Probabilistic and nondeterministic aspects of anonymity

Romain Beauxis, Catuscia Palamidessi

Probabilistic and Nondeterministic Aspects of Anonymity

Romain Beauxis and Catuscia Palamidessi

INRIA Saclay and LIX, École Polytechnique

Abstract. The concept of anonymity comes into play in a wide range of situations, varying from voting and anonymous donations to postings on bulletin boards and sending emails. The protocols for ensuring anonymity often use random mechanisms which can be described probabilistically, while the agents' behavior may be totally unpredictable, irregular, and hence expressible only nondeterministically. Formal definitions of the concept of anonymity have been investigated in the past either in a totally nondeterministic framework, or in a purely probabilistic one. In this paper, we investigate a notion of anonymity which combines both probability and nondeterminism, and which is suitable for describing the most general situation in which the protocol and the users can have both probabilistic and nondeterministic behavior. We also investigate the properties of the definition for the particular cases of purely nondeterministic users and purely probabilistic users. We formulate the notions of anonymity in terms of probabilistic automata, and we describe protocols and users as processes in the probabilistic π-calculus, whose semantics is again based on probabilistic automata. Throughout the paper, we illustrate our ideas by using the example of the dining cryptographers.

1 Introduction

The concept of *anonymity* comes into play in those cases in which we want to keep secret the identity of the agents participating in a certain event. There is a wide range of situations in which this property may be needed or desirable; for instance: voting, anonymous donations, and posting on bulletin boards.

Anonymity is often formulated in a more general way as an information-hiding property, namely the property that a part of information relative to a certain event is maintained secret. One should be careful, though, not to confuse anonymity with other properties that fit the same description, notably *confidentiality* (aka *secrecy*). Let us emphasize the difference between the two concepts with respect to sending messages: confidentiality refers to situations in which the content of the message is to be kept secret; in the case of anonymity, on the contrary, it is the identity of the originator, or of the recipient, that has to be kept secret. Analogously, in voting, anonymity means that the identity of the voter associated with each vote must be hidden, and not the vote itself or the candidate voted for. A discussion about the difference between anonymity and other information-hiding properties can be found in [16].

An important characteristic of anonymity is that it is usually relative to the capabilities of the observer. In general the activity of a protocol can be observed by diverse kinds of observers, differing in the information they have access to. The anonymity property depends critically on what we consider as observables. For example, in the situation of an anonymous bulletin board, a posting by one member of the group is kept anonymous to the other members; however, it may be possible that the administrator of the board has access to some privileged information that may allow him to infer the member who posted it.

In general anonymity may be required for a subset of the agents only. In order to completely define anonymity for a protocol it is therefore necessary to specify which set(s) of members has to be kept anonymous. A further generalization is the concept of *group anonymity*: the members are divided into a number of sets, and it is revealed which one, among the groups, is responsible for an event, but the information as to which particular member has performed the event remains hidden. In this paper, however, we only consider the case of a single group of anonymous users.

Various formal definitions and frameworks for analyzing anonymity have been developed in literature. They can be classified into approaches based on process-calculi and transition systems [33, 30], epistemic logic [37, 16], and "function views" [19]. In this paper, we focus on an approach based on a (probabilistic) process-calculus and on probabilistic automata.

The framework and techniques of process calculi have been used extensively in the area of security, to formally define security properties, and to verify cryptographic protocols. See, for instance, [2, 21, 29, 32, 3]. The common denominator is that the various parties involved in the protocol are specified as concurrent processes and present typically a nondeterministic behavior. In [33, 30], the nondeterminism plays a crucial role in the definition of the concept of anonymity, a definition which is based on the so-called "principle of confusion": a system is anonymous if the set of the possible observable outcomes is saturated with respect to the intended anonymous users. More precisely, if in one computation the *culprit* (the user who performs the action) is i and the observable outcome is o, then for every other agent j there must be a computation where j is the culprit and the observable is still o.

The principle of anonymity described above is elegant and general, however it is limited in that it does not cope with quantitative information. Now, many protocols for anonymity use random mechanisms, see, for example, Crowds [28], Onion Routing [38], and Freenet [12]. The probability distribution of these may be known or become known through statistical experiments. From this knowledge, and the observables, one may be able to differentiate the agents quantitatively, namely to deduce that one agent is more likely (has higher probability) to be the culprit than the others. This means that we do not have perfect anonymity. However the definition of the non-deterministic approach (in which of course the random mechanisms are approximated by nondeterministic mechanisms) may still be satisfied, as long as it is possible for each of the other agents to be the culprit, even with very low probability. In other words, the approach in [33, 30], is based on the membership relation, and it is therefore only able to detect the difference between possible and impossible, not the quantitative differences.

Another advantage in taking into account probabilistic information is that it allows to classify various notions of anonymity according to their strength. See for instance the hierarchy proposed by Reiter and Rubin [28]. In this paper we explore a notion of anonymity which corresponds to the strongest one in [28], namely *beyond suspicion*[1]: from the observables all agents appear equally likely to be the culprit.

A probabilistic notion of anonymity was developed (as a part of a general epistemological approach) in [16]. The approach there is purely probabilistic, in the sense that both the protocol and the users are assumed to act probabilistically. In particular the emphasis is on the probability of the users to be the culprit.

In this work, we take the opposite point of view, namely we assume that we may know nothing about the users. They may be totally unpredictable, and change attitude every time, so that the choice of being the culprit cannot be quantified probabilistically, not even by repeating statistical observations. Namely, it is a typical nondeterministic choice[2]. We regard this as a special case, though: In general, we assume that the behavior of the users may be in part probabilistic and in part nondeterministic. As for the protocol, it may use mechanisms like coin tossing, or random selection of a nearby node, which are supposed to exhibit a certain regularity and obey a probabilistic distribution. On the other hand, also the protocol can behave nondeterministically in part, due, for instance, to the (unpredictable) interleaving of the parallel components.

