Information Hiding in Probabilistic Concurrent Systems

Miguel E. Andrés[1], Catuscia Palamidessi[2], Peter van Rossum[1], and Ana Sokolova[3].
[1]Institute for Computing and Information Sciences, The Netherlands.
[2]INRIA and LIX, École Polytechnique Palaiseau, France.
[3]Department of Computer Sciences, University of Salzburg, Austria.

***Abstract*—Information hiding is a general concept which refers to the goal of preventing an adversary to infer secret information from the observables. Anonymity and Information Flow are examples of this notion. We study the problem of information hiding in systems characterized by the presence of randomization and concurrency. It is well known that the raising of nondeterminism, due to the possible interleavings and interactions of the parallel components, can cause unintended information leaks. One way to solve this problem is to fix the strategy of the scheduler beforehand. In this work, we propose a milder restriction on the schedulers, and we define the notion of strong (probabilistic) information hiding under various notions of observables. Furthermore, we propose a method, based on the notion of automorphism, to verify that a system satisfies the property of strong information hiding, namely strong anonymity or no-interference, depending on the context.**

I. Introduction

The problem of information hiding consists in trying to prevent the adversary to infer confidential information from the observables. Instances of this issue are Anonymity and Information Flow. In both fields there is a growing interest in the quantitative aspects, see for instance [20], [2], [10], [12], [8], [31]. This is justified by the fact that often we have some a priori knowledge about the likelihood of the various secrets, and by the fact that protocols often use randomized actions to obfuscate the link between secret and observable, like in anonymity protocols such as the DC Nets [9], Crowds [26], Onion Routing [32], and Freenet [11].

In a concurrent setting, like in the case of multi-agent systems, there is also another source of uncertainty, which derives from the fact that the various entities may interleave and interact in ways that are usually unpredictable, either because they depend on factors that are too complex to analyze, or because (in the case of specifications) they are implementation-dependent. This uncertainty is commonly modeled as nondeterminism.

The formal analysis of systems which exhibit probabilistic and nondeterministic behavior usually involves the use of *schedulers*, which are functions that select, for each path, only one possible (probabilistic) transition, thus delivering a purely probabilistic execution tree.

In the area of security, there is the problem that among all possible schedulers there are also those which take different decisions depending on the secret values, and these decisions may induce different observable behaviors, thus leaking the secret. Hence the security properties are usually violated if we consider all possible schedulers: "obviously secure" protocols are not secure anymore. This is a well known problem for which various solutions have already been proposed. We will come back to these in the "Related work" section.

A. Contribution

The main contribution of this work consists in the following:

- We define a class of partial-information schedulers, which we call *admissible*. These are a restricted version of the standard (full-information) schedulers. The restriction is rather flexible and has strong structural properties, which facilitate the reasoning about security properties. In short, our systems consist of parallel components with certain restrictions on the secret and nondeterministic choices. The scheduler selects the next component (or components, in case of synchronization) for the subsequent step independently of the secret choices. We then formalize the notion of quantitative information leakage under this restricted notion of scheduler.
- We propose alternative definitions to the property of strong anonymity defined in [2]. The differences of our proposal are: (1) the system should be strongly anonymous for all admissible schedulers instead of all schedulers, and (2) we consider several variants of adversaries, namely (in increasing level of power): external adversaries, internal adversaries, and adversaries in collusion with the scheduler. Additionally, we use admissible schedulers to extend the notions of multiplicative [31] and additive leakage [3] to the case of concurrent system.
- We propose a sufficient technique to prove probabilistic strong anonymity, and probabilistic noninterference, based on automorphisms. The idea is the following: In the purely nondeterministic setting, the strong anonymity of a system is often defined and proved as follows: take two users A and B and a trace in which user A is 'the culprit'. Now find a trace that looks the same to the adversary, but in which user B is 'the culprit' [20], [16], [23], [21]. This new trace is often most easily obtained by *switching the behavior of A and B*. Non-interference can be proved in the same way (where A and B are high information and the trace is the low information).

  In this work, we develop this technique for systems where probability and nondeterminism coexist, and we need to cope with the restrictions on the schedulers. We formalize

the notion of *switching the behaviors* of A and B in terms of the existence of an automorphism between A and B, and then we show that the existence of an automorphism implies strong anonymity.
- We use the Dining Cryptographers [9] to illustrate the problem caused by full-information schedulers, our solution based on admissible schedulers, and our proving technique.

B. Related Work

The problem of the full-information scheduler has already been extensively investigated in literature. The works [4] and [5] consider probabilistic automata and introduce a restriction on the scheduler to the purpose of making them suitable to applications in security. 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. Their notion of task scheduler is similar to our notion of admissible scheduler, but more restricted since the strategy of the task scheduler is decided entirely before the execution.

The work in [7], [6] 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.

In [1] we extend the framework presented here by allowing internal nondeterminism and adding a second type of scheduler to resolve it, to the aim of investigating angelic vs demonic nondeterminism in equivalence-based properties.

The fact that full-information schedulers are unrealistic has also been observed in fields other than security. First attempts used restricted schedulers in order to obtain rules for compositional reasoning [14]. 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. However that work considers a synchronous parallel composition, so the setting is rather different from ours. Later on, it was shown that model checking is unfeasible in its general form for the restricted schedulers in [14] (see [18] and, more recently, [17]). 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 [19].

To the best of our knowledge, this is the first work using automorphisms as a sound proof technique to prove strong anonymity and non-interference. The closest line of work we are aware of is in the field of model checking, where isomorphisms have been used to identify symmetries in the system and exploited to alleviate the state space explosion (see for instance [22]).

C. Plan of the paper

Looking ahead, after reviewing some preliminaries (Section II) we formalize the notions of systems and components (Section III). In Section IV we present admissible schedulers. We then formalize the notions of internal and external strong anonymity in a probabilistic and nondeterministic setting for admissible schedulers (Section V). Finally, we turn our attention to the verification problem, in Section VI we present a strong-anonymity proving technique based on automorphisms. We conclude and outline some future work in Section VII.

II. Preliminaries

In this section we gather preliminary notions and results related to probabilistic automata [29], [28].

A. Probabilistic automata

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

A *probabilistic automaton* is a quadruple $M = (Q, \Sigma, \hat{q}, \theta)$ where Q is a countable set of *states*, Σ a finite set of *actions*, $\hat{q}$ the *initial* state, and θ a *transition function* $\theta : Q \to \mathcal{P}(\mathcal{D}(\Sigma \times Q))$. Here $\mathcal{P}(X)$ is the set of all subsets of X.

