understand the problem we present an example that we will use along the paper. Consider a protocol where a client $\mathcal{C}_l$ chooses randomly one out of two available service providers $\mathcal{S}_a$ and $\mathcal{S}_b$ (see Fig. 1). This component asks for service with an encrypted message (a_1 or b_1, depending on the chosen provider). To misguide possible attackers it also interchanges a dummy (encrypted) message with the other provider (a_0 or b_0). Once the provider receives any of the notifications it acknowledges reception of the message. If the encryption scheme is secure, the selection and its outcome should be known only by the client and the providers. Any other component of the system (in particular, the adversary $\mathcal{A}$) should not be able to infer it with certainty.

Resolution of nondeterminism is done by the so called *schedulers* which are functions that select the next step based on the past execution of the system. Traditional probabilistic model checking will consider the scheduler that, whenever enabled, selects action ga_1 (for "guess $\mathcal{S}_a$ got 1") if action a_1 appears in the execution, and it selects gb_1 ("guess $\mathcal{S}_b$ got 1") if b_1 appears in the execution. This is a valid scheduler that lets the adversary $\mathcal{A}$ guess with probability 1.

One of the reasons this happens is because the resolution of the local nondeterminism in $\mathcal{A}$ is made using global knowledge. This problem has been recently observed by several authors [9,13,14,11] who proposed to restrict to the so called *distributed schedulers*. Distributed schedulers enforce that *local* decisions of the processes are based only on local knowledge [14]. However, nondeterminism originated by interleaving in parallel composition is still resolved using global knowledge in this new framework. Since distributed schedulers were not defined in a context were secrecy was important, actions a_1 and b_1 are considered different from a_0 and b_0. As they are part of the global knowledge, the resolution of the interleaving nondeterminism can be different in each case. This may lead to some unrealistic leakage of information. We present an example in detail in Sec. 3.

In this paper we adapt distributed schedulers to deal also with secrecy, presented as an equivalence class on local states and actions. Therefore, two actions in the same equivalence class can only be distinguished locally in components with the appropriate clearance but cannot be distinguished globally.

In general, reachability under distributed schedulers is an undecidable problem [13]. However, if restricted to reachability properties where the goal states should be reached within a given number of steps (i.e. *(time-)bounded* reachability properties), the problem becomes decidable [5]. This is done by reducing the bounded reachability problem to a polynomial optimization problem. In this paper we adapt and improve the algorithm of [5] to work on the class of distributed schedulers under secrecy. As an example of the application to information hiding, we automatically verify the anonymity property in a case study.

2 Modeling Probabilistic Distributed System

To assert or refute a quantitative reachability property about a distributed system, we first construct a model of each component of the system. The full model is constructed by composing these submodels, considering possible interactions between the components and all possible interleaving. Each component is described with a state-based model that combines discrete-time Markov chains and labeled transition systems. In this work we use a restricted variant of interactive probabilistic chains (IPCs) [10,5], a formalism where probabilistic transitions and action-labelled transitions are handled orthogonally.

Definition 1. *A* basic I/O-IPC *is a tuple $\langle S, A, \rightarrow, \Rightarrow, \hat{s}\rangle$ where S is a finite set of states with initial state $\hat{s} \in S$, $A = A^I \cup A^O$ is a finite set of actions consisting of disjoint sets of input actions (A^I) and output actions (A^O), $\rightarrow \subseteq S \times A \times S$ is the set of interactive transitions, and $\Rightarrow : S \rightharpoonup \mathit{Dist}(S)$ is a partial function representing probabilistic transitions. We require that the I/O-IPC is (i)* input enabled*, i.e. for all $s \in S$ and $a \in A^I$, there exists $s' \in S$ s.t. $s \xrightarrow{a} s'$; and (ii)* action deterministic*, i.e. for all $s \in S$ and $a \in A$, $s \xrightarrow{a} s'$ and $s \xrightarrow{a} s''$ implies $s' = s''$.*

We use the shorthand notation $s \xrightarrow{a} s'$ for an interactive transition $(s, a, s') \in \rightarrow$, $s \Rightarrow \mu$ for $(s, \mu) \in \Rightarrow$ and $s \not\rightarrow$ if there is no $a \in A^O$ and s' such that $s \xrightarrow{a} s'$.

The requirement of input action determinism is present in this work to simplify presentation. The general model can be treated just like in [14].

Parallel Composition. Distributed I/O-IPC are obtained by composing basic ones through the parallel composition. We define it simultaneously on a finite set of basic I/O-IPC. To avoid unnecessary technicalities, we require *output isolation*, i.e. no output action can be performed by more than one basic process.

Similarly to action determinism, the framework may be extended to models without output isolation (see [14,12]).

Definition 2. *A finite set of basic I/O-IPCs* $\mathcal{P}_i = \langle S_{\mathcal{P}_i}, A_{\mathcal{P}_i}, \rightarrow_{\mathcal{P}_i}, \Rightarrow_{\mathcal{P}_i}, \hat{s}_{\mathcal{P}_i} \rangle$, $1 \leq i \leq n$, *is* composable *if* $A^O_{\mathcal{P}_i} \cap A^O_{\mathcal{P}_j} = \emptyset, \forall i \neq j$. *Provided that* $\{\mathcal{P}_i\}_{i=1}^n$ *is composable, the* parallel composition $\mathcal{C} = \mathcal{P}_1 \parallel \cdots \parallel \mathcal{P}_n$ *is defined by the I/O-IPC* $\langle S_\mathcal{C}, A^I_\mathcal{C} \cup A^O_\mathcal{C}, \rightarrow_\mathcal{C}, \Rightarrow_\mathcal{C}, \hat{s}_\mathcal{C} \rangle$*, where* $S_\mathcal{C} = S_{\mathcal{P}_1} \times \cdots \times S_{\mathcal{P}_n}$ *and* $\hat{s}_\mathcal{C} = (\hat{s}_{\mathcal{P}_1}, \ldots, \hat{s}_{\mathcal{P}_n})$ *is the initial state;* $A^O_\mathcal{C} := \bigcup_{i=1}^n A^O_{\mathcal{P}_i}$, $A^I_\mathcal{C} := \bigcup_{i=1}^n A^I_{\mathcal{P}_i} \setminus A^O_\mathcal{C}$ *and the transition relations are defined according to the following rules:*

$$\frac{\{s_i \xrightarrow{a}_{\mathcal{P}_i} s'_i \mid a \in A_{\mathcal{P}_i}\} \qquad t_i = \textbf{ if } (a \in A_{\mathcal{P}_i}) \textbf{ then } s'_i \textbf{ else } s_i}{(s_1, \ldots, s_n) \xrightarrow{a}_\mathcal{C} (t_1, \ldots, t_n)} \tag{1}$$

$$\frac{\{s_i \Rightarrow_{\mathcal{P}_i} \mu_i \ \wedge \ s_i \not\rightarrow \mid 1 \leq i \leq n\}}{(s_1, \ldots, s_n) \Rightarrow_\mathcal{C} \mu_1 \times \cdots \times \mu_n} \tag{2}$$

with $\mu_1 \times \cdots \times \mu_n$ *denoting the product distribution on* $S_{\mathcal{P}_1} \times \cdots \times S_{\mathcal{P}_n}$ *defined by* $(\mu_1 \times \cdots \times \mu_n)(s_1, \ldots, s_n) = \prod_{i=1}^n \mu_i(s_i)$ *for all* $s_i \in S_{\mathcal{P}_i}$.

Notice that composability guarantees that at most one processes performs an output action when synchronizing according to rule (1). Moreover, every other process that *knows* the action will perform the step because of input enabledness.

We consider that output transitions are immediate and probabilistic transitions are timed. We assume that immediate transitions always take precedence over timed transitions. This assumption is known as *maximal progress* [17] and it is reflected on rule (2). Following this criteria, a state is called *vanishing* if at least one output-labeled transition is enabled on it. If only probabilistic or input-labeled transitions are enabled in a state then it is called *tangible.* The introduction of maximal progress may induce an infinite sequence of consecutive output-labeled transitions. We consider this as *Zeno behaviour* and require that composed models *do not* exhibit this type of behaviour. We will also assume that composed systems are *closed*, that is, its set of input actions is empty. Dealing with open systems in our context demands some assumption on the environment behaviour. In any case, such assumption (even the most general) can be encoded in one or more extra components such that the whole system is finally closed.