In summary, we investigate a notion of anonymity which combines both probability and nondeterminism, and which is suitable for describing the general situation in which both the users and the protocol can exhibit a combination of probabilistic and nondeterministic behavior. We also investigate the properties of the definition for the particular cases of purely nondeterministic users and purely probabilistic users.

One of the results of our investigation is that the property of anonymity does not depend on the probabilities of the users (cfr. Section 6.1). We consider this independence to be a fundamental property of a good notion of anonymity. In fact, a protocol for anonymity should be able to guarantee this property for every group of users, no matter what is their probability distribution for being the culprit.

In order to define the notion of probability we need, of course, a model of computation able to express both probabilistic and nondeterministic choices. This kind of systems is by now well established in literature, see for instance the probabilistic automata of [34], and has been provided with solid mathematical foundations and sophisticated tools for verification. For expressing the protocols, we will use the probabilistic asynchronous π-calculus introduced in [17, 27], whose semantics is based on a model similar to [34].

Some of the results of this paper have appeared (without proofs) in [5].

---

[1] To be more precise we should say that *we think that it corresponds to the intended notion of beyond suspicion in* [28]. We cannot prove this correspondence because the definition there is given only informally.

[2] Some people consider nondeterministic choice as a probabilistic choice with unknown probabilities. Our opinion is that the two concepts are different: the notion of probability implies that we can gain knowledge of the distribution by repeating the experiment under the same conditions and by observing the frequency of the outcomes. In other words, from the past we can predict the future. This prediction element is absent from the notion of nondeterminism.

1.1 Plan of the paper

The paper is organized as follows: In Section 2 we recall the nondeterministic approach of [33, 30] to the notion of anonymity. In Section 3 we recall the dining cryptographers' Problem by Chaum [11], which will serve as a running example throughout the paper, and we motivate the necessity of copying with probabilities. In Section 4 we briefly recall some basic notions about probabilistic automata and the probability of events (a more formal and detailed presentation can be found in Appendix A.2). In Section 5 we illustrate the notions and assumptions which are at the basis of our notion of anonymity. In Section 6 we propose our notion of anonymity and we show some of its properties, notably the independence from the probability of the users. In Section 7 we consider the special case of purely nondeterministic users. In Section 8 we consider, on the contrary, purely probabilistic systems, and we prove that, under certain conditions, our definition corresponds to what in literature is known as *conditional anonymity*. In Section 11 we discuss other related work. Finally, in Section 12 we conclude.

2 The nondeterministic approach to anonymity

In this section we briefly recall the approach in [33, 30]. In these works, the actions of a system S are classified into three sets (see Fig. 1):

- A: the actions whose performer is intended to remain anonymous for the observer,
- B: the actions that are intended to be completely visible to the observer,
- C: the actions that are intended to be hidden from the observer.

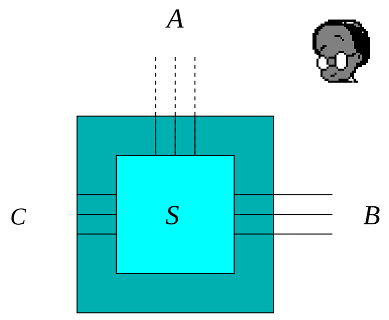

Fig. 1. Classification of the actions in an anonymous system (cfr. [30]).

Typically the set A consists of actions of the form $a(i)$, where a is a fixed "abstract" action (the same for all the elements of A), and i represents the identity of an anonymous user. Hence:

$$A = \{a(i) \mid i \in I\},$$

where I is the set of all the identities of the anonymous users.

Consider a dummy action d (different from all actions in S) and let f be the function on the actions of $A \bigcup B$ defined by $f(\alpha) = d$ if $\alpha \in A$, and $f(\alpha) = \alpha$ otherwise. Then S is said to be (strongly) anonymous on the actions in A if

$$f^{-1}(f(S \backslash C)) \sim_T S \backslash C,$$

where, following the CSP notation [18], $S \backslash C$ is the system resulting from hiding C in S, $f(S')$ is the system obtained from S' by applying the relabeling f to each (visible) action, f^{-1} is the relation inverse of f, and $\sim_T$ represents trace equivalence[3].

Intuitively, the above definition means that for any action sequence $\vec{\alpha} \in A^*$, if an observable trace t containing $\vec{\alpha}$ (not necessarily as a consecutive sequence) is a possible outcome of $S \backslash C$, then, any trace t' obtained from t by replacing $\vec{\alpha}$ with an arbitrary $\vec{\alpha}\,' \in A^*$ must also be a possible outcome of $S \backslash C$.

We now illustrate the above definition on the example of the dining cryptographers.

3 The dining cryptographers' problem

This problem, described by Chaum in [11], involves a situation in which three cryptographers are dining together. At the end of the dinner, each of them is secretly informed by the master whether he should pay the bill or not. So, either the master will pay, or he will ask one of the cryptographers 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. Of course, we assume that the master himself will not reveal this information, and also we want the solution to be distributed, i.e. communication can be achieved only via message passing, and there is no central memory or central 'coordinator' which can be used to find out this information.

A possible solution to this problem, described in [11], is the following: Each cryptographer tosses a coin, which is visible to himself and to the neighbor to his left. Each cryptographer then 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. However, if he is paying then he 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, 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' problem will be a running example throughout the paper.

3.1 Nondeterministic dining cryptographers

In the approach of [33, 30] the dining cryptographers are formalized as a purely nondeterministic system: the coins are approximated by nondeterministic coins, and the choice on who pays the bill is also nondeterministic.