If $\theta(q) = \emptyset$, then q is a *terminal* state. We write $q{\to}\mu$ for $\mu \in \theta(q),\ q \in Q$. Moreover, we write $q\xrightarrow{a}r$ for $q, r \in Q$ whenever $q{\to}\mu$ and $\mu(a, r) > 0$. A *fully probabilistic automaton* is a probabilistic automaton satisfying $|\theta(q)| \leq 1$ for all states. In case $\theta(q) \neq \emptyset$ in a fully probabilistic automaton, we will overload notation and use $\theta(q)$ to denote the distribution outgoing from q. A *path* in a probabilistic automaton is a sequence $\sigma = q_0 \overset{a_1}{\to} q_1 \overset{a_2}{\to} \cdots$ where $q_i \in Q$, $a_i \in \Sigma$ and $q_i \overset{a_{i+1}}{\to} q_{i+1}$. 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, *first*(σ) denotes its first state, and if σ is finite then *last*(σ) denotes its last state. A *cycle* is a path σ such that *last*(σ) = *first*(σ). 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 $\Sigma^* \cup \Sigma^\infty$ obtained by removing the states, hence for the above path σ we have *trace*$(\sigma) = a_1 a_2 \ldots$. If $\Sigma' \subseteq \Sigma$, then $\mathit{trace}_{\Sigma'}(\sigma)$ is the projection of *trace*(σ) on the elements of Σ'.

Let $M = (Q, \Sigma, \hat{q}, \theta)$ 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 $M = (Q, \Sigma, \hat{q}, \theta)$ 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 $\mathbf{P}_q(\sigma \xrightarrow{a} q') = \mathbf{P}_q(\sigma)\ \mu(a, q')$, where $last(\sigma) \to \mu$.

Let $\Omega_q \stackrel{\text{def}}{=} \text{CPaths}_q(M)$ 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 (full-information) scheduler for a probabilistic automaton M is a function $\zeta\colon \text{Paths}^\star(M) \to (\mathcal{D}(\Sigma \times Q) \cup \{\bot\})$ such that for all finite paths σ, if $\theta(last(\sigma)) \neq \emptyset$ then $\zeta(\sigma) \in \theta(last(\sigma))$, 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. Note that a scheduler is history dependent since it can take different decisions for the same state s according to the past evolution of the system.

B. Noisy Channels

This section briefly recalls the notion of noisy channels from Information Theory [13].

A *noisy channel* is a tuple $\mathcal{C} \stackrel{\text{def}}{=} (\mathcal{X}, \mathcal{Y}, P(\cdot|\cdot))$ where $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ is a finite set of *input values*, modeling the *secrets* of the channel, and $\mathcal{Y} = \{y_1, y_2, \dots, y_m\}$ is a finite set of *output values*, the *observables* of the channel. For $x_i \in \mathcal{X}$ and $y_j \in \mathcal{Y}$, $P(y_j|x_i)$ is the conditional probability of obtaining the output y_j given that the input is x_i. These conditional probabilities constitute the so called *channel matrix*, where $P(y_j|x_i)$ is the element at the intersection of the i-th row and the j-th column. For any input distribution P_X on $\mathcal{X}$, P_X and the channel matrix determine a joint probability $P_\wedge$ on $\mathcal{X} \times \mathcal{Y}$, and the corresponding marginal probability P_Y on $\mathcal{Y}$ (and hence a random variable Y). P_X is also called *a priori distribution* and it is often denoted by π. The probability of the input given the output is called *a posteriori distribution*.

C. Information leakage

We recall here the definitions of *multiplicative leakage* proposed in [31], and of *additive leakage* proposed in [3][1]. We assume given a noisy channel $\mathcal{C} = (\mathcal{X}, \mathcal{Y}, P(\cdot|\cdot))$ and a random variable X on $\mathcal{X}$. The *a priori vulnerability* of the secrets in $\mathcal{X}$ is the probability of guessing the right secret, defined as $V(X) \stackrel{\text{def}}{=} \max_{x\in\mathcal{X}} P_X(x)$. The rationale behind this definition is that the adversary's best bet is on the secret with highest probability. The *a posteriori vulnerability* of the secrets in $\mathcal{X}$ is the probability of guessing the right secret, after the output has been observed, averaged over the probabilities of the observables. The formal definition is $V(X\,|\,Y) \stackrel{\text{def}}{=} \sum_{y\in\mathcal{Y}} P_Y(y) \max_{x\in\mathcal{X}} P(x\,|\,y)$. Again, this definition is based on the principle that the adversary will choose the secret with the highest a posteriori probability.

Note that, using Bayes theorem, we can write the a posteriori vulnerability in terms of the channel matrix and the a priori distribution, or in terms of the joint probability:

$$V(X\,|\,Y) = \sum_{y\in\mathcal{Y}} \max_{x\in\mathcal{X}} (P(y\,|\,x) P_X(x)) = \sum_{y\in\mathcal{Y}} \max_{x\in\mathcal{X}} P_\wedge(x, y)$$

The *multiplicative* leakage is $\mathcal{L}_\times(\mathcal{C}, P_X) \stackrel{\text{def}}{=} \frac{V(X|Y)}{V(X)}$ whereas the *additive* leakage is $\mathcal{L}_+(\mathcal{C}, P_X) \stackrel{\text{def}}{=} V(X|Y) - V(X)$.

[1]The notion proposed by Smith in [31] was given in a (equivalent) logarithmic form, and called simply *leakage*. For uniformity sake we use here the terminology and formulation of [3].

D. Dining Cryptographers

This problem, described by Chaum in [9], involves a situation in which three cryptographers are dining together. At the end of the dinner, each of them is secretly informed by a central agency (master) whether he should pay the bill, or not. So, either the master will pay, or one of the cryptographers will be asked to pay. The cryptographers (or some external observer) would like to find out whether the payer is one of them or the master. However, if the payer is one of them, they also wish to maintain anonymity over the identity of the payer.

A possible solution to this problem, described in [9], is that each cryptographer tosses a coin, which is visible to himself and his neighbor to the left. Each cryptographer observes the two coins that he can see and announces *agree* or *disagree*. If a cryptographer is not paying, he will announce *agree* if the two sides are the same and *disagree* if they are not. The paying cryptographer will say the opposite. It can be proved that if the number of disagrees is even, then the master is paying; otherwise, one of the cryptographers is paying. Furthermore -for the case of fair coins, if one of the cryptographers is paying, then neither an external observer nor the other two cryptographers can identify, from their individual information, who exactly is paying. The Dining Cryptographers (DC) will be a running example through the paper.

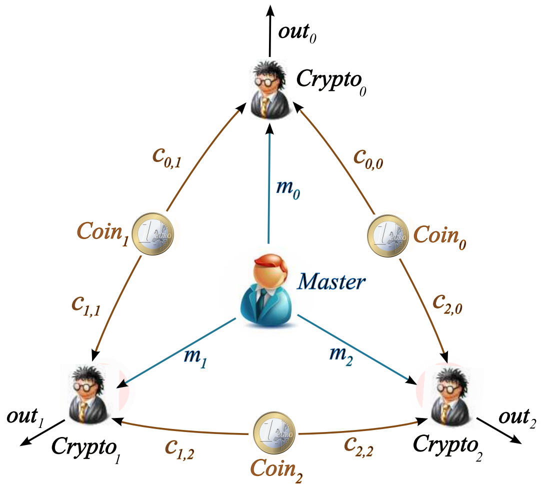

Fig. 1. Chaum's system for the Dining Cryptographers ([9])

III. 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*. These systems are parallel compositions of purely probabilistic processes, that we call *components*. They are equipped with a unique identifier, that we call *tag*, or *label*, of the component. Note that, because of the restriction that the components are fully deterministic[2], nondeterminism is generated only from the interleaving of the parallel components. Furthermore, because of the uniqueness of the tags, each transition from a node is associated to a different tag / pair of two tags (one in case only one component makes a step, and two in case of a synchronization step among two components).