Resolving Nondeterminism through Schedulers. To obtain the probability of reaching some states, nondeterministic choices between enabled transitions have to be resolved. This is achieved by the so called *schedulers* (also adversaries or policies). A scheduler is a function mapping each partial execution (or finite path) to a distribution on actions enabled in the last state of the execution.

Let $\mathcal{P} = \langle S, A = A^I \cup A^O, \rightarrow, \Rightarrow, \hat{s} \rangle$ be an I/O-IPC. We define $\mathit{enab}(s) = \{a \in A^O \mid \exists s'. s \xrightarrow{a} s'\}$, the set of output actions enabled in s.

A *finite path* of $\mathcal{P}$ is a sequence $\sigma = s_0 a_0 s_1 a_1 \cdots a_{n-1} s_n$ with $s_0 = \hat{s}$, where states and either actions or distributions alternate, that is, for $i = 0, \ldots, n-1$: (i) $a_i \in \mathit{enab}(s_i)$ and $s_i \xrightarrow{a_i} s_{i+1}$, or (ii) $a_i \in \mathit{Dist}(S)$, $s_i \Rightarrow a_i$, $a_i(s_{i+1}) > 0$, and $\mathit{enab}(s_i) = \emptyset$. The last state s_n is denoted by $\mathit{last}(\sigma)$, and $\mathit{length}(\sigma) = n$ is the total number of transitions (interactive or probabilistic) along the path.

Moreover, we denote $\mathit{enab}\,(\mathit{last}\,(\sigma))$ as $\mathit{enab}\,(\sigma)$ for finite path σ. An *infinite path* of $\mathcal{P}$ is an infinite sequence $s_0a_0s_1a_1\cdots$ alternating states and actions or distributions satisfying the previous condition. We denote with $\mathit{Paths}\,(\mathcal{P})$ and $\mathit{Paths}_{\mathit{fin}}\,(\mathcal{P})$ the set of paths and finite paths of $\mathcal{P}$, respectively.

Definition 3. *A* scheduler *for $\mathcal{P}$ is a function $\eta_{\mathcal{P}} : \mathit{Paths}_{\mathit{fin}}\,(\mathcal{P}) \to \mathit{Dist}\,(A)$ such that $\eta_{\mathcal{P}}(\sigma)(\mathit{enab}(\sigma)) = 1$ (that is $\eta_{\mathcal{P}}(\sigma)(a) > 0$ implies $a \in \mathit{enab}(\sigma)$).*

Probability Measure Induced by a Scheduler. When all the nondeterministic choices in an I/O-IPC are resolved by a scheduler the resulting system is a (possibly infinite) Markov chain. Hence it is possible to define a probability measure over the sets of infinite paths of the model.

For the sake of simplicity we will assume that the I/O-IPC does not contain tangible states which do not have an outgoing probabilistic transition. That is, for every tangible state s, $\Rightarrow(s)$ is defined. If this is not the case, we just complete it by extending $\Rightarrow$ with the tuple (s, δ_s) where $\delta_s(s) = 1$.

We first define the σ-algebra on the set of infinite paths of the I/O-IPC and then the probability measure on this σ-algebra induced by a given scheduler. The *cylinder* induced by the finite path σ is the set of infinite paths $\sigma^{\uparrow} = \{\sigma' \in \mathit{Paths}\,(\mathcal{P}) \mid \sigma' \text{ is infinite and } \sigma \text{ is a prefix of } \sigma'\}$. Define $\mathcal{F}$ to be the σ-algebra on the set infinite paths of $\mathcal{P}$ generated by the set of cylinders.

Definition 4. *Let η be a scheduler for $\mathcal{P}$. The* probability measure induced by η *on $\mathcal{F}$ is the unique probability measure $\Pr_{\eta}$ such that, for any state $s \in S$, any action $a \in A$ and any distribution $\mu \in \mathit{Dist}(S)$:*

$$\begin{array}{lll}
\Pr_{\eta}(s^{\uparrow}) & = 1 & \text{if } s = \hat{s} \\
\Pr_{\eta}(\sigma a s^{\uparrow}) & = \Pr_{\eta}(\sigma^{\uparrow}) \cdot \eta(\sigma)(a) & \text{if } \mathit{enab}(\sigma) \neq \emptyset \text{ and } \mathit{last}(\sigma) \xrightarrow{a} s \\
\Pr_{\eta}(\sigma \mu s^{\uparrow}) & = \Pr_{\eta}(\sigma^{\uparrow}) \cdot \mu(s) & \text{if } \mathit{enab}(\sigma) = \emptyset \text{ and } \mathit{last}(\sigma) \Rightarrow \mu \\
\Pr_{\eta}(\sigma^{\uparrow}) & = 0 & \text{in any other case}
\end{array}$$

By the assumption that tangible states are in the domain of $\Rightarrow$, $\mathit{enab}\,(\sigma) = \emptyset$ implies that $\mathit{last}\,(\sigma) \Rightarrow \mu$ for some μ. Hence, $\Pr_{\eta}$ is indeed a probability measure.

Time-bounded reachability properties demand that the system reaches a goal state from a given set G within a given time t. In our case, the notion of time is given by each probability step. So, for any finite path σ, we let $\mathit{time}(\sigma)$ be the number of probability steps appearing on σ. Then, given a scheduler η for $\mathcal{P}$, the probability of reaching a goal state in G within time t, can be computed by $\Pr_{\eta}(\Diamond^{\leq t} G) = \Pr_{\eta}(\bigcup\{\sigma^{\uparrow} \mid \mathit{time}(\sigma) \leq t \wedge \mathit{last}(\sigma) \in G\})$. Let $\bar{G}$ be the set of paths $\sigma \in \mathit{Paths}_{\mathit{fin}}\,(\mathcal{P})$ such that $\mathit{last}(\sigma) \in G$ and for any proper prefix $\hat{\sigma}$ of σ, $\mathit{last}(\hat{\sigma}) \notin G$. Notice that for every $\sigma' \in \mathit{Paths}\,(\mathcal{P})$ reaching a state in G, there exists a unique $\sigma \in \bar{G}$ such that $\sigma' \in \sigma^{\uparrow}$. Then, we have that

$$\Pr_{\eta}\left(\Diamond^{\leq t} G\right) = \Pr_{\eta}\left(\biguplus\left\{\sigma^{\uparrow} \mid \mathit{time}(\sigma) \leq t \wedge \sigma \in \bar{G}\right\}\right) = \sum_{\substack{\sigma \in \bar{G} \\ \mathit{time}(\sigma) \leq t}} \Pr_{\eta}(\sigma^{\uparrow}). \quad (3)$$

The model checking problem on nondeterministic probabilistic systems is focused on finding worst case scenarios. Therefore, it aims to find the maximum or minimum probability of reaching a set of goal states ranging over a particular class of schedulers. That is, if $\mathcal{K}$ is a class of schedulers (i.e. a set of all schedulers satisfying some given condition), then we are interested on finding $\sup_{\eta \in \mathcal{K}} \Pr_{\eta}(\Diamond^{\leq t} G)$ or $\inf_{\eta \in \mathcal{K}} \Pr_{\eta}(\Diamond^{\leq t} G)$.

Distributed Schedulers. Not all resolutions of nondeterminism are appropriate in a distributed setting. There are "almighty schedulers" that allow a component to guess the outcome of a probabilistic choice of a second component even when they have no communication at all (not even indirectly). The reader is referred to, e.g., [14,12] for a discussion. Therefore, we restrict to the class of *distributed schedulers* [14]. This leads to a more realistic resolution of nondeterminism, but unfortunately it also renders the model checking problem undecidable in general [13,12].