[3] The definition given here corresponds to that in [33]. In [30] the authors use a different (but equivalent) definition: they require $\rho(S \backslash C) \sim_T S \backslash C$ for every permutation ρ in A.

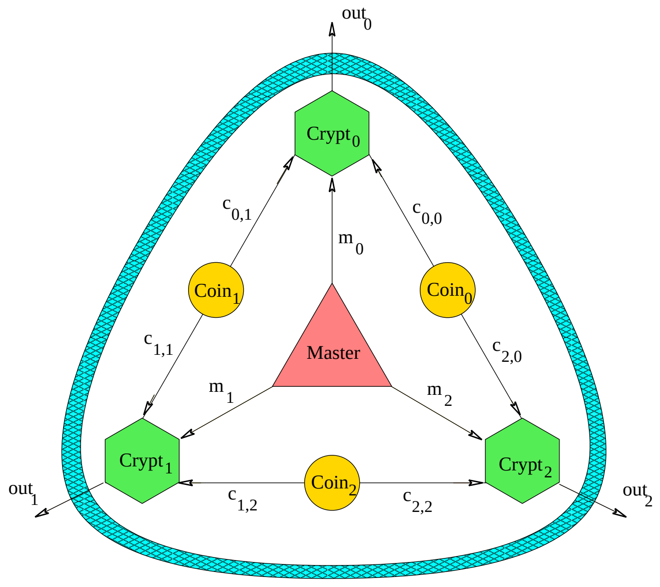

Fig. 2. Chaum's protocol for the dining cryptographers [11, 30].

The specification of the solution can be given in a process calculus style as illustrated below. In the original works [33, 30] the authors used CSP [18]. For the sake of uniformity here we use the π-calculus [25]. We recall that $+$ ($\sum$) is the nondeterministic sum and $|$ (Π) is the parallel composition. 0 is the empty process. τ is the silent (or internal) action. $\overline{c}m$ and $c(x)$ are, respectively, send and receive actions on channel c, where m is the message being transmitted and x is the formal parameter. ν is an operator that, in the π-calculus, has multiple purposes: it provides abstraction (hiding), enforces synchronization, and generates new names. For more details on the π-calculus and its semantics, we refer to Appendix A.1.

In the code below, $\oplus$ and $\ominus$ represent the sum and the subtraction modulo 3. Messages p and n sent by the master are the requests to pay or to not pay, respectively. $\overline{pay}_i$ is the action of paying for cryptographer i.

We remark that we do not need all the expressive power of the π-calculus for this program. In particular, we do not need guarded choice (all the choices are internal because they start with τ), and we do not need neither name-passing nor scope extrusion, thus ν is used just like the restriction operator of CCS [24].

$$
\begin{array}{rl}
Master & = \sum_{i=0}^{2} \tau \,.\, \overline{m}_i \mathsf{p} \,.\, \overline{m}_{i\oplus 1} \mathsf{n} \,.\, \overline{m}_{i\oplus 2} \mathsf{n} \,.\, 0 \\
& \qquad + \;\; \tau . \overline{m}_0 \mathsf{n} \,.\, \overline{m}_1 \mathsf{n} \,.\, \overline{m}_2 \mathsf{n} \,.\, 0 \\
Crypt_i & = m_i(x) \,.\, c_{i,i}(y) \,.\, c_{i,i\oplus 1}(z) \,. \\
& \qquad \text{if } x = \mathsf{p} \\
& \qquad\qquad \text{then } \overline{pay}_i \;.\, \text{if } y = z \\
& \qquad\qquad\qquad \text{then } \overline{out}_i\, disagree \\
& \qquad\qquad\qquad \text{else } \overline{out}_i\, agree \\
& \qquad\qquad \text{else if } y = z \\
& \qquad\qquad\qquad \text{then } \overline{out}_i\, agree \\
& \qquad\qquad\qquad \text{else } \overline{out}_i\, disagree \\
Coin_i & = \tau \,.\, Head_i + \tau \,.\, Tail_i \\
Head_i & = \overline{c}_{i,i} head \,.\, \overline{c}_{i\ominus 1,i} head \,.\, 0 \\
Tail_i & = \overline{c}_{i,i} tail \,.\, \overline{c}_{i\ominus 1,i} tail \,.\, 0 \\
Collect & = out_0(y_0) \,.\, out_1(y_1) \,.\, out_2(y_2) \,.\, \overline{outall}\langle y_0, y_1, y_2 \rangle \\
DCP & = (\nu \vec{c})(\nu \vec{m})(\nu \vec{out})(Master \mid \prod_i Crypt_i \mid \prod_h Coin_h \mid Collect)
\end{array}
$$

Let us consider the point of view of an external observer. The actions that are to be hidden (the set C) are the communications of the decision of the master and the results of the coins ($\vec{m}$, $\vec{c}$). These are already hidden in the definition of the system DCP. The anonymous users are of course the cryptographers, and the anonymous actions (the set A) is constituted by the $\overline{pay}_i$ actions, for $i = 0, 1, 2$. The observable actions (the set B) is constituted by those of the form $\overline{outall}\langle x_0, x_1, x_2 \rangle$ with $x_i \in \{agree, disagree\}$, for $i = 0, 1, 2$.

Let f be the function $f(\overline{pay}_i) = \overline{pay}$ and $f(\alpha) = \alpha$ for all the other actions. It is possible to check that $f^{-1}(f(DCP))) \sim_T DCP$, where we recall that $\sim_T$ stands for trace equivalence. Hence the nondeterministic notion of anonymity, as defined in Section 2, is satisfied.