A. Tagged Probabilistic Automata

We now formalize the notion of TPA.

Definition 1. *A* tagged probabilistic automaton *(TPA) is a tuple* $(Q, L, \Sigma, \hat{q}, \theta)$*, where*

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

with the additional requirement that for every $q \in Q$ *and every* $\ell \in L$ *there is at most one* $\mu \in D(\Sigma \times Q)$ *such that* $(\ell, \mu) \in \theta(q)$*.*

A path for a TPA is a sequence $\sigma = q_0 \xrightarrow{l_1, a_1} q_1 \xrightarrow{l_2, a_2} q_2 \cdots$. In this way, the process with identifier l_i induces the system to move from q_{i-1} to q_i performing the action a_i, and it does so with probability $\mu_{l_i}(a_i, q_i)$, where μ_{l_i} is *the* distribution associated to the choice made by the component l_i. Finite paths and complete paths are defined in a similar manner.

In a TPA, the scheduler's choice is determined by the choice of the tag. We will use $enab(q)$ to denote the tags of the components that are enabled to make a transition. Namely,

$$enab(q) \stackrel{\text{def}}{=} \{\ell \in L \mid \exists\, \mu \in D(\Sigma \times Q) \,:\, (\ell, \mu) \in \theta(q)\} \quad (1)$$

We assume that the scheduler is forced to select a component among those which are enabled, i.e., 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 [30]. 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 which are possible from the last state.

Definition 2. *A* scheduler *for a TPA* $M = (Q, L, \Sigma, \hat{q}, \theta)$ *is a function* $\zeta \colon \mathrm{Paths}^\star(M) \to (L \cup \{\bot\})$ *such that for all finite paths* σ*,* $\zeta(\sigma) \in enab(last(\sigma))$ *if* $enab(last(\sigma)) \neq \emptyset$ *and* $\zeta(\sigma) = \bot$ *otherwise.*

B. Components

To specify the components we use a sort of probabilistic version of CCS [24], [25]. We assume a set of *secret actions* Σ_S with elements $s, s_1, s_2, \cdots$, and a disjoint set of *observable actions* Σ_O with elements $a, a_1, a_2, \cdots$. Furthermore we have *communication actions*, which are also observable, of the form $c(x)$ (receive x on channel c, where x is a formal parameter), or $\bar{c}\langle v\rangle$ (send v on channel c, where v is a value on some domain V). Sometimes we need only to synchronize without transmitting any value, in which case we will use simply c and $\bar{c}$. We denote the set of channel names by C.

A component q is specified by the following grammar:

Components

q	$::=$	0	termination
	$\mid$	$a.q$	observable prefix
	$\mid$	$\sum_i p_i : q_i$	blind choice
	$\mid$	$\sum_i p_i : s_i.q_i$	secret choice
	$\mid$	$if\ x = v\ then\ q_1\ else\ q_2$	conditional
	$\mid$	A	process call

Observables

a	$::=$	$c \mid \bar{c}$	simple synchronization
	$\mid$	$c(x) \mid \bar{c}\langle v\rangle$	synchronization and communication

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 $\sum_i p_i = 1$. When no confusion arises, we use simply $+$ for a binary choice. The process call A is a simple process identifier. For each of them, 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 remark once again that each component contains only probabilistic and sequential constructs. In particular, there is no internal parallelism. Hence each component corresponds to a purely probabilistic automaton (apart from the input nondeterminism, which disappears in the definition of a system), as described by the operational semantics below.

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

$$\text{PRF1} \quad \frac{v \in V}{c(x).q \to \delta(c(v), q[v/x])}$$

$$\text{PRF2} \quad \frac{}{a.q \to \delta(a, q)} \quad \text{if } a \neq c(x)$$

$$\text{INT} \quad \frac{}{\sum_i p_i : q_i \to \sum_i p_i \cdot \delta(\tau, q_i)}$$

$$\text{SECR} \quad \frac{}{\sum_i p_i : s_i.q_i \to \sum_i p_i \cdot \delta(s_i, q_i)}$$

$$\text{CALL} \quad \frac{q \to \mu}{A \to \mu} \quad \text{if } A \stackrel{\text{def}}{=} q$$

[2] In [1] we extend our framework by allowing nondeterministic choices in the components, and we use an additional scheduler to handle such "internal" nondeterminism.

$$\text{CND1}\quad \frac{}{\mathit{if}\ v = v\ \mathit{then}\ q_1\ \mathit{else}\ q_2 \to \delta(\tau, q_1)}$$

$$\text{CND2}\quad \frac{v \neq v'}{\mathit{if}\ v = v'\ \mathit{then}\ q_1\ \mathit{else}\ q_2 \to \delta(\tau, q_2)}$$

$\sum_i p_i \cdot \mu_i$ is the distribution μ such that $\mu(x) = \sum_i p_i \mu_i(x)$. We use $\delta(x)$ to represent the delta of Dirac, which assigns probability 1 to x. The silent action, τ, is a special action different from all the observable and the secret actions. $q[v/x]$ stands for the process q in which any occurrence of x has been replaced by v. To shorten the notation, in the examples throughout the paper, we omit writing explicit termination, i.e., we omit the symbol 0 at the end of a term.

C. Systems

A system consists of n processes (components) in parallel, and restricted at the top-level on the set of channel names C:

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

The restriction on C enforces synchronization (and possibly communication) on the channel names belonging to C, in accordance with the CCS spirit. Since C is the set of all channels, all of them are forced to synchronize. This is to eliminate, at the level of systems, the nondeterminism generated by the rule for the receive prefix, PRF1.

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 $q \xrightarrow{\ell} \mu$ where $\mu \in (\mathcal{D}(\Sigma \times Q))$ and $\ell \in L$ is either the identifier of the component which makes the move, or a two-element set of identifiers representing the two partners of a synchronization. The following two rules (INT) and (SYNC/COMM) provide the operational semantics rules in the case of interleaving and synchronisation/communication, respectively.

(INT) If $a_j \notin C$

$$\frac{q_i \to \sum_j p_j \cdot \delta(a_j, q_{ij})}{(C)\ q_1 \parallel \cdots \parallel q_i \parallel \cdots \parallel q_n \xrightarrow{i} \sum_j p_j \cdot \delta(a_j, (C)\ q_1 \parallel \cdots \parallel q_{ij} \parallel \cdots \parallel q_n)}$$

where i indicates the tag of the component making the step.