Distributed schedulers consider a notion of local knowledge of each component which is obtained by partially observing the global system behaviour. Thus, a component can only see the global execution through its local states and the actions it performs. Hence, two different global execution may appear the same to a single component. We implement this with a *projection function.*

From now on we will consider a composed model $\mathcal{C} = \mathcal{P}_1 \parallel \cdots \parallel \mathcal{P}_n = \langle S_{\mathcal{C}}, A_{\mathcal{C}}, \rightarrow_{\mathcal{C}}, \Rightarrow_{\mathcal{C}}, \hat{s}_{\mathcal{C}} \rangle$, where each $\mathcal{P}_i = \langle S_i, A_i = A_i^I \cup A_i^O, \rightarrow_i, \Rightarrow_i, \hat{s}_i \rangle$. For every path $\sigma \in \mathit{Paths}\,(\mathcal{C})$, the *projection* $\sigma\,[\mathcal{P}_i] \in \mathit{Paths}\,(\mathcal{P}_i)$ *of* σ *over* $\mathcal{P}_i$ is defined as:

$$\begin{aligned}
(s_1, \ldots, s_n)\,[\mathcal{P}_i] &= s_i \\
\sigma a(s_1, \ldots, s_n)\,[\mathcal{P}_i] &= \textbf{ if } a \in A_i \textbf{ then } (\sigma\,[\mathcal{P}_i]) a s_i \textbf{ else } \sigma\,[\mathcal{P}_i] \\
\sigma(\mu_1 \times \cdots \times \mu_n)(s_1, \ldots, s_n)\,[\mathcal{P}_i] &= (\sigma\,[\mathcal{P}_i]) \mu_i s_i
\end{aligned}$$

In a (composed) state with enabled interactive transitions, the nondeterminism is resolved in two phases. First, a component is chosen among all enabled components. This choice may be probabilistic, and it is performed by the so called *interleaving scheduler*. Afterwards, the chosen component decides which transition to perform among all its enabled output transitions. This local choice is resolved by a *local scheduler* taking into account only local knowledge. Hence they are functions on *local* executions which are obtained by properly projecting the global executions. Therefore, a local scheduler is a scheduler as defined in Def. 3, only that its domain is the set of all finite paths of the local component. An interleaving scheduler is defined on finite paths of the composed system (hence, *global* executions) as follows.

Definition 5. *A function* $\mathcal{I} : \mathit{Paths}_{\mathit{fin}}\,(\mathcal{C}) \rightarrow \mathit{Dist}\,(\{\mathcal{P}_1, \ldots, \mathcal{P}_n\})$ *is an* interleaving scheduler *if for all* $\sigma \in \mathit{Paths}_{\mathit{fin}}\,(\mathcal{C})$, $\mathcal{I}\,(\sigma)\,(\mathcal{P}_i) > 0 \;\Rightarrow\; \mathit{enab}\,(\sigma\,[\mathcal{P}_i]) \neq \emptyset$.

A *distributed scheduler* is obtained by properly composing the interleaving schedulers with all local schedulers.

Definition 6. *A function* $\eta_{\mathcal{C}} : \mathit{Paths}_{\mathit{fin}}\,(\mathcal{C}) \rightarrow \mathit{Dist}(A_{\mathcal{C}})$ *is a* distributed scheduler *if there are local schedulers* $\eta_{\mathcal{P}_1}, \ldots, \eta_{\mathcal{P}_n}$ *and an interleaving scheduler* $\mathcal{I}$*, such that, for all* $\sigma \in \mathit{Paths}_{\mathit{fin}}\,(\mathcal{C})$ *with* $\mathit{enab}(\sigma) \neq \emptyset$ *and for all* $a \in A_{\mathcal{C}}$*:* $\eta_{\mathcal{C}}(\sigma)(a) = \sum_{i=1}^{n} \mathcal{I}(\sigma)(\mathcal{P}_i) \cdot \eta_{\mathcal{P}_i}(\sigma\,[\mathcal{P}_i])(a)$.

Since at most one component can output action a, last equation reduces to: $\eta_{\mathcal{C}}(\sigma)(a) = \mathcal{I}(\sigma)(\mathcal{P}_i) \cdot \eta_{\mathcal{P}_i}(\sigma\,[\mathcal{P}_i])(a)$ whenever $a \in A_{\mathcal{P}_i}^O$.

It has been shown that distributed scheduler as defined above still permeates information from one component to others with no apparent reason (see [14,12]). In essence, the problem is that the interleaving scheduler may use information from a component $\mathcal{P}_1$ to decide how to pick between components $\mathcal{P}_2$ and $\mathcal{P}_3$. Therefore, we need to restrict the set of possible interleaving schedulers. We will require that, if neither components $\mathcal{P}_2$ and $\mathcal{P}_3$ can distinguish execution σ from σ', the interleaving scheduler has to consider the same relative (i.e. conditional) probabilities for choosing $\mathcal{P}_2$ or $\mathcal{P}_3$ after both paths.

Definition 7. *A scheduler η of $\mathcal{C}$ is said to be* strongly distributed *if it is distributed and for every two components $\mathcal{P}_i, \mathcal{P}_j$, and $\sigma, \sigma' \in \mathit{Paths}_{\mathit{fin}}(\mathcal{C})$ the interleaving scheduler $\mathcal{I}$ of η satisfies that, whenever* (i) $\sigma[\mathcal{P}_i] = \sigma[\mathcal{P}_j]$ *and* $\sigma'[\mathcal{P}_i] = \sigma'[\mathcal{P}_j]$, *and* (ii) $\mathcal{I}(\sigma)(\mathcal{P}_i) + \mathcal{I}(\sigma)(\mathcal{P}_j) \neq 0$ *and* $\mathcal{I}(\sigma')(\mathcal{P}_i) + \mathcal{I}(\sigma')(\mathcal{P}_j) \neq 0$, *it holds that* $\frac{\mathcal{I}(\sigma)(\mathcal{P}_i)}{\mathcal{I}(\sigma)(\mathcal{P}_i)+\mathcal{I}(\sigma)(\mathcal{P}_j)} = \frac{\mathcal{I}(\sigma')(\mathcal{P}_i)}{\mathcal{I}(\sigma')(\mathcal{P}_i)+\mathcal{I}(\sigma')(\mathcal{P}_j)}$.

The previous restriction generalizes to $\frac{\mathcal{I}(\sigma)(\mathcal{P}_j)}{\sum_{i\in I}\mathcal{I}(\sigma)(\mathcal{P}_i)} = \frac{\mathcal{I}(\sigma')(\mathcal{P}_j)}{\sum_{i\in I}\mathcal{I}(\sigma')(\mathcal{P}_i)}$, where $I \subseteq \{1,\ldots,n\}$ and $j \in I$ [14, Theorem 3.4].

3 Distributed Schedulers in Systems with Secrecy

Recall the example of Fig. 1 and consider the following paths:

$$\sigma_a = \langle c_0, sa_0, sb_0, ad_0\rangle\, \mu\, \langle c_1, sa_1, sb_1, ad_1\rangle\, a_1\, \langle c_3, sa_2, sb_1, ad_1\rangle\, b_0\, \langle c_5, sa_2, sb_3, ad_1\rangle$$
$$\sigma_b = \langle c_0, sa_0, sb_0, ad_0\rangle\, \mu\, \langle c_2, sa_1, sb_1, ad_1\rangle\, a_0\, \langle c_4, sa_3, sb_1, ad_1\rangle\, b_1\, \langle c_6, sa_3, sb_2, ad_1\rangle$$
$$\sigma'_a = \sigma_a\, ack_a\, \langle c_5, sa_4, sb_3, ad_1\rangle \qquad \sigma'_b = \sigma_b\, ack_b\, \langle c_6, sa_3, sb_4, ad_1\rangle$$

with $\mu(\langle c_1, sa_1, sb_1, ad_1\rangle) = \mu(\langle c_2, sa_1, sb_1, ad_1\rangle) = \frac{1}{2}$.