3.2 Limitations of the nondeterministic approach

As a consequence of approximating the coins by nondeterministic coins, we cannot differentiate between a fair coin and a biased one. However, it is evident that the fairness of the coins is essential to ensure the anonymity property in the system, as illustrated by the following example.

Example 1. Assume that, whenever a cryptographer pays, an external observer obtains *almost always* one of the three outcomes represented in Fig. 3, where a stands for *agree*

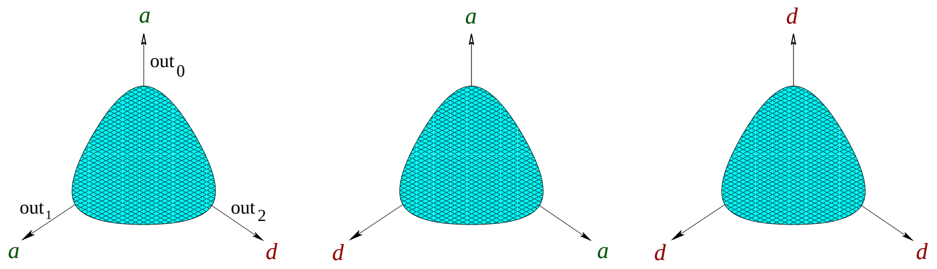

Fig. 3. Illustration of Example 1: the results that are observed with high frequency.

and d for *disagree*. More precisely, assume that these three outcomes appear with a frequency of 33% each, while the missing configuration, d, a, a, appears with a frequency of only 1%. What can the observer deduce? We note that $Crypt_1$ and $Crypt_2$ exhibit the same behavior while $Crypt_0$ behaves differently. Hence, by symmetry considerations ($Coin_0$, $Coin_1$ and $Coin_2$ are opposite to $Crypt_1$, $Crypt_2$ and $Crypt_0$ respectively) he can deduce that $Coin_0$ and $Coin_1$ must be biased in the same way, and $Coin_2$ must be biased differently. More precisely, $Coin_0$ and $Coin_1$ must produce almost always *head*, and $Coin_2$ must produce almost always *tail* (or vice-versa). From this estimation, it is immediate to conclude that, in the first case, the payer is *almost for sure* $Crypt_1$, in the second case $Crypt_2$, and in the third case $Crypt_0$.

In the situation illustrated in the above example, clearly, the system does not provide anonymity. However the nondeterministic definition of anonymity is still satisfied (and it is satisfied in general, as long as "almost always" is not "always", i.e. the fourth configuration d, a, a also appears, from time to time). The problem is that the nondeterministic definition can only express whether or not it is possible to have a particular outcome, but cannot express whether one outcome is more likely than the other.

3.3 Probabilistic dining cryptographers

The probabilistic version of the protocol can be obtained from the nondeterministic one by attaching probabilities to the coins. We wish to remark that this is the essential change with respect to [33,30]: we faithfully model the *random mechanisms* of the protocol as probabilistic, rather than approximate them as nondeterministic.

Concerning the choices of the users (represented in this example as the choice of the master), those are in a sense independent from the protocol, and can be either nondeterministic, or probabilistic, or both.

We use the probabilistic π-calculus (π_p) introduced in [17,27]. The essential difference with respect to the π-calculus is the presence of a *probabilistic choice operator* of the form

$$\sum_i p_i \alpha_i.P_i$$

where the p_i's represents probabilities, i.e. they satisfy $p_i \in [0,1]$ and $\sum_i p_i = 1$, and the α_i's are non-output prefixes, i.e. either input or silent prefixes. (Actually, for the purpose of this paper, only silent prefixes are used.) The detailed presentation of this calculus is in Appendix A.2.

With respect to the program presented in Section 3.1, the definition of the $Coin_i$'s must be modified as follows (p_h and p_t represent the probabilities of the outcomes of the coin tossing):

$$Coin_i = p_h \tau \,.\, Head_i + p_t \tau \,.\, Tail_i$$

It is clear that the system obtained in this way combines probabilistic and nondeterministic behavior, not only because the master may be nondeterministic, but also because the various components of the system and their internal interactions can follow different scheduling policies, selected nondeterministically (although it can easily be seen that this latter form of nondeterminism is not relevant for this particular protocol).

4 Probabilistic automata

The models of computation combining probabilistic and nondeterministic behavior are by now well established in literature, see for instance the probabilistic automata of [34], and have been provided with solid mathematical foundations and tools for verification.

By unfolding a probabilistic automaton we obtain a computation tree, whose nodes, in general, offer both probabilistic and nondeterministic choices. In the probabilistic choices, the arcs are weighted with probabilities. The canonical way of defining the probabilistic notions relevant for our work is the following: First we *solve* the nondeterminism, i.e. we choose a function ς (called *scheduler*) which, for each nondeterministic choice in the computation tree, selects one of the possible alternatives. After pruning the tree from all the non-selected alternatives, we obtain a *fully probabilistic tree*. In such a tree, determined by ς, an *execution* (or *run*) is a maximal path, and an *event* is a (measurable) set of executions. In the finite case, we define the probability of an execution as the product of all the weights in its arcs, and the probability of an event e, $p_\varsigma(e)$, as the sum of the probabilities of the executions in e. For the infinite case, and for more details about the above notions, we refer to Appendix A.2.

It should be clear, from the description above, that in general the probability of an event *depends on the chosen scheduler*. For example, in Fig. 4 the tree P represents a computation tree. From its root there is a nondeterministic choice between two transition groups (aka *steps*), which represent probabilistic choices. We adopt the convention of identifying a step by drawing a curve across its transitions. Analogously, there is a nondeterministic choice between two steps in the fourth node at the level immediately below the root. The trees Q, R and S represent the result of pruning P under different schedulers. Let us denote these schedulers by ς, ϑ and φ respectively. The probability of the event b, under each of these schedulers, is: $p_\varsigma(b) = 1/3 + 1/9 = 4/9$, $p_\vartheta(b) = 1/2$, and $p_\varphi(b) = 0$.