(SYNC/COMM)

$$\frac{q_i \to \delta(\bar{c}\langle v\rangle, q_i') \qquad q_j \to \delta(c(v), q_j')}{(C)\ q_1 \parallel \cdots \parallel q_i \parallel \cdots \parallel q_n \xrightarrow{\{i,j\}} \delta(\tau, (C)\ q_1 \parallel \cdots \parallel q_i' \parallel \cdots \parallel q_j' \parallel \cdots \parallel q_n)}$$

here $\{i, j\}$ is the tag indicating that the components making the step are i and j. For simplicity we write $\xrightarrow{i,j}$ instead of $\xrightarrow{\{i,j\}}$. The rule for synchronization without communication is similar, the only difference is that we do not have $\langle v\rangle$ and (v) in the actions. Note that c can only be an observable action (neither a secret nor τ), by the assumption that channel names can only be observable actions.

We note that both interleaving and synchronization rules generate nondeterminism. The only other source of nondeterminism is PRF1, the rule for a receive prefix $c(x)$. However the latter is not real nondeterminism: it is introduced in the semantics of the components but it disappears in the semantics of the systems, given that the channel c is restricted at the top-level. In fact the restriction enforces communication, and when communication takes place, only the branch corresponding to the actual value v transmitted by the corresponding send action is maintained, all the others disappear.

Proposition 1. *The operational semantics of a system is a TPA with the following characteristics:*

(a) Every step $q \xrightarrow{\ell} \mu$ is either

a blind choice: $\mu = \sum_i p_i \cdot \delta(\tau, q_i)$, *or*

a secret choice: $\mu = \sum_i p_i \cdot \delta(s_i, q_i)$, *or*

a delta of Dirac: $\mu = \delta(\alpha, q')$ *with* $\alpha \in \Sigma_O$ *or* $\alpha = \tau$.

(b) If $q \xrightarrow{\ell} \mu$ and $q \xrightarrow{\ell} \mu'$ then $\mu = \mu'$.

Example 1. We now present the components for the Dining Cryptographers using the introduced syntax. They correspond to Figure 1 and to the automata depicted in Figure 3. As announced before, we omit the symbol 0 for explicit termination at the end of each term. The secret actions s_i represent the choice of the payer. The operators $\oplus, \ominus$ represent the sum modulo 2 and the difference modulo 2, respectively. The test $i == n$ returns 1 (true) if $i = n$, and 0 otherwise. The set of restricted channel names is $C = \{c_{0,0}, c_{0,1}, c_{1,1}, c_{1,2}, c_{2,0}, c_{2,2}, m_0, m_1, m_2\}$.

$$\begin{aligned}
\text{Master} &\stackrel{\text{def}}{=} p : \overline{m}_0\langle 0\rangle \,.\, \overline{m}_1\langle 0\rangle \,.\, \overline{m}_2\langle 0\rangle \,+ \\
&\qquad (1-p) : \textstyle\sum_{i=0}^{2} p_i : s_i \,. \\
&\qquad \overline{m}_0\langle i == 0\rangle \,.\, \overline{m}_1\langle i == 1\rangle \,.\, \overline{m}_2\langle i == 2\rangle \\
\text{Crypt}_i &\stackrel{\text{def}}{=} m_i(\mathit{pay}) \,.\, c_{i,i}(\mathit{coin}_1) \,.\, c_{i,i\oplus 1}(\mathit{coin}_2) \,. \\
&\qquad \overline{\mathit{out}}_i\langle \mathit{pay} \oplus \mathit{coin}_1 \oplus \mathit{coin}_2\rangle \\
\text{Coin}_i &\stackrel{\text{def}}{=} 0.5 : \bar{c}_{i,i}\langle 0\rangle \,.\, \bar{c}_{i\ominus 1,i}\langle 0\rangle \ + \ 0.5 : \bar{c}_{i,i}\langle 1\rangle \,.\, \bar{c}_{i\ominus 1,i}\langle 1\rangle \\
\text{System} &\stackrel{\text{def}}{=} (C)\ \text{Master} \parallel \textstyle\prod_{i=0}^{2} \text{Crypt}_i \parallel \prod_{i=0}^{2} \text{Coin}_i
\end{aligned}$$

Fig. 2. Dining Cryptographers CCS

The operation $\mathit{pay} \oplus \mathit{coin}_1 \oplus \mathit{coin}_2$ in Figure 2 is syntactic sugar, it can be defined using the *if-then-else* operator. Note that, in this way, if a cryptographer is not paying (pay = 0), then he announces 0 if the two coins are the same (agree) and 1 if they are not (disagree).

IV. Admissible Schedulers

We now introduce the class of admissible schedulers.

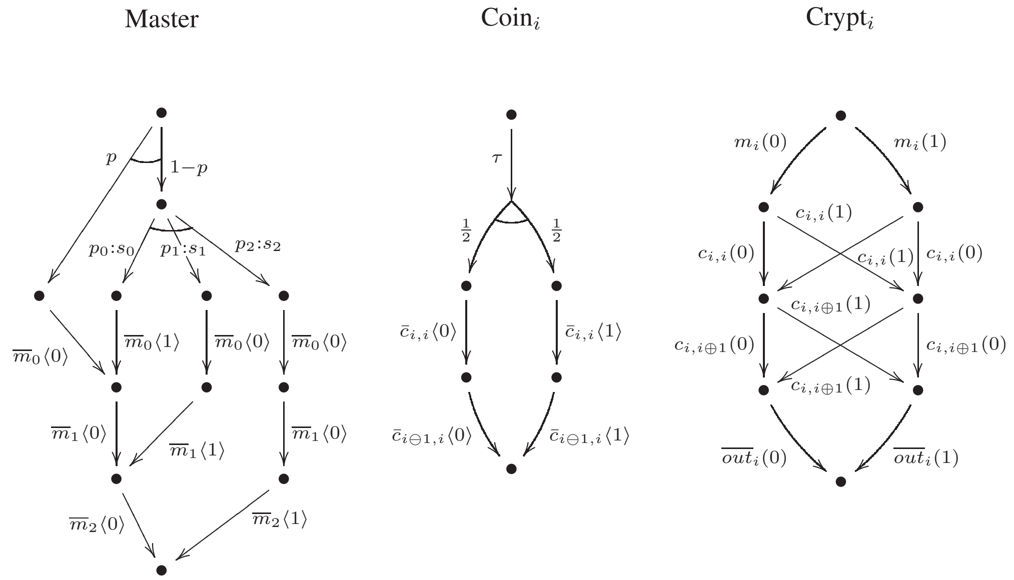

Fig. 3. Dining Cryptographers Automata

Standard (full-information) schedulers have access to all the information about the system and its components, and in particular the secret choices. Hence, such schedulers can leak secrets by making their decisions depend on the secret choice of the system. This is the case with the Dining Cryptographers protocol of Section II-D: among all possible schedulers for the protocol, there are several that leak the identity of the payer. In fact the scheduler has the freedom to decide the order of the announcements of the cryptographers (interleaving), so a scheduler could choose to let the payer announce lastly. In this way, the attacker learns the identity of the payer simply by looking at the interleaving of the announcements.

A. *The screens intuition*

Let us first describe admissible schedulers informally. As mentioned in the introduction, admissible schedulers can base their decisions only on partial information about the evolution of the system, in particular admissible schedulers cannot base their decisions on information concerned with the internal behavior of components (such as secret choices).