σ_a and σ_b are the only two possible paths of the system in which $\mathcal{C}_l$ executes all its outputs before any other component. σ'_a (resp., σ'_b) is the path in which server $\mathcal{S}_a$ (resp., $\mathcal{S}_b$) acknowledge reception after both servers were contacted by the client $\mathcal{C}_l$. Define the interleaving scheduler $\mathcal{I}$ by $\mathcal{I}(\sigma_a)(\mathcal{S}_a) = \mathcal{I}(\sigma_b)(\mathcal{S}_b) = \mathcal{I}(\sigma'_a)(\mathcal{A}) = \mathcal{I}(\sigma'_b)(\mathcal{A}) = 1$, and $\mathcal{I}(\sigma)(\mathcal{C}_l) = 1$ if σ is a prefix of σ_a or σ_b. The definition of $\mathcal{I}$ in any other case is not relevant as long as it satisfies the condition of Def. 7. Then $\mathcal{I}$ satisfies in general the condition of Def. 7. (Notice, in particular, that $\sigma_a[\mathcal{S}_a] \neq \sigma_b[\mathcal{S}_a]$ and $\sigma_a[\mathcal{S}_b] \neq \sigma_b[\mathcal{S}_b]$.)

Define the local scheduler for the adversary $\mathcal{A}$ by $\eta_{\mathcal{A}}(\sigma\, ack_a\, ad_1)(ga_1) = \eta_{\mathcal{A}}(\sigma\, ack_b\, ad_1)(gb_1) = 1$. I.e., the scheduler chooses with probability 1 to guess that server $\mathcal{S}_a$ has being selected (action ga_1) if the last acknowledgement it observes comes from $\mathcal{S}_a$ (action ack_a). Instead, if $\mathcal{S}_b$ is the last to acknowledge, the scheduler choses to guess $\mathcal{S}_b$. Note that all other local schedulers are trivial.

Let η be the strongly distributed scheduler obtained by properly combining the previous interleaving and local schedulers. It is not difficult to verify that

$$\begin{aligned}
&\Pr{}_\eta\left((\sigma'_a\, ga_1\, \langle c_5, sa_4, sb_3, ad_2\rangle)^{\uparrow} \cup (\sigma'_b\, gb_1\, \langle c_6, sa_3, sb_4, ad_3\rangle)^{\uparrow}\right) = \\
&\quad = \Pr{}_\eta({\sigma'_a}^{\uparrow}) \cdot \mathcal{I}(\sigma'_a)(\mathcal{A}) \cdot \eta_{\mathcal{A}}(\sigma'_a[\mathcal{A}])(ga_1) + \Pr{}_\eta({\sigma'_b}^{\uparrow}) \cdot \mathcal{I}(\sigma'_b)(\mathcal{A}) \cdot \eta_{\mathcal{A}}(\sigma'_b[\mathcal{A}])(gb_1) \\
&\quad = \Pr{}_\eta({\sigma'_a}^{\uparrow}) + \Pr{}_\eta({\sigma'_b}^{\uparrow}) = \tfrac{1}{2} + \tfrac{1}{2} = 1.
\end{aligned}$$

Notice that the adversary $\mathcal{A}$ guesses right whenever the system reaches a state of the form $\langle c_5, *, *, ad_2\rangle$ or $\langle c_6, *, *, ad_3\rangle$. Therefore, the previous calculation states that the adversary guesses the server chosen by the client with probability 1.

It is nonetheless clear that there is no apparent reason for the adversary $\mathcal{A}$ to guess right all the time. In fact, scheduler $\eta_{\mathcal{A}}$ is defined in a pretty arbitrary manner. The correct guessing is actually a consequence of the interleaving scheduler $\mathcal{I}$ that let the chosen server acknowledge first and immediately after it passes the control to the adversary $\mathcal{A}$ (hence making $\mathcal{A}$ guess through the arbitrary scheduler $\eta_{\mathcal{A}}$).

Observe that the interleaving scheduler of the previous example decides which is the next enabled component based on the outcome of a secret. However, a secret should not be directly observed by the environment. Hence, the interleaving should not be able to distinguish action a_1 from a_0 and neither b_1 from b_0. Similarly, internal states of a component that only differ on the value of confidential information should not influence the decision of the interleaving scheduler.

Therefore the notion of a valid interleaving scheduler of Def. 7 needs to be strengthen. In the new definition, the interleaving scheduler has to consider the same relative (i.e. conditional) probabilities for two components if both of them show the same behaviour to the environment. By "showing the same behaviour" we mean the projected traces are the same *after hiding* secret information.

The way in which we have chosen to hide information is through an equivalence relation. Thus, two actions that are equivalent share a secret and should not be distinguished by the environment, and similarly for states. This idea of indistinguishability is local to each component. So for each component $\mathcal{P}_i$ we consider an equivalence relation $\sim_{A_i}$ on actions and another equivalence relation $\sim_{S_i}$ on states. In our example, we need to define equivalence relations for $\mathcal{C}_l$, $\mathcal{S}_a$, and $\mathcal{S}_b$ such that:

$$\begin{array}{lllll}
a_1 \sim_{A_{\mathcal{C}_l}} a_0 & a_1 \sim_{A_{\mathcal{S}_a}} a_0 & c_1 \sim_{S_{\mathcal{C}_l}} c_2 & sa_2 \sim_{S_{\mathcal{S}_a}} sa_3 & sb_2 \sim_{S_{\mathcal{S}_b}} sb_3 \\
b_1 \sim_{A_{\mathcal{C}_l}} b_0 & & c_3 \sim_{S_{\mathcal{C}_l}} c_4 & sa_4 \sim_{S_{\mathcal{S}_a}} sa_5 & sb_4 \sim_{S_{\mathcal{S}_b}} sb_5 \\
 & b_1 \sim_{A_{\mathcal{S}_b}} b_0 & c_5 \sim_{S_{\mathcal{C}_l}} c_6 & &
\end{array}$$

This relations can be lifted to an equivalence relation on paths as expected: if $\sim_{S_i} \subseteq S_i \times S_i$ and $\sim_{A_i} \subseteq A_i \times A_i$ are equivalence relations, we define $\sim_{\mathcal{P}_i} \subseteq$ *Paths* $(\mathcal{P}_i) \times$ *Paths* $(\mathcal{P}_i)$ recursively by

$$\begin{array}{rcl}
s \sim_{\mathcal{P}_i} s' & \Leftrightarrow & s \sim_{S_i} s' \\
\sigma a s \sim_{\mathcal{P}_i} \sigma' a' s' & \Leftrightarrow & \sigma \sim_{\mathcal{P}_i} \sigma' \wedge a \sim_{A_i} a' \wedge s \sim_{S_i} s' \\
\sigma \mu s \sim_{\mathcal{P}_i} \sigma' \mu' s' & \Leftrightarrow & \sigma \sim_{\mathcal{P}_i} \sigma' \wedge s \sim_{S_i} s'
\end{array}$$

Notice that in our example, $\sigma_a\,[\mathcal{S}_a] \sim_{\mathcal{S}_a} \sigma_b\,[\mathcal{S}_a]$, $\sigma_a\,[\mathcal{S}_b] \sim_{\mathcal{S}_b} \sigma_b\,[\mathcal{S}_b]$, and $\sigma_a\,[\mathcal{C}_l] \sim_{\mathcal{C}_l} \sigma_b\,[\mathcal{C}_l]$. Therefore, under some secrecy assumptions, the environment is not able to distinguish the local execution of $\mathcal{S}_a$ under σ_a from the local execution under σ_b (and similarly $\mathcal{S}_b$ and $\mathcal{C}_l$). Hence, the interleaving scheduler should not make a difference on the relative probabilities of choosing $\mathcal{S}_a$ or $\mathcal{S}_b$. Therefore, we define distributed scheduler under secrecy as follows.