5 Our framework for probabilistic anonymity

In this section we illustrate the notions and assumptions which constitute the basis for our definition of probabilistic anonymity.

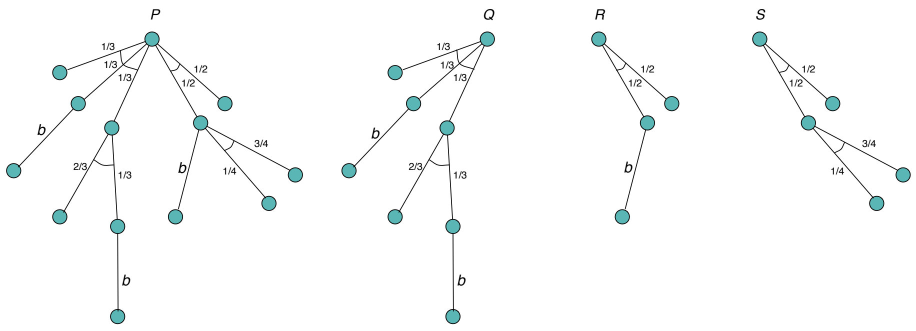

Fig. 4. A computation tree P and the fully probabilistic trees which derive from it under different schedulers. The irrelevant labels are omitted, and the probability is omitted when it is 1.

The system in which the anonymous users live and operate is modeled as a probabilistic automaton M [34], see Appendix A.2. Like in Section 2, we classify the actions of M into the three sets A, B and C, which are determined by the anonymous users, the specific kind of action on which we want anonymity, and the capabilities of the observer:

– The set of the anonymous actions:

$$A = \{a(i) \mid i \in I\}$$

where I is the set of the identities of the anonymous users and a is an injective functions from I to the set of actions which we call *abstract action*.

– The set of the visible actions, B. We will use $b, b', \ldots$ to denote the elements of this set.

– The set of the hidden actions C.

In the following we assume that the actions in C are already restricted in the system, so we do not need to mention them explicitly.

It should be remarked that the term "visible" here is relative: we assume that the observer can see only B and a, but, to the purpose of defining anonymity and checking whether a system is anonymous, we need to leave the actions $a(i)$'s visible (i.e. not restricted) as well.

Differently from [33, 30], we do not assume that the observables are necessarily the traces of visible actions. A trace, indeed, contains not only the information on *which actions* have been executed, but also *in what order* they have been executed. Now, the observer may or may not be able to detect the order, and this is important because the order may give information on the culprit. Another reason for considering such abstraction is to make the analysis simpler. If we know, for instance, that the order of the visible actions does not give any information about the culprit (for instance because we know that the interleaving choices do not depend on the choice of the culprit) then we can forget about it.

In general, we abstract from the (visible) traces by assuming a partition O on them. The observables of the system are then the elements of this partition, denoted by $o, o', \ldots$. Note that each of them is a set of traces.

Another difference from [33, 30] is that we consider the possibility that only certain schedulers are allowed. Thus, our notion of anonymity depends on the set of allowed schedulers, regarded as a parameter. One reason for this choice is to make more efficient the verification of the anonymity property. If we know, for instance, that the interleavings do not give any information about the culprit then we can fix one particular interleaving, thus reducing the number of schedulers to be taken into account.

Definition 1. *An anonymity system is a tuple (M, I, a, O, Z, p), where M is a probabilistic automaton, I is the set of anonymous users, a is the abstract anonymous action, O is a set of observables, Z is a set of schedulers for M, and for every $\varsigma \in Z$, p_ς is a probability measure on the event space generated by the execution tree of M under ς (denoted by $etree(M, \varsigma)$), i.e. the σ–field generated by the cones in $etree(M, \varsigma)$ (cfr. Appendix A.2).*

Note that, as expressed by the above definition, given a scheduler ς, an *event* is a set of executions in $etree(M, \varsigma)$. We introduce the following notation to represent the events of interest:

– $a(i)$: all the executions in $etree(M, \varsigma)$ containing the action $a(i)$
– a : all the executions in $etree(M, \varsigma)$ containing an action $a(i)$ for an arbitrary i
– o : all the executions in $etree(M, \varsigma)$ containing an element of o.

We use the symbols $\cup$, $\cap$ and $\neg$ to represent the union, the intersection, and the complement of events, respectively. By definition of a, we have

$$a = \bigcup_{i \in I} a(i)$$

Furthermore, by definition of O, all the observables are pairwise disjoint:

$$\forall \varsigma \in Z.\, \forall o_1, o_2 \in O.\;\; o_1 \neq o_2 \;\Rightarrow\; p_\varsigma(o_1 \cup o_2) = p_\varsigma(o_1) + p_\varsigma(o_2) \qquad (1)$$

and they cover all possible traces:

$$\forall \varsigma \in Z.\;\; p_\varsigma(\bigcup_{o \in O} o) = 1 \qquad (2)$$

In this paper we assume there is at most one culprit per run. In other words, we assume that all the $a(i)$'s are pairwise disjoint. This assumption is fundamental for the notion of anonymity we propose, and for the results we obtain.

Assumption 1 (At most one culprit)

$$\forall \varsigma \in Z.\, \forall i, j \in I.\;\; i \neq j \;\Rightarrow\; p_\varsigma(a(i) \cup a(j)) = p_\varsigma(a(i)) + p_\varsigma(a(j))$$

6 Probabilistic anonymity for users with probabilistic and nondeterministic behavior

In this section we develop a notion of anonymity for the general case in which also the users, besides the protocol, combine probabilistic and nondeterministic behavior.