We follow the subsequent intuition: admissible schedulers are entities that have access to a screen with buttons, where each button represents one (current) available option. At each point of the execution the scheduler decides the next step among the available options (by pressing the corresponding button). Then (if any) the output of the selected component becomes available to the scheduler and the screen is refreshed with the new available options (the ones corresponding to the system after making the selected step). We impose that the scheduler can base its decisions only on such information, namely: the screens and outputs he has seen up to that point of the execution (and, of course, the decisions he has made).

Example 2. Consider $S \stackrel{\text{def}}{=} (\{c_1, c_2\})\ q_1 \,\|\, q_2 \,\|\, q_3$, where

$$q_1 \stackrel{\text{def}}{=} 0.5 : s_1.\bar{c}_1.\bar{c}_2 + 0.5 : s_2.\bar{c}_1.\bar{c}_2,$$
$$q_2 \stackrel{\text{def}}{=} c_1.(0.5 : a_1 + 0.5 : b_1), \quad q_3 \stackrel{\text{def}}{=} c_2.(0.5 : a_2 + 0.5 : b_2).$$

Figure 4 shows the sequence of screens corresponding to a particular sequence of choices taken by the scheduler[3]. Interleaving and communication options are represented by yellow and red buttons, respectively. An arrow between two screens represents the transition from one to the other (produced by the scheduler pressing a button), additionally, the decision taken by the scheduler and corresponding outputs are depicted above each arrow.

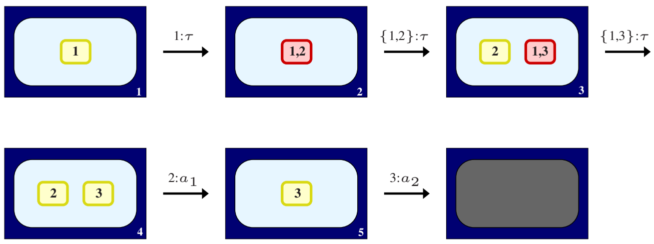

Fig. 4. Screens intuition

Note that this system has exactly the same problem as the DC protocol: a full-information scheduler could reveal the secret by basing the interleaving order (q_2 first or q_3 first) on the secret choice of the component q_1. However, the same does not hold anymore for admissible schedulers (the scheduler cannot deduce the secret choice by just looking at the screens and outputs). This is also the case for the DC protocol, i.e., admissible schedulers cannot leak the secret of the protocol.

B. *The formalization*

Before formally defining admissible schedulers we need to formalize the ingredients of the screens intuition. The buttons on the screen (available options) are the enabled options given by the function *enab* (see (1)), the decision made

[3]The transitions from screens 4 and 5 represent 2 steps each (for simplicity we omit the τ-steps generated by blind choices)

by the scheduler is the tag of the selected enabled option, observable actions are obtained by sifting the secret actions to the schedulers by means of the following function:

$$\mathit{sift}(\alpha) \stackrel{\text{def}}{=} \begin{cases} \alpha & \text{if } \alpha \in \Sigma_O \cup \{\tau\}, \\ \tau & \text{if } \alpha \in \Sigma_S. \end{cases}$$

The partial information of a certain evolution of the system is given by the map t defined as follows.

Definition 3. *Let $\hat{q} \xrightarrow{\ell_1,\alpha_1} \cdots \xrightarrow{\ell_n,\alpha_n} q_{n+1}$ be a finite path of the system, then we define t as:*

$$t\left(\hat{q} \xrightarrow{\ell_1,\alpha_1} \cdots \xrightarrow{\ell_n,\alpha_n} q_{n+1}\right) \stackrel{\text{def}}{=} \begin{array}{l}(\mathit{enab}(\hat{q}), \ell_1, \mathit{sift}(\alpha_1)) \\ \qquad\vdots \\ (\mathit{enab}(q_n), \ell_n, \mathit{sift}(\alpha_n)).\end{array}$$

Finally, we have all the ingredients needed to define admissible schedulers.

Definition 4 (Admissible schedulers). *A scheduler ζ is admissible if for all $\sigma, \sigma' \in \mathrm{Paths}^\star$*

$$t(\sigma) = t(\sigma') \quad \textit{implies} \quad \zeta(\sigma) = \zeta(\sigma').$$

In this way, admissible schedulers are forced to take the same decisions on paths that they cannot tell apart. Note that this is a restriction on the original definition of (full-information) schedulers where t is the identity map over finite paths (and consequently the scheduler is free to choose differently).

V. Information-hiding properties in presence of nondeterminism

In this section we revise the standard definition of information flow and anonymity in our framework of controlled nondeterminism.

We first consider the notion of adversary. We consider three possible notions of adversaries, increasingly more powerful.

A. Adversaries

External adversaries: Clearly, an adversary should be able, by definition, to see at least the observable actions. For an adversary external to the system S, it is natural to assume that these are also the only actions that he is supposed to see. Therefore, we define the observation domain, for an external adversary, as the set of the (finite) sequences of observable actions, namely:

$$\mathcal{O}_{\mathrm{e}} \stackrel{\text{def}}{=} \Sigma_O^*.$$

Correspondingly, we need a function that extracts the observables from the executions:

$$t_{\mathrm{e}} : \mathrm{Paths}^\star(S) \to \mathcal{O}_{\mathrm{e}}$$

defined as

$$t_{\mathrm{e}}\left(q_0 \xrightarrow{\ell_1,\alpha_1} \cdots \xrightarrow{\ell_n,\alpha_n} q_{n+1}\right) \stackrel{\text{def}}{=} \mathit{sieve}(\alpha_1) \cdots \mathit{sieve}(\alpha_n)$$

where

$$\mathit{sieve}(\alpha) \stackrel{\text{def}}{=} \begin{cases} \alpha & \text{if } \alpha \in \Sigma_O, \\ \epsilon & \text{if } \alpha \in \Sigma_S \cup \{\tau\}. \end{cases}$$

Internal adversaries: An internal adversary may be able to see, besides the observables, also the intearleaving and synchronizations of the various components, i.e. which component(s) are active, at each step of the execution. Hence it is natural to define the observation domain, for an internal adversary, as the sequence of pairs of observable action and tag (i.e. the identifier(s) of the active component(s)), namely:

$$\mathcal{O}_{\mathrm{i}} \stackrel{\text{def}}{=} (L \times (\Sigma_O \cup \{\tau\}))^*.$$

Correspondingly, we need a function that extracts the observables from the executions:

$$t_{\mathrm{i}} : \mathrm{Paths}^\star(S) \to \mathcal{O}_{\mathrm{i}}$$

defined as

$$t_{\mathrm{i}}\left(q_0 \xrightarrow{\ell_1,\alpha_1} \cdots \xrightarrow{\ell_n,\alpha_n} q_{n+1}\right) \stackrel{\text{def}}{=} (\ell_1, \mathit{sieve}(\alpha_1)) \cdots (\ell_n, \mathit{sieve}(\alpha_n)).$$

Note that in this definition we could have equivalently used sift instead than sieve.