Definition 8. *Let $\mathcal{C} = \mathcal{P}_1 \parallel \cdots \parallel \mathcal{P}_n$ and let $\sim_{S_i} \subseteq S_i \times S_i, \sim_{A_i} \subseteq A_i \times A_i$, with $i = 1, \ldots, n$, be equivalence relations. A scheduler η of $\mathcal{C}$ is a* distributed scheduler under secrecy *if it is distributed and, for every* $I \subseteq \{1, \dots, n\}$*, and* $\sigma, \sigma' \in \mathit{Paths}_{\mathit{fin}}(\mathcal{C})$*, the interleaving scheduler* $\mathcal{I}$ *of* η *satisfies that, whenever*

(i) $\sigma[\mathcal{P}_i] \sim_{\mathcal{P}_i} \sigma'[\mathcal{P}_i]$ *for all* $i \in I$*,*
(ii) $\sum_{i \in I} \mathcal{I}(\sigma)(\mathcal{P}_i) \neq 0$ *and* $\sum_{i \in I} \mathcal{I}(\sigma')(\mathcal{P}_i) \neq 0$*, and*
(iii) $\mathit{enab}(\sigma[\mathcal{P}_i]) \neq \emptyset \Leftrightarrow \mathit{enab}(\sigma'[\mathcal{P}_i]) \neq \emptyset$*, for all* $i \in I$*,*

it holds that, for every $j \in I$*,* $\frac{\mathcal{I}(\sigma)(\mathcal{P}_j)}{\sum_{i \in I} \mathcal{I}(\sigma)(\mathcal{P}_i)} = \frac{\mathcal{I}(\sigma')(\mathcal{P}_j)}{\sum_{i \in I} \mathcal{I}(\sigma')(\mathcal{P}_i)}$*. We let* $\mathcal{DSS}$ *denote the set of all* distributed scheduler under secrecy*.*

Just like for the case of strongly distributed schedulers, it suffices to consider a pair of components for the restriction of $\mathcal{I}$ (i.e., sets I s.t. $\#I = 2$). We choose instead this more general definition because it plays an important role later.

Conditions (i) and (ii) already appear in the definition of strongly distributed schedulers (see Def. 7) only that (i) is consider under equality rather than the secrecy equivalences $\sim_{\mathcal{P}_i}$. Condition (iii) is new here and not needed in Def. 7. This has to do with the fact that, though $\sigma[\mathcal{P}_i]$ and $\sigma'[\mathcal{P}_i]$ may appear the same to the environment due to hidden secrets, they may be different executions of $\mathcal{P}_i$ and hence enable different sets of output actions. This is not the case for Def. 7 in which item (i) requires that $\sigma[\mathcal{P}_i] = \sigma'[\mathcal{P}_i]$. In fact, strongly distributed schedulers are a particular case of distributed schedulers under secrecy in which there is no secret (i.e. $\sim_{A_i}$ and $\sim_{S_i}$ are the identity relation).

Returning to our running example, notice that the interleaving scheduler defined at the beginning of this section is *not* a valid interleaving scheduler for a distributed scheduler under secrecy. In effect, notice that (i) $\sigma_a[\mathcal{S}_a] \sim_{\mathcal{S}_a} \sigma_b[\mathcal{S}_a]$ and $\sigma_a[\mathcal{S}_b] \sim_{\mathcal{S}_b} \sigma_b[\mathcal{S}_b]$, (ii) $\mathcal{I}(\sigma_a)(\mathcal{S}_a) + \mathcal{I}(\sigma_a)(\mathcal{S}_b) = 1$ and $\mathcal{I}(\sigma_b)(\mathcal{S}_a) + \mathcal{I}(\sigma_b)(\mathcal{S}_b) = 1$ (since $\mathcal{I}(\sigma_a)(\mathcal{S}_a) = \mathcal{I}(\sigma_b)(\mathcal{S}_b) = 1$), and (iii) $\mathit{enab}(\sigma_a[\mathcal{S}_a]) = \mathit{enab}(\sigma_b[\mathcal{S}_a]) = \{\mathit{ack}_a\}$ and $\mathit{enab}(\sigma_a[\mathcal{S}_b]) = \mathit{enab}(\sigma_b[\mathcal{S}_b]) = \{\mathit{ack}_b\}$; hence conditions (i), (ii), and (iii) from Def. 8 hold. However, $\frac{\mathcal{I}(\sigma_a)(\mathcal{S}_a)}{\mathcal{I}(\sigma_a)(\mathcal{S}_a) + \mathcal{I}(\sigma_a)(\mathcal{S}_b)} = 1 \neq 0 = \frac{\mathcal{I}(\sigma_b)(\mathcal{S}_a)}{\mathcal{I}(\sigma_b)(\mathcal{S}_a) + \mathcal{I}(\sigma_b)(\mathcal{S}_b)}$, contradicting the requirement on $\mathcal{I}$ in Def. 8.

4 Parametric Characterization

A scheduler resolves all nondeterministic choices of an I/O-IPC thorough probabilistic choices. Therefore, it defines an (infinite) Markov chain which is a particular instance of the original I/O-IPC where all nondeterminism has been replaced by a probabilistic transition.

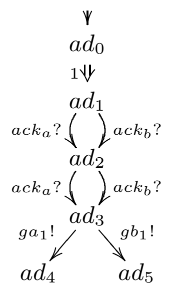

Fig. 2. $\mathcal{A}$

To illustrate this, consider our example of Fig. 1 with component $\mathcal{A}$ replaced by the one in Fig. 2 (which we adopt for simplicity). The resolution through a scheduler of the composed system would look very much like the tree of Fig. 3, except that variables x_i should be omitted and variables y_i should be interpreted as probability values in the interval $[0, 1]$ properly defining a probabilistic distribution (e.g., $y_1 = 1$ and $y_3 + y_4 = 1$). However, if variables y_i are not interpreted we

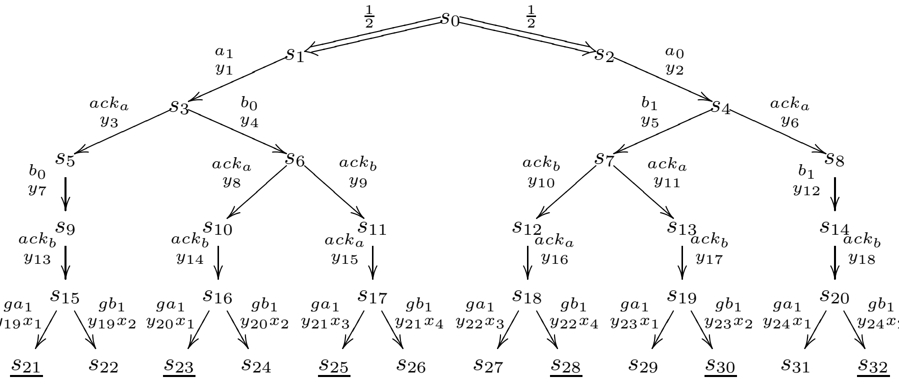

Fig. 3. Parametric scheduler η. States are the obvious tuples. Underlined states are guessing states.

could think of such parametric tree as a symbolic representation of *any possible* scheduler. (Keep ignoring x_i's by the time being.)

The probability of path $\sigma = s_0\,\mu\,s_1\,a_1\,s_3\,b_0\,s_6\,ack_a\,s_{10}\,ack_b\,s_{16}$ can be calculated using Def. 4 for the *symbolic* scheduler: $\Pr(\sigma^{\uparrow}) = \frac{1}{2} \cdot y_1 \cdot y_4 \cdot y_8 \cdot y_{14}$.

To construct a distributed scheduler under secrecy, we need to consider the interleaving scheduler and the local schedulers. Notice that in our example, only the local scheduler of $\mathcal{A}$ is relevant. In the parametric scheduler η of Fig. 3, variables y_i's correspond to the probabilistic choices of the interleaving scheduler, while variables x_j's correspond to the probabilistic choices of the local scheduler of $\mathcal{A}$ (all other local schedulers only have trivial choices and hence we omit them). The multiplication of variables y_i's and x_j's in the last step corresponds to the composition of the interleaving scheduler with the local scheduler in order to define the distributed scheduler η as it is in Def. 6 (which extends to Def. 8). Notice that, contrarily to the fact that variables y_i's are all different, x_1, x_2, x_3, and x_4 repeat in several branches. This has to do with the fact that some local paths of $\mathcal{A}$ are the same for different paths of the composed system. For example, the choice of the local scheduler of $\mathcal{A}$ at states s_{15}, s_{16}, s_{19}, and s_{20} is determined by the same local path $ad_0\,\mu'\,ad_1\,ack_a\,ad_2\,ack_b\,ad_3$, with $\mu'(ad_1) = 1$. (Notice that, e.g. $(s_0\,\mu\,s_1\,a_1\,s_3\,ack_a\,s_5\,b_0\,s_9\,ack_b\,s_{15})\,[\mathcal{A}] = ad_0\,\mu'\,ad_1\,ack_a\,ad_2\,ack_b\,ad_3$.)