Example 2. An example of such kind of behavior in the dining cryptographers is the following: assume the master may have a different attitude depending on the group of cryptographers that meet for dinner. Say that there are two groups, and which of them will meet for dinner is decided nondeterministically. The master will select the payer with probabilities $p_0, \ldots, p_3$ in the case of the first group, and $q_0, \ldots, q_3$ in the case of the second. Note that this situation may be quite common in practice: a certain protocol may be used by different groups of users, which may act probabilistically, but whose probability distribution may vary from one group to the other.

Such a master can be represented in π_p as follows:

$$\begin{aligned}
Master\ &= \tau.Master_1 + \tau.Master_2 \\
Master_1 &= \textstyle\sum_{i=0}^{2} p_i\ \tau\,.\,\overline{m}_i\mathsf{p}\,.\,\overline{m}_{i\oplus 1}\mathsf{n}\,.\,\overline{m}_{i\oplus 2}\mathsf{n}\,.\,0 \\
&\qquad +\ \ p_3\tau.\overline{m}_0\mathsf{n}\,.\,\overline{m}_1\mathsf{n}\,.\,\overline{m}_2\mathsf{n}\,.\,0 \\
Master_2 &= \textstyle\sum_{i=0}^{2} q_i\ \tau\,.\,\overline{m}_i\mathsf{p}\,.\,\overline{m}_{i\oplus 1}\mathsf{n}\,.\,\overline{m}_{i\oplus 2}\mathsf{n}\,.\,0 \\
&\qquad +\ \ q_3\tau.\overline{m}_0\mathsf{n}\,.\,\overline{m}_1\mathsf{n}\,.\,\overline{m}_2\mathsf{n}\,.\,0
\end{aligned}$$

Note that the choice in *Master* is nondeterministic while the choices in $Master_1$ and $Master_2$ are probabilistic.

The notion of anonymity must take into account the probabilities of the $a(i)$'s. When we observe a certain event o, the probability of o having been induced by $a(i)$ must be the same as the probability of o having been induced by $a(j)$ for any other $j \in I$. To formalize this notion, we need the concept of *conditional probability*. Recall that, given two events x and y with $p(y) > 0$, the *conditional probability* of x given y, denoted by $p(x \,|\, y)$, is equal to the probability of x *and* y, divided by the probability of y:

$$p(x \,|\, y) = \frac{p(x \cap y)}{p(y)}$$

We are now ready to propose our notion of anonymity:

Definition 2 (Probabilistic anonymity for users with probabilistic and nondeterministic behavior). *A system* (M, I, a, O, Z, p) *is anonymous if*

$$\begin{aligned}
&\forall \varsigma, \vartheta \in Z.\, \forall i, j \in I.\, \forall o \in O. \\
&\qquad (p_\varsigma(a(i)) > 0 \wedge p_\vartheta(a(j)) > 0) \Rightarrow p_\varsigma(o \,|\, a(i)) = p_\vartheta(o \,|\, a(j))
\end{aligned}$$

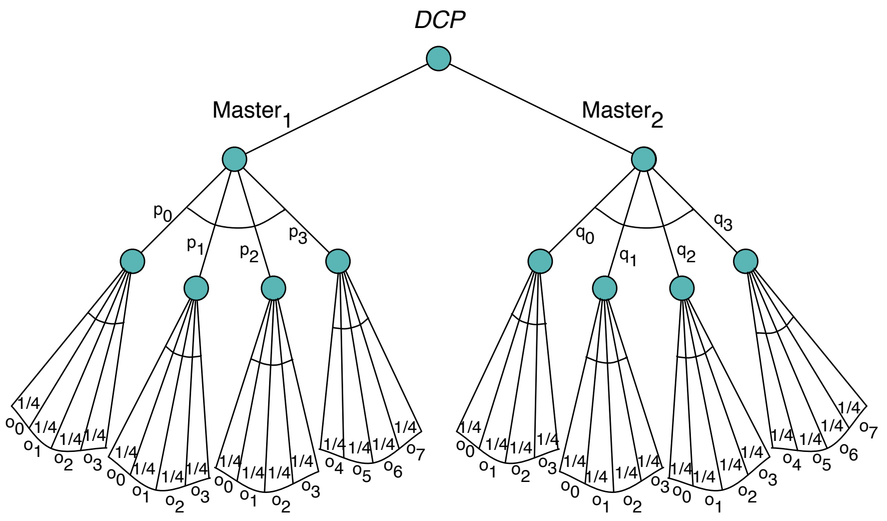

Fig. 5. Illustration of Example 3.

Example 3. Consider the system in Example 2. Assume that the coins are totally fair. For simplicity, let us fix the order of execution of the various components (interleaving)[4], so that the only nondeterministic choice is the choice of the master. Hence we have $Z = \{\varsigma, \vartheta\}$ where ς and ϑ selects $Master_1$ and $Master_2$ respectively. Assume now that $Master_1$ and $Master_2$ select as the payers $i \in I$ with probability p_i and $j \in I$ with probability q_j, respectively. The possible observable events, in both cases, are $o_0 = \langle a, a, d \rangle$, $o_1 = \langle a, d, a \rangle$, $o_2 = \langle d, a, a \rangle$, and $o_3 = \langle d, d, d \rangle$. These are the results in case one of the cryptographers is the payer. The case in which none of them is the payer gives the 4 configurations with an even number of *d*, which we will indicate by $o_4, \ldots, o_7$. Consider now the possible outcomes of the coins. These are 8: $\langle h, h, h \rangle$, $\langle h, h, t \rangle, \ldots \langle t, t, t \rangle$. It is easy to see that, independently from which cryptographer is the payer, each of the above observables is produced by exactly two configurations. If the coins are fair, then, independently from the probability of the selected cryptographer, each observable o corresponds to a cone in the tree (rooted in the node immediately after the selection of the cryptographer) which has probabilistic measure $1/4$ (cfr Fig. 5). Therefore $p_\varsigma(o \,|\, a(i)) = 1/4 = p_\vartheta(o \,|\, a(j))$. Hence Definition 2 is satisfied.