Adversaries in collusion with the scheduler: Finally, we consider the case in which the adversary is in collusion with the scheduler, or possibly the adversary *is* the scheduler, like in the Dolev-Yao model. Here the observation domain coincides with the one of the scheduler:

$$\mathcal{O}_{\mathrm{s}} \stackrel{\text{def}}{=} (\mathcal{P}(L) \times L \times (\Sigma_O \cup \{\tau\}))^*.$$

The corresponding function

$$t_{\mathrm{s}} : \mathrm{Paths}^\star(S) \to \mathcal{O}_{\mathrm{s}}$$

is defined as the one of the scheduler, i.e. $t_{\mathrm{s}} = t$.

B. Information leakage

In Information Flow and Anonymity there is a converging consensus for formalizing the notion of leakage as the difference or the ratio between the a priori uncertainty that the adversary has about the secret, and the a posteriori uncertainty, that is, the residual uncertainty of the adversary once it has seen the outcome of the computation. The uncertainty can be measured in different ways. One popular approach is the information-theoretic one, according to which the system is seen as a noisy channel between the secret inputs and the observable output, and uncertainty corresponds to the Shannon entropy of the system (see preliminaries, section B). In this approach, the leakage is represented by the so-called mutual information, which expresses the correlation between the input and the output.

The above approach, however, has been recently criticized by Smith [31], who has argued that Shannon entropy is not suitable to represent the security threats in the typical case in which the adversary is interested in figuring out the the secret in one-try attempt, and he has proposed to use Rényi's min entropy instead, or equivalently, the average probability of succeeding. This leads to interpret the uncertainty in terms of the notion of *vulnerability* defined in the preliminaries, section C. The corresponding notion of leakage, in the pure probabilistic case, have been investigated in [31] (multiplicative case) and in [3] (additive case).

Here we adopt the vulnerability-based approach to define the notion of leakage in our probabilistic and nondeterministic context. The Shannon-entropy-based approach could be extended to our context as well, because in both cases we only need to specify how to determine the conditional probabilities which constitute the channel matrix, and the marginal probabilities that constitute the input and the output distribution.

We will denote by S the random variable associated to the set of secrets $\mathcal{S} = \Sigma_S$, and by O_x the random variables associated to the set of observables $\mathcal{O}_x$, where $x \in \{\mathrm{e}, \mathrm{i}, \mathrm{s}\}$. So, $\mathcal{O}_x$ represents the observation domains for the various kinds of adversaries defined above.

Our results require some structural properties for the system: we assume that there is a single component in the system containing a secret choice and this component contains a single secret choice. This hypothesis is general enough to allow expressing protocols like the Dining Cryptographers, Crowds, voting protocols, etc., where the secret is chosen only once.

Assumption 1. *A system contains exactly one component with a syntactic occurrence of a secret choice, and such a choice does not occur in the scope of any recursive call.*

Note that the assumption implies that the choice appears exactly once in the operational semantics of the component. It would be possible to relax the assumption and allow more than one secret choice in a component, as long as there are no observable actions between the secret choices. But for the sake of simplicity in this paper we impose the more restrictive requirement. As a consequence, we have that the operational semantics of systems satisfies the following property:

Proposition 2. *If $q \xrightarrow{\ell} \mu$ and $q' \xrightarrow{\ell'} \mu'$ are both secret choices, then $\ell = \ell'$ and there exist p_i's, q_i's and q'_i's such that:*

$$\mu = \sum_i p_i \cdot \delta(s_i, q_i) \quad \textit{and} \quad \mu' = \sum_i p_i \cdot \delta(s_i, q'_i)$$

i.e., μ and μ' differ only for the continuation states.

Given a system, each scheduler ζ determines a fully probabilistic automaton, and, as a consequence, the probabilities

$$\mathbf{P}_\zeta\,(s, o) \overset{\text{def}}{=} \mathbf{P}_\zeta \left(\bigcup \left\{ \langle\sigma\rangle \mid \begin{array}{l} \sigma \in \mathrm{Paths}^\star(S), \\ t_x(\sigma) = o, secr(\sigma) = s \end{array} \right\} \right)$$

for each secret $s \in \mathcal{S}$ and observable $o \in \mathcal{O}_x$, where $x \in \{\mathrm{e}, \mathrm{i}, \mathrm{s}\}$. Here *secr* is the map from paths to their secret action. From these we can derive, in standard ways, the marginal probabilities $\mathbf{P}_\zeta\,(s)$, $\mathbf{P}_\zeta\,(o)$, and the conditional probabilities $\mathbf{P}_\zeta\,(o \mid s)$.

We have that the probabilities of the secrets are actually independent from the scheduler:

Proposition 3. *Given a system, for every pair of schedulers ζ and ζ' we have that $\mathbf{P}_\zeta\,(s) = \mathbf{P}_{\zeta'}\,(s)$, for every secret s.*

Because of the previous proposition, we can omit ζ in $\mathbf{P}_\zeta$.

Every scheduler leads to a (generally different) noisy channel, whose matrix is determined by the conditional probabilities as follows:

Definition 5. *Let $x \in \{\mathrm{e}, \mathrm{i}, \mathrm{s}\}$. Given a system and a scheduler ζ, the corresponding channel matrix $\mathcal{C}^x_\zeta$ has rows indexed by $s \in \mathcal{S}$ and columns indexed by $o \in \mathcal{O}_x$. The value in (s, o) is given by $\mathbf{P}_\zeta\,(o \mid s) \overset{\text{def}}{=} \frac{\mathbf{P}_\zeta(s,o)}{\mathbf{P}_\zeta(s)} = \frac{\mathbf{P}_\zeta(s,o)}{\mathbf{P}(s)}$.*

Given a scheduler ζ, the multiplicative leakage can be defined as $\mathcal{L}_\times(\mathcal{C}^x_\zeta, P_S)$, while the additive leakage can be defined as $\mathcal{L}_+(\mathcal{C}^x_\zeta, P_S)$ where P_S is the a priori distribution on the set of secrets (see preliminaries, section C). However, we want a notion of leakage independent from the scheduler, and therefore it is natural to consider the worst case over all possible admissible schedulers.

Definition 6 (x-leakage). *Let $x \in \{\mathrm{e}, \mathrm{i}, \mathrm{s}\}$. Given a system, the multiplicative leakage is defined as*

$$\mathcal{ML}_\times(\mathcal{C}^x_\zeta, P_S) \overset{\text{def}}{=} \max_{\zeta \in Adm} \mathcal{L}_\times(\mathcal{C}^x_\zeta, P_S),$$

while the additive leakage is defined as

$$\mathcal{ML}_+(\mathcal{C}^x_\zeta, P_S) \overset{\text{def}}{=} \max_{\zeta \in Adm} \mathcal{L}_+(\mathcal{C}^x_\zeta, P_S),$$

where Adm is the class of admissible schedulers defined in the previous section.