Following Def. 4 and equation (3), the *parametric* probability of reaching a guessing state (which are underlined in Fig. 3), is given by the polynomial

$$\tfrac{1}{2} \cdot y_1 \cdot y_3 \cdot y_7 \cdot y_{13} \cdot y_{19} \cdot x_1 + \tfrac{1}{2} \cdot y_1 \cdot y_4 \cdot y_8 \cdot y_{14} \cdot y_{20} \cdot x_1 + \tfrac{1}{2} \cdot y_1 \cdot y_4 \cdot y_9 \cdot y_{15} \cdot y_{21} \cdot x_3 +$$
$$\tfrac{1}{2} \cdot y_2 \cdot y_5 \cdot y_{10} \cdot y_{16} \cdot y_{22} \cdot x_4 + \tfrac{1}{2} \cdot y_2 \cdot y_5 \cdot y_{11} \cdot y_{17} \cdot y_{23} \cdot x_2 + \tfrac{1}{2} \cdot y_2 \cdot y_6 \cdot y_{12} \cdot y_{18} \cdot y_{24} \cdot x_2\,.$$

Maximizing (resp. minimizing) the previous polynomial under the obvious constraints (each variable takes a value within $[0, 1]$, and they define proper probability distribution, e.g. $y_3 + y_4 = 1$), yields to the maximum (resp. minimum) probability under distributed schedulers. This was presented in [5].

However, this is not sufficient to characterize distributed schedulers under secrecy. Notice that, for $\sigma_a = s_0\mu s_1 a_1 s_3 b_0 s_6$ and $\sigma_b = s_0\mu s_2 a_0 s_4 b_1 s_7$ (which are the same σ_a and σ_b of Sec. 3), and components $\mathcal{S}_a$ and $\mathcal{S}_b$, we are under conditions (i), (ii), and (iii) of Def. 8. Therefore, it should hold that $\frac{\mathcal{I}(\sigma_a)(\mathcal{S}_a)}{\mathcal{I}(\sigma_a)(\mathcal{S}_a)+\mathcal{I}(\sigma_a)(\mathcal{S}_b)} = \frac{\mathcal{I}(\sigma_b)(\mathcal{S}_a)}{\mathcal{I}(\sigma_b)(\mathcal{S}_a)+\mathcal{I}(\sigma_b)(\mathcal{S}_b)}$. Since $\mathcal{I}(\sigma_a)(\mathcal{S}_a) = y_8$, $\mathcal{I}(\sigma_a)(\mathcal{S}_b) = y_9$, $\mathcal{I}(\sigma_b)(\mathcal{S}_a) = y_{11}$ and $\mathcal{I}(\sigma_b)(\mathcal{S}_b) = y_{10}$, this means that constraint $\frac{y_8}{y_8+y_9} = \frac{y_{11}}{y_{10}+y_{11}}$ needs to be considered in the optimization problem. Similarly, constraints $\frac{y_9}{y_8+y_9} = \frac{y_{10}}{y_{10}+y_{11}}$, $\frac{y_3}{y_3+y_4} = \frac{y_6}{y_5+y_6}$, and $\frac{y_4}{y_3+y_4} = \frac{y_5}{y_5+y_6}$ are also needed.

In the following, we give the formal construction of the optimization problem. Let $\mathit{Paths}^{\leq t}(\mathcal{C}) = \{\sigma \in \mathit{Paths}_{\mathit{fin}}(\mathcal{C}) \mid \mathit{time}(\sigma) \leq t\}$. We consider the following set of variables

$$\begin{aligned} V = &\{y^i_\sigma \mid \sigma \in \mathit{Paths}^{\leq t}(\mathcal{C}) \wedge 1 \leq i \leq \#\mathcal{C} \wedge \mathit{enab}(\sigma[\mathcal{P}_i]) \neq \emptyset\} \cup \\ &\{x^a_{\sigma[\mathcal{P}_i]} \mid \sigma \in \mathit{Paths}^{\leq t}(\mathcal{C}) \wedge 1 \leq i \leq \#\mathcal{C} \wedge a \in \mathit{enab}(\sigma[\mathcal{P}_i])\} \cup \\ &\{w^{j,I}_f \mid I \subseteq \{1,\dots,\#\mathcal{C}\} \wedge j \in I \wedge f : I \to \mathit{Paths}_{\mathit{fin}} \wedge \\ &\qquad \exists \sigma \in \mathit{Paths}^{\leq t}(\mathcal{C}) : \forall i \in I : \mathit{enab}(\sigma[\mathcal{P}_i]) \neq \emptyset \wedge f(i) = [\sigma[\mathcal{P}_i]]_{\sim_i}\} \end{aligned} \tag{4}$$

Variables y^i_σ and $x^a_{\sigma[\mathcal{P}_i]}$ are associated to the interleaving and local schedulers respectively, and we expect that $\mathcal{I}(\sigma)(i) = y^i_\sigma$ and $\eta_{\mathcal{P}_\mathrm{i}}(\sigma[\mathcal{P}_i])(a) = x^a_{\sigma[\mathcal{P}_i]}$. Variables $w^{j,I}_f$ are associated to the restriction of the interleaving scheduler in Def. 8. We expect that $w^{j,I}_f = \frac{\mathcal{I}(\sigma)(\mathcal{P}_j)}{\sum_{i\in I}\mathcal{I}(\sigma)(\mathcal{P}_i)} = \frac{y^j_\sigma}{\sum_{i\in I} y^i_\sigma}$ whenever $f(i) = [\sigma[\mathcal{P}_i]]_{\sim_i}$ for all $i \in I$. Notice that, if there is another σ' such that $f(i) = [\sigma'[\mathcal{P}_i]]_{\sim_i}$ for all $i \in I$, then also $w^{j,I}_f = \frac{\mathcal{I}(\sigma')(\mathcal{P}_j)}{\sum_{i\in I}\mathcal{I}(\sigma')(\mathcal{P}_i)}$. This ensure the desired equality.

In our example of Fig. 3, if $\sigma = s_0\,\mu\,s_1\,a_1\,s_3\,b_0\,s_6$, then y_8 and y_9 correspond respectively to $y^{\mathcal{S}_a}_\sigma$ and $y^{\mathcal{S}_b}_\sigma$. If $\hat\sigma = ad_0\,\mu'\,ad_1\,ack_a\,ad_2\,ack_b\,ad_3$, then x_1 and x_2 correspond respectively to $x^{ga_1}_{\hat\sigma}$ and $x^{gb_1}_{\hat\sigma}$. Moreover, notice that if $\sigma' = s_0\,\mu\,s_1\,a_1\,s_3\,b_0\,s_6\,ack_a\,s_{10}\,ack_b\,s_{16}$ and $\sigma'' = s_0\,\mu\,s_1\,a_1\,s_3\,ack_a\,s_5\,b_0\,s_9\,ack_b\,s_{15}$, then $x^{ga_1}_{\hat\sigma}$, $x^{ga_1}_{\sigma'[\mathcal{A}]}$, and $x^{ga_1}_{\sigma''[\mathcal{A}]}$ are the same variable.