The behavior of a master which combines nondeterministic and probabilistic behavior can be much more complicated than the one illustrated above. However it is easy to see, by following the reasoning in the example above, that as long as the master does not influence the behavior of the coins, and these are fair, the conditional probability of each observable for a given payer is $1/4$.

[4] It is easy to see that, for this example, it does not matter which interleaving we consider.

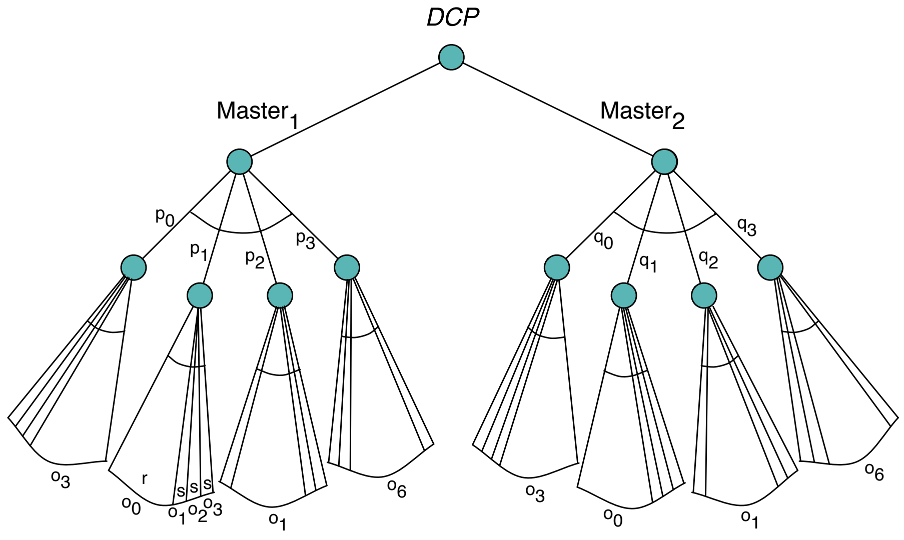

Fig. 6. Illustration of Example 4.

Proposition 1. *Consider the dining cryptographers with arbitrary master (possibly combining nondeterminism and probability). If the coins are fair under every scheduler, then the system is probabilistically anonymous.*

The proof of the above proposition is a straightforward generalization of the Example 3.

Example 4. Consider again the system in Example 2, but assume now that the coins are biased. Say, $Coin_0$ and $Coin_1$ give *head* with probability $9/10$ and *tail* with probability $1/10$, and vice-versa $Coin_2$ gives *head* with probability $1/10$ and *tail* with probability $9/10$. (This situation is analogous to that illustrated in Example 1.) Let us consider the observable $o_0 = \langle a, a, d \rangle$. In case $Crypt_1$ is the payer, then the probability to get o_0 is equal to the probability that the result of the coins is $\langle h, h, t \rangle$, plus the probability that the result of the coins is $\langle t, t, h \rangle$, which is $r = 9/10{*}9/10{*}9/10{+}1/10{*}1/10{*}1/10 = 730/1000$. In case $Crypt_2$ is the payer, then the probability to get $\langle a, a, d \rangle$ is equal to the probability that the result of the coins is $\langle h, h, h \rangle$, plus the probability that the result of the coins is $\langle t, t, t \rangle$, which is $s = 9/10 * 9/10 * 1/10 + 1/10 * 1/10 * 9/10 = 90/1000$. It is easy to see that the same probability holds for the other cryptographers. Fig. 6 illustrates the situation. The observables $o_1, \ldots, o_3$ are as before, o_6 is $\langle a, d, d \rangle$.

Hence, in the biased case, Definition 3 is not satisfied. And this is what we expect, because the system, intuitively, is not anonymous: when we observe $\langle a, a, d \rangle$, $Crypt_1$ is much more likely to be the payer than any of the others.

6.1 Independence from the probability distribution of the users

One important property of Definition 2 is that it is independent from the probability distribution of the users. Intuitively, this is due to the fact that the condition of anonymity simply means that the cones for o rooted in the nodes that result from the (probabilistic and/or nondeterministic) selection of the culprit, have the same probabilistic measure, and this measure is independent from the probability of the culprit.

We consider this property fundamental for a good notion of anonymity: An anonymity protocol, in fact, should guarantee anonymity independently from (the attitude of) the users who use it, as long as they follow the protocol.