We have that the classes of observables e, i, and s determine an increasing degree of leakage:

Proposition 4. *Given a system, for the multiplicative leakage we have*

$$\mathcal{ML}_\times(\mathcal{C}^{\mathrm{e}}_\zeta, P_S) \leq \mathcal{ML}_\times(\mathcal{C}^{\mathrm{i}}_\zeta, P_S) \leq \mathcal{ML}_\times(\mathcal{C}^{\mathrm{s}}_\zeta, P_S).$$

Similarly for the additive leakage.

C. Strong anonymity (revised)

We consider now the situation in which the leakage is the minimum for all possible admissible schedules. In the purely probabilistic case, we know that the minimum possible multiplicative leakage is 1, and the minimum possible additive one is 0. We also know that this is the case for all possible input distributions if and only if the capacity of the channel matrix is 0, which corresponds to the case in which the rows of the matrix are all the same. This corresponds to the notion of strong probabilistic anonymity defined in [2]. In the framework of information flow, it would correspond to probabilistic non-interference. Still in [2], the authors considered also the extension of this notion in presence of nondeterminism, and required the condition to hold under all possible schedulers. This is too strong in practice, as we have argued in the introduction: in most cases we can build a scheduler that leaks the secret by changing the interleaving order. We therefore tune this notion by requiring the condition to hold only under the admissible schedulers.

Definition 7 (x-strongly anonymous). *Let $x \in \{\mathrm{e}, \mathrm{i}, \mathrm{s}\}$. We say that a system is x*-strongly-anonymous *if for all admissible schedulers ζ we have*

$$\mathbf{P}_\zeta\,(o \mid s_1) = \mathbf{P}_\zeta\,(o \mid s_2)$$

for all $s_1, s_2 \in \Sigma_S$, and $o \in \mathcal{O}_x$.

The following corollary is an immediate consequence of previous proposition.

Corollary 8.

1) *If a system is s-strongly-anonymous, then it is also i-strongly-anonymous.*
2) *If a system is i-strongly-anonymous, then it is also e-strongly-anonymous.*

The converse of point (2), in previous the corollary, does not hold, as shown by the following example:

Example 3. *Consider the system $S \stackrel{\text{def}}{=} (\{c_1, c_2\})\ P \,||\, Q \,||\, T$ where*

$$P \stackrel{\text{def}}{=} (0.5 : s_1 : \overline{c}_1) + (0.5 : s_2 : \overline{c}_2) \quad Q \stackrel{\text{def}}{=} c_1 : o \quad T \stackrel{\text{def}}{=} c_2 : o.$$

It is easy to check that S is e-strongly anonymous but not i-strongly anonymous, showing that (as expected) internal adversaries can "distinguish more" than external adversaries.

On the contrary, for point (1) of Corollary 8, also the other direction holds:

Proposition 5. *A system is s-strongly-anonymous if and only if it is e-strongly-anonymous.*

VI. On the verification of strong anonymity: a proving technique based on automorphisms

As mentioned in the introduction, several problems involving restricted schedulers have been shown undecidable (including computing maximum/minimum probabilities for the case of standard model checking [18], [17]). These results are discouraging in the aim to find algorithms for verifying strong anonymity/non-interference using our notion of admissible schedulers (and most definitions based on restricted schedulers). Despite the fact that the problem seems to be undecidable in general, in this section we present a sufficient (but not necessary) anonymity proving technique: we show that the existence of automorphisms between pair of secrets implies strong anonymity.

A. The proving technique

In practice proving anonymity often happens in the following way. Given a trace in which user A is the 'culprit', we construct an observationally equivalent trace in which user B is the 'culprit' [20], [16], [23], [21]. This new trace is typically obtained by 'switching' the behavior of users A and B. We formalize this idea by using the notion of automorphism, cf. e.g. [27].

Definition 9 (Automorphism). *Given a TPA $(Q, L, \Sigma, \hat{q}, \theta)$ we say that a bijection $f : Q \to Q$ is an* automorphism *if it satisfies $f(\hat{q}) = \hat{q}$ and*

$$q \xrightarrow{\ell} \sum_i p_i \cdot \delta(\alpha_i, q_i) \Longleftrightarrow f(q) \xrightarrow{\ell} \sum_i p_i \cdot \delta(\alpha_i, f(q_i)).$$

In order to prove anonymity it is enough (but not necessary) to prove that the behaviors of any two 'culprits' can be exchanged without the adversary noticing. We will express this in the Theorem 1 by means of the existence of automorphisms that exchange a given pair of secret s_i and s_j.

Our proving technique requires Assumption 1. Before presenting the main theorem of this section we need to introduce one last definition. Let $S = (C)\ q_1 || \cdots || q_n$ be a system and M its corresponding TPA. We define M_τ as the automaton obtained after "hiding" all the secret actions of M. The idea is to replace every occurrence of a secret s in M by the silent action τ. Note that this can be formalized by replacing the secret choice by a blind choice in the corresponding component q_i of the system S.

We can now state the relation between automorphisms and strong anonymity.

Theorem 1. *Let S be a system that satisfies Assumption 1 and M its tagged probabilistic automaton. If for every pair of secrets $s_i, s_j \in \Sigma_S$ there exists an automorphism f of M_τ such that for any state q we have*

$$q \xrightarrow{\ell, s_i}_M q' \Longrightarrow f(q) \xrightarrow{\ell, s_j}_M f(q'),$$

then S is s-strongly-anonymous.

Note that, since s-strong anonymity implies i-strong anonymity and e-strong anonymity, the existence of such an automorphism implies all the notions of strong anonymity presented in this work.

Proposition 6. *The converse does not hold, i.e. strong anonymity does not imply the existence of automorphisms.*

We now show that the definition of x-strong-anonymity is independent of the particular distribution over secrets, i.e., if a system is x-strongly-anonymous for a particular distribution over secrets, then it is x-strongly-anonymous for all distributions over secrets.

Theorem 2. *Consider a system $S = (C)\ q_1 \parallel \cdots \parallel q_i \parallel \cdots \parallel q_n$. Let q_i be the component which contains the secret choice, and assume that it is of the form $\sum_j p_j : s_j \,.\, q_j$. Consider now the system $S' = (C)\ q_1 \parallel \cdots \parallel q_i' \parallel \cdots \parallel q_n$, where q_i' is identical to q_i except for the secret choice, which is replaced by $\sum_j p_j' : s_j \,.\, q_j$. Then we have that:*

1) *For every s_i, s_j there is an automorphism on S satisfying the assumption of Theorem 1 if and only if the same holds for S'.*
2) *S is x-strongly-anonymous if and only if S' is x-strongly-anonymous.*

Note: 1) *does not imply* 2)*, because in principle neither S not S' may have the automorphism, and still one of the two could be strongly anonymous.*

B. An Application: Dining Cryptographers

Now we show how to apply the proving technique presented in this section to the Dining Cryptographers protocol. Concretely, we show that there exists an automorphism f exchanging the behavior of the Crypt_0 and Crypt_1; by symmetry, the same holds for the other two combinations.