Let G be the set of goal states and let t be the time bound of the time-bounded reachability property under study. Let $\mathit{Paths}^{\leq t}_G(\mathcal{C}) = \mathit{Paths}^{\leq t}(\mathcal{C}) \cap \bar{G}$. The function $\mathbf{P}$ that assigns a polynomial term with variables in V to each path in $\mathit{Paths}^{\leq t}_G(\mathcal{C})$, is defined by

$$\begin{aligned} \mathbf{P}(\hat{s}_\mathcal{C}) &= 1 \\ \mathbf{P}(\sigma\alpha s) &= \begin{cases} \mathbf{P}(\sigma)\cdot y^i_\sigma \cdot x^\alpha_{\sigma[\mathcal{P}_i]} & \text{if } \mathit{enab}(\sigma) \neq \emptyset \wedge \alpha \in A^O_{\mathcal{P}_i} \wedge \mathit{last}(\sigma) \xrightarrow{\alpha} s \\ \mathbf{P}(\sigma)\cdot \alpha(s) & \text{if } \mathit{enab}(\sigma) = \emptyset \wedge \mathit{last}(\sigma) \Rightarrow \alpha \end{cases} \\ \mathbf{P}(\sigma) &= 0 \qquad \text{in any other case} \end{aligned} \tag{5}$$

Then, following (3), the objective polynomial of the optimization problem is

$$\textstyle\sum_{\sigma\in \mathit{Paths}^{\leq t}_G(\mathcal{C})} \mathbf{P}(\sigma) \tag{6}$$

and it is subject to the following constraints:

$$0 \le v \le 1 \qquad \text{if } v \in V \tag{7a}$$

$$\textstyle\sum_{a\in A} x^a_{\sigma[\mathcal{P}_i]} = 1 \qquad \text{if } \sigma \in \mathit{Paths}^{\le t}(\mathcal{C}),\ 1 \le i \le \#\mathcal{C},\ A = \mathit{enab}(\sigma[\mathcal{P}_i]) \tag{7b}$$

$$\textstyle\sum_{i\in I} y^i_\sigma = 1 \qquad \text{if } \sigma \in \mathit{Paths}^{\le t}(\mathcal{C}),\ I = \{i \mid 1 \le i \le \#\mathcal{C}, \mathit{enab}(\sigma[\mathcal{P}_i]) \neq \emptyset\} \tag{7c}$$

$$w^{j,I}_f(\textstyle\sum_{i\in I} y^i_\sigma) = y^j_\sigma \ \text{ if} \begin{cases} \sigma \in \mathit{Paths}^{\le t}(\mathcal{C}),\ I \subseteq \{i \mid 1{\le}i{\le}\#\mathcal{C}, \mathit{enab}(\sigma[\mathcal{P}_i]) \neq \emptyset\}, \\ j \in I,\ \forall i \in I : \mathit{enab}(\sigma[\mathcal{P}_i]) \neq \emptyset \wedge f(i) = [\sigma[\mathcal{P}_i]]_{\sim_i} \end{cases} \tag{7d}$$

Equations (7a–7c) ensure that the probabilisitic choices of the local and interleaving schedulers are well defined. Equation (7d) is a rewriting of $w^{j,I}_f = \frac{y^j_\sigma}{\sum_{i\in I} y^i_\sigma}$ to avoid possible division by 0. (Notice that, when $\sum_{i\in I} y^i_\sigma = 0$, the constraint becomes trivially valid.) These constraints encode the restriction on the interleaving scheduler. In effect, let σ and σ' be such that, for all $i \in I$, $\sigma[\mathcal{P}_i] \sim_{\mathcal{P}_i} \sigma'[\mathcal{P}_i]$, $\mathit{enab}(\sigma[\mathcal{P}_i]) \neq \emptyset$ and $\mathit{enab}(\sigma'[\mathcal{P}_i]) \neq \emptyset$. Then, equations $w^{j,I}_f(\sum_{i\in I} y^i_\sigma) = y^j_\sigma$ and $w^{j,I}_f(\sum_{i\in I} y^i_{\sigma'}) = y^j_{\sigma'}$, with $f(i) = [\sigma[\mathcal{P}_i]]_{\sim_i} = [\sigma'[\mathcal{P}_i]]_{\sim_i}$, are present in the constraints and hence $\frac{y^j_\sigma}{\sum_{i\in I} y^i_\sigma} = \frac{y^j_{\sigma'}}{\sum_{i\in I} y^i_{\sigma'}}$.

We have the following theorem, whose proof we omit as it follows closely the proof of [5, Theorem 2].

Theorem 1. *Time-bounded reachability for a distributed I/O-IPC $\mathcal{C}$ under the class $\mathcal{DSS}$ is equivalent to solve the polynomial optimization problem with objective function in (6) under constraints (7). The result of maximizing (resp. minimizing) polynomial (6) is* $\sup_{\eta\in\mathcal{DSS}} \Pr_\eta(\Diamond^{\le t} G)$ *(resp.* $\inf_{\eta\in\mathcal{DSS}} \Pr_\eta(\Diamond^{\le t} G)$*).*

5 Implementation

We developed a prototypical tool to produce the optimization problem. It takes as input the model of each component of the system (as transitions between states with the initial state and equivalence classes indicated); a list of goal states G and a time-bound t.

The tool computes elements of $\mathit{Paths}^{\le t}(\mathcal{C})$ from the composed initial state following rules (1) and (2). While generating paths, new variables and constraints are defined if the conditions of Eq. (7) hold. Also the expression $\mathbf{P}$ of the path is determined according to Eq. (5). This process is iterated for each generated path as long as its last state is not in G and it has successors in $\mathit{Paths}^{\le t}(\mathcal{C})$, i.e., as long as the enabled transitions also lead to a finite path within the requested time-bound. When all the elements of $\mathit{Paths}^{\le t}_G(\mathcal{C})$ are identified, the tool exports the constrained optimization problem. For an exact solution, we set a quantifier elimination problem over the real domain as an input for Redlog[1] (within the Reduce computer algebra system) or QEPCAD[2]. For a numeric solution, we generate source code for compiling against the IPOPT[3] library.

[1] `http://redlog.dolzmann.de`
[2] `http://www.usna.edu/cs/~qepcad`
[3] `https://projects.coin-or.org/Ipopt`

The complexity of the algorithm is clear: the optimization problem grows exponentially with the number of components in the system and the degree of local nondeterminism. Therefore it is essential to find ways to reduce it to a smaller equivalent optimization problem. Curiously enough, the fact of considering arbitrary summations in the constraints for the interleaving scheduler (see Def. 8) rather than only binary, permits a drastic reduction on the number of variables and constraints as well as the size and degree of the objective polynomial.

Notice that if $I = \{i \mid \mathit{enab}(\sigma\,[\mathcal{P}_i])\}$ in (7d), we know by (7c) that $\sum_{i\in I} y^i_\sigma = 1$. As a consequence, $w^{j,I}_f = y^j_\sigma$ for all $j \in I$. If there is another σ' such that $I = \{i \mid \mathit{enab}(\sigma'\,[\mathcal{P}_i])\}$ and $f(i) = [\sigma\,[\mathcal{P}_i]]_{\sim_i}$ for all $i \in I$, then $w^{j,I}_f = y^j_{\sigma'} = y^j_\sigma$ for all $j \in I$. This only simplification has allowed us to eliminate a large number of variables and, more importantly, non-linear constraint introduced by (7d).

Moreover, this unification of variables reduces also the size and degree of the polynomial. Often, after substitution, we find out that by factorizing, we can single out $\sum_{i\in I} y^i_\sigma$ which, when $I = \{i \mid \mathit{enab}(\sigma\,[\mathcal{P}_i])\}$, equals to 1. This reduces the number of terms in $I - 1$ and the degree of these terms by 1.

Case Study: The Dining Cryptographers Protocol. As a case study we automated the verification of sender untraceability in dc-net, a protocol inspired on a solution for the *dining cryptographers* problem [8]: Three cryptographers are having dinner at a restaurant while the waiter informs that the bill has been paid anonymously. If one of the cryptographers is paying they want to respect his anonymity, but they like to know if their boss is paying. Briefly, the protocol goes as follows. Each participant toss a fair coin and share the outcome with his neighbour at the left, then he publicly announces if his outcome and the one of the neighbour are the same, but if the participant is paying, he announces the opposite. If the number of "different" announces is odd, one of the participants is paying but the others cannot distinguish which one.

This protocol has been analyzed a large number of times with different techniques. It is noticeable that most of the proofs, if not all (in particular model-based fully automated proofs), fix the order in which the participants make their announcements. Most generally, this has to do precisely with the inability of the techniques to control the arbitrariness of the scheduler. The model we verify does not impose such restrictions. We anyway prove that the protocol preserves anonymity when the participants' announcement do not follow a fixed order.

We consider the case of three cryptographers C_1, C_2 and C_3. We fix the probability of the boss paying in $\frac{1}{2}$. Otherwise, the probability of any of the participants paying is uniformly distributed ($\frac{1}{6}$ each). We want to know what is the probability P_{guess} that participant C_3 correctly guesses if C_1 or C_2 have paid, knowing that one of them has actually paid. We will then calculate the maximum and minimum probability of reaching the set of states $G = \{s \mid C_3 \text{ guesses } C_i \text{ and } C_i \text{ paid}, i = 1, 2\}$, say $P^+_{\Diamond G}$ and $P^-_{\Diamond G}$, respectively. Knowing that the probability that C_1 or C_2 have paid is $\frac{1}{3}$, if it turns out that $P^+_{\Diamond G} = P^-_{\Diamond G} = p$, the conditional probability that we are looking for is $P_{\text{guess}} = p/\frac{1}{3} = 3p$.

Indeed, after running our tool, we verified that $P^{+}_{\Diamond G} = P^{-}_{\Diamond G} = \frac{1}{6}$. Hence $P_{\text{guess}} = \frac{1}{2}$, proving that C_3 has no better knowledge than previously known (i.e. that any of C_1 or C_2 pay with probability $\frac{1}{2}$). The original system contains 9606 variables, 1348 linear constraints, 16176 nonlinear constraints and the polynomial has 3456 terms and degree 7. After unification of variables the numbers are 774, 200, 36, 3456, and 7, resp., and after factorization and elimination of irrelevant variables, numbers reduces to 351, 17, 0, 544, and 5, resp. Given the complexity of the optimization problem, we were unable to solve it exactly, but the numerical computation was almost immediate. On an aside note, a similar verification of our running example was solved using only the simplifications.

6 Concluding Remarks

Related Work. The interest on understanding and verifying probabilistic distributed systems under the assumption that not all information is shared by all components has appeared several times before in the literature (e.g. [9,14,11,1]). We base this work on extending the framework of [12] and the algorithm of [5].

Our ideas about equivalence relations and limiting the interleaving scheduler based on projections under equivalences are inspired in task-PIOAs [6]. Task-PIOAs are a variation of probabilistic I/O automata with an equivalence relation over the set of "controllable actions" of the composed system. The model restrict to output isolation and action determinism. The set of schedulers they considered are a combination of a *task schedule* with a regular total-information scheduler. The resulting scheduler maps equivalent execution fragments to probability measures that ensure that equivalent actions receive the same probability value. This roughly corresponds to our interleaving scheduling under secrecy assumptions. Because of output isolation, no other scheduler is needed. Despite that [6] provides techniques to analyze time-bounded attacks in Task-PIOA, to our knowledge, no fully automatic algorithm is provided.

[2] proposes another restricted family of schedulers over *tagged probabilistic automata*, a formalism with similar semantics to ours. The actions in composed systems are "tagged" with the components engaged in the action (or components, in case of synchronization). The set of enabled tags in a state is part of the observable behavior. Then, the scheduler is defined as a function from *observable* traces to tags. In this sense, the schedulers are quite like our interleaving schedulers, and no local scheduler is needed. Another restriction is that they only consider deterministic schedulers (which are strictly less expressive than randomised schedulers, see [14]). [2] also provides an automatic technique based on automorphism that check for sufficient conditions to ensure anonymity in systems whose components do not have internal nondeterminism (comparable to our output nondeterminism).

A somewhat similar approach is presented in [7] where labels are also used for the scheduler to resolve the nondeterminism. In this case the scheduler is provided explicitly as a deterministic component that only synchronizes with the system through labels. Again randomized schedulers are not considered. A particularity of this work is that the "equivalence classes" (which are actually defined by how labels are associated to actions) can change dynamically.

Conclusion and Further Work. We refined the schedulers of [14] to deal with information hiding and adapted the technique of [5] to the new setting. Moreover, the generation of the polynomial optimization problem has been significantly improved, first by avoiding the generation of the intermediate parametric Markov chain, and then by adding simplification rules that drastically reduced the size of the optimization problem. In addition, the connection to numerical tools allow for effective and rapid calculations. In particular, our technique allowed for the verification of a more nondeterministic version of the dining cryptographers, without constraining the ordering of independent actions.

In the future, we propose to revise the synchronisation mechanism. Notice that, in the running example, adversary $\mathcal{A}$ does not observe the communication of the secret at all. However, it is reasonable that $\mathcal{A}$ observes the transmission of the *encrypted* secret, that is, $\mathcal{A}$ should synchronize with the equivalence class without knowing which particular action was executed. Another further work is to study the use of our framework to analize timing attacks, after all it is already prepared for it. Moreover, by properly bounding the numbers of steps of the attackers, we may explore the possibility of restricting the computational complexity of the attack.

References

1. Andrés, M.: Quantitative Analysis of Information Leakage in Probabilistic and Nondeterministic Systems. Ph.D. thesis, Radboud Universiteit Nijmegen (2011)
2. Andrés, M.E., Palamidessi, C., van Rossum, P., Sokolova, A.: Information hiding in probabilistic concurrent systems. Theor. Comput. Sci. 412(28), 3072–3089 (2011)
3. Bhargava, M., Palamidessi, C.: Probabilistic anonymity. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 171–185. Springer, Heidelberg (2005)
4. Bianco, A., de Alfaro, L.: Model checking of probabilistic and nondeterministic systems. In: Thiagarajan, P.S. (ed.) FSTTCS 1995. LNCS, vol. 1026, pp. 499–513. Springer, Heidelberg (1995)
5. Calin, G., Crouzen, P., D'Argenio, P.R., Hahn, E., Zhang, L.: Time-bounded reachability in distributed input/output interactive probabilistic chains. In: van de Pol, J., Weber, M. (eds.) SPIN 2010. LNCS, vol. 6349, pp. 193–211. Springer, Heidelberg (2010); extended version: AVACS Technical Report No. 64 (June 2010)
6. Canetti, R., Cheung, L., Kaynar, D., Liskov, M., Lynch, N., Pereira, O., Segala, R.: Analyzing security protocols using time-bounded Task-PIOAs. Discrete Event Dynamic Systems 18, 111–159 (2008)
7. Chatzikokolakis, K., Palamidessi, C.: Making random choices invisible to the scheduler. Information and Computation 208(6), 694–715 (2010)
8. Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology 1, 65–75 (1988)
9. Cheung, L., Lynch, N., Segala, R., Vaandrager, F.: Switched PIOA: Parallel composition via distributed scheduling. Theor. Comput. Sci. 365(1-2), 83–108 (2006)

10. Coste, N., Hermanns, H., Lantreibecq, E., Serwe, W.: Towards performance prediction of compositional models in industrial GALS designs. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 204–218. Springer, Heidelberg (2009)
11. Georgievska, S., Andova, S.: Retaining the probabilities in probabilistic testing theory. In: Ong, L. (ed.) FOSSACS 2010. LNCS, vol. 6014, pp. 79–93. Springer, Heidelberg (2010)
12. Giro, S.: On the automatic verification of distributed probabilistic automata with partial information. Ph.D. thesis, Universidad Nacional de Córdoba (2010)
13. Giro, S., D'Argenio, P.R.: Quantitative model checking revisited: Neither decidable nor approximable. In: Raskin, J.-F., Thiagarajan, P.S. (eds.) FORMATS 2007. LNCS, vol. 4763, pp. 179–194. Springer, Heidelberg (2007)
14. Giro, S., D'Argenio, P.R.: On the expressive power of schedulers in distributed probabilistic systems. Electron. Notes Theor. Comput. Sci. 253(3), 45–71 (2009)
15. Reiter, M.K., Rubin, A.D.: Crowds: Anonymity for web transactions. ACM Trans. Inf. Syst. Secur. 1(1), 66–92 (1998)
16. Smith, G.: On the foundations of quantitative information flow. In: de Alfaro, L. (ed.) FOSSACS 2009. LNCS, vol. 5504, pp. 288–302. Springer, Heidelberg (2009)
17. Wang, Y.: Real-time behaviour of asynchronous agents. In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR 1990. LNCS, vol. 458, pp. 502–520. Springer, Heidelberg (1990)