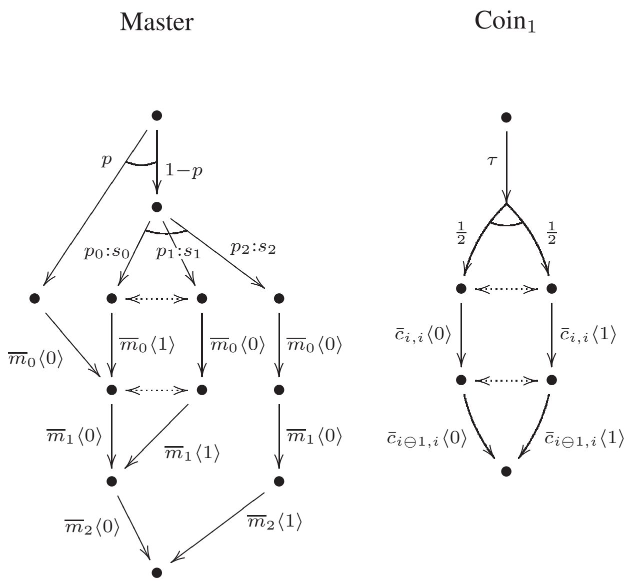

Fig. 5. Automorphism between Crypt$_0$ and Crypt$_1$

Consider the automorphisms of Master and Coin$_1$ indicated in Figure 5. The states that are not explicitly mapped (by a dotted arrow) are mapped to themselves.

Also consider the identity automorphism on Crypt$_i$ (for $i = 0, 1, 2$) and on Coin$_i$ (for $i = 0, 2$). It is easy to check that the product of these seven automorphisms is an automorphism for Crypt$_0$ and Crypt$_1$.

VII. Conclusion and future work

We have defined a class of partial-information schedulers which can only base their decisions on the information they have available. In particular they cannot base their decisions on the internal behavior of the components.

We have used admissible schedulers to resolve nondeterminism in a realistic way, and to tune the definition of strong anonymity proposed in [2].

We have presented a technique to prove the various definitions of strong anonymity proposed in the paper. This is particularly interesting considering that many problems related to restricted schedulers have been shown to be undecidable. In particular we have shown how to use the technique to prove that the DC protocol is strongly anonymous when considering admissible schedulers, in contrast to the situation when considering full-information schedulers.

We plan to investigate the decidability problem for the various definitions of strong anonymity we have proposed. Another interesting direction for future work is to extend well known isomorphism-checking algorithms and tools (see [15] for a survey) to our setting in order to verify automatically strong anonymity (in case an automorphism exists - recall that this is not a necessary condition).

Acknowledgement

The authors wish to thank Flavio Garcia, Pedro D'Argenio, Sergio Giro, and Mariëlle Stoelinga for useful comments on an earlier version of this paper, as well as the anonymous reviewers for thoroughly reading the paper and providing thoughtful recommendations.

References

[1] M. S. Alvim, M. E. Andrés, C. Palamidessi, and P. van Rossum. Safe Equivalences for Security Properties. In *Proc. of TCS*, IFIP AICT, 2010.

[2] M. Bhargava and C. Palamidessi. Probabilistic anonymity. In *Proc. of CONCUR*, volume 3653 of *LNCS*, pages 171–185. Springer, 2005.

[3] C. Braun, K. Chatzikokolakis, and C. Palamidessi. Quantitative notions of leakage for one-try attacks. In *Proc. of MFPS*, *ENTCS* 249, 2009.

[4] 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.

[5] 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*, *LNCS* 4167, pages 238–253, 2006.

[6] K. Chatzikokolakis, G. Norman, and D. Parker. Bisimulation for demonic schedulers. In *Proc. of FOSSACS*, *LNCS* 5504, 2009.

[7] K. Chatzikokolakis and C. Palamidessi. Making random choices invisible to the scheduler. In *Proc. of CONCUR*, *LNCS* 4703, 2007.

[8] K. Chatzikokolakis, C. Palamidessi, and P. Panangaden. Anonymity protocols as noisy channels. *Inf. and Comp.*, 206(2–4):378–401, 2008.

[9] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[10] D. Clark, S. Hunt, and P. Malacaria. Quantified interference for a while language. In *Proc. of QAPL*, *ENTCS* 112, pages 149–166, 2005.

[11] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proc. of DIAU*, volume 2009 of *LNCS*, pages 44–66. Springer, 2000.

[12] M. R. Clarkson, A. C. Myers, and F. B. Schneider. Belief in information flow. *J. of Comp. Security*, 17(5):655–701, 2009.

[13] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. J. Wiley & Sons, Inc., second edition, 2006.

[14] L. de Alfaro, T. A. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In *Proc. of CONCUR*, *LNCS* 2154, 2001.

[15] P. Foggia, C. Sansone, and M. Vento. A performance comparison of five algorithms for graph isomorphism. In *Proc. of the IAPR TC-15 Ws. on Graph-based Repres. in Pattern Recognition*, pages 188–199, 2001.

[16] F. D. Garcia, I. Hasuo, P. van Rossum, and W. Pieters. Provable anonymity. In *Proc. of FMSE*, pages 63–72. ACM, 2005.

[17] S. Giro. Undecidability results for distributed probabilistic systems. In *Proc. of SBMF*, volume 5902 of *LNCS*, pages 220–235. Springer, 2009.

[18] S. Giro and P. R. D'Argenio. Quantitative model checking revisited: Neither decidable nor approximable. In *Proc. of FORMATS*, 2007.

[19] 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*, volume 5710 of *LNCS*, pages 338–353. Springer, 2009.

[20] J. Y. Halpern and K. R. O'Neill. Anonymity and information hiding in multiagent systems. *J. of Comp. Security*, 13(3):483–512, 2005.

[21] I. Hasuo and Y. Kawabe. Probabilistic anonymity via coalgebraic simulations. In *Proc. of ESOP*, *LNCS* 4421, pages 379–394, 2007.

[22] M. Z. Kwiatkowska, G. Norman, and D. Parker. Symmetry reduction for probabilistic model checking. In *Proc. of CAV*, *LNCS* 4144, 2006.

[23] S. Mauw, J. Verschuren, and E. de Vink. A formalization of anonymity and onion routing. In *Proc. of ESORICS*, *LNCS* 3193, 2004.

[24] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[25] R. Milner. *Communicating and mobile systems: the π-calculus*. CUP, 1999.

[26] M. K. Reiter and A. D. Rubin. Crowds: anonymity for Web transactions. *ACM Trans. on Information and System Security*, 1(1):66–92, 1998.

[27] J. J. Rutten. Universal coalgebra: A theory of systems. *Theor. Comp. Sci.*, 249:3–80, 2000.

[28] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, 1995. Tech. Rep. MIT/LCS/TR-676.

[29] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. In *Proc. of CONCUR*, *LNCS* 836, 1994.

[30] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[31] G. Smith. On the foundations of quantitative information flow. In *Proc. of FOSSACS*, volume 5504 of *LNCS*, pages 288–302. Springer, 2009.

[32] P. Syverson, D. Goldschlag, and M. Reed. Anonymous connections and onion routing. In *Proc. of S&P*, pages 44–54, 1997.