Theorem 1. *If (M, I, a, O, Z, p) is anonymous then for any p' which differs from p only on the $a(i)$'s, (M, I, a, O, Z, p') is anonymous.*

Proof. Assume that (M, I, a, O, Z, p) is anonymous. Consider a scheduler $\varsigma \in Z$. It is sufficient to show that for every $o \in O$ and every $i \in I$, $p_\varsigma(o|a(i)) = p'_\varsigma(o|a(i))$, or equivalently

$$\frac{p_\varsigma(o \cap a(i))}{p_\varsigma(a(i))} = \frac{p'_\varsigma(o \cap a(i))}{p'_\varsigma(a(i))}$$

Assume that the choice of the $a(j)'s$ is the first choice in $etree(M, \varsigma)$. If this is not the case, then we can transform the tree into one which satisfy this assumption, and in which the probability of events is the same. Fig. 7 shows the basic step of this transformation, in the case c is an anonymous action. In case c is an observable actions, then in the second tree c and the rightmost b must be switched in order to preserve the order in the trace (in case such order is observable). Let $\varXi$ be the set of runs whose visible traces are of the form $a(i).b_1.b_2.\dots.b_k$ for $b_1.b_2.\dots.b_k \in o$. We have that $p_\varsigma(o \cap a(i)) = p_\varsigma(\varXi)$. Observe now that $p_\varsigma(\varXi) = p_\varsigma(a(i))p_\varsigma(\varXi')$, where $\varXi'$ is the set of suffixes of the runs that start at the node resulting from the choice of $a(i)$. Hence we have $p_\varsigma(o \cap a(i)) = p_\varsigma(a(i))p_\varsigma(\varXi')$. Analogously, $p'_\varsigma(o \cap a(i)) = p'_\varsigma(a(i))p'_\varsigma(\varXi')$. But, because of the hypothesis that p_ς and p'_ς only differ on the $a(i)'s$, we have $p_\varsigma(\varXi') = p'_\varsigma(\varXi')$, which concludes the proof. □

6.2 Alternative characterization

We give here an alternative characterization of the notion of anonymity. The idea is that a system is anonymous if the probability, under a certain scheduler, of an observable o, given $a(i)$, is the same as the probability of o given a under every other scheduler.

Proposition 2. *A system (M, I, a, O, Z, p) is anonymous iff*

$$\forall \varsigma, \vartheta \in Z.\, \forall i \in I.\, \forall o \in O.\; (p_\varsigma(a(i)) > 0 \wedge p_\vartheta(a) > 0) \Rightarrow p_\varsigma(o \,|\, a(i)) = p_\vartheta(o \,|\, a)$$

Proof. **If part)** Let $\varsigma, \vartheta \in Z$ and $i, j \in I$ such that $p_\varsigma(a(i)) > 0$ and $p_\vartheta(a(j)) > 0$. Since $p_\vartheta(a(j)) > 0$ implies $p_\vartheta(a) > 0$, by the hypothesis we have $p_\varsigma(o \,|\, a(i)) = p_\vartheta(o \,|\, a)$. Furthermore, by replacing in the hypothesis ς with ϑ and i with j we have $p_\vartheta(o \,|\, a(j)) = p_\vartheta(o \,|\, a)$.

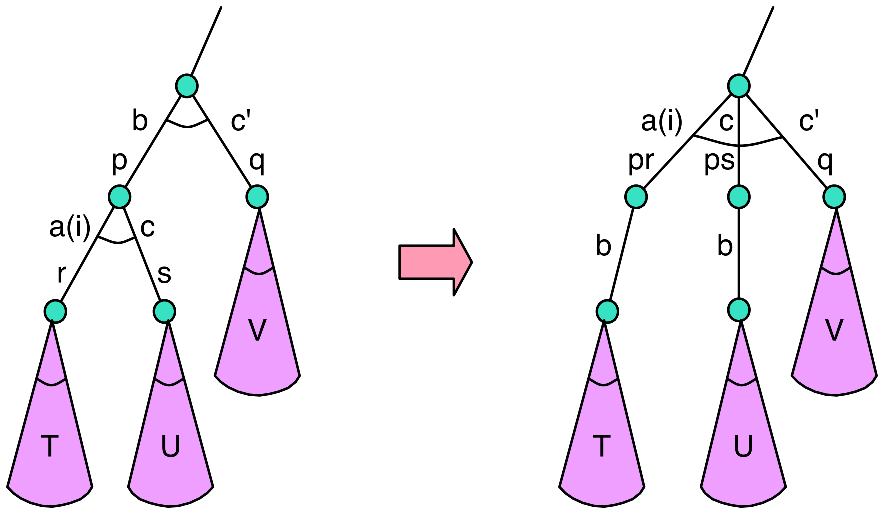

Fig. 7. Basic transformation step for the proof of Theorem 1.

Only if part) Let $\varsigma, \vartheta \in Z$ and $i \in I$ such that $p_\varsigma(a(i)) > 0$ and $p_\vartheta(a) > 0$.

$$\begin{aligned}
p_\vartheta(o \cap a) &= p_\vartheta(o \cap \textstyle\bigcup_{j \in I} a(j)) \\
&= p_\vartheta(\textstyle\bigcup_{j \in I} (o \cap a(j))) \\
&= \textstyle\sum_{j \in I}\ p_\vartheta(o \cap a(j)) && \text{(by Assumption 1)} \\
&= \textstyle\sum_{p_\vartheta(a(j))>0}\ p_\vartheta(o \cap a(j)) \\
&= \textstyle\sum_{p_\vartheta(a(j))>0}\ p_\vartheta(o \,|\, a(j))\ p_\vartheta(a(j)) \\
&= p_\varsigma(o \,|\, a(i))\ \textstyle\sum_{p_\vartheta(a(j))>0}\ p_\vartheta(a(j)) && \text{(by Definition 2)} \\
&= p_\varsigma(o \,|\, a(i))\ p_\vartheta(a)
\end{aligned}$$

Hence $p_\vartheta(o \,|\, a) = p_\vartheta(o \cap a)/p_\vartheta(a) = p_\varsigma(o \,|\, a(i))$. □

7 Probabilistic Anonymity for nondeterministic users

The case in which the users are purely nondeterministic is characterized by the fact that each scheduler determines completely whether an action of the form $a(i)$ takes place or not. Formally:

$$\forall \varsigma \in Z.\, \forall i \in I.\ p_\varsigma(a(i)) = 0 \lor p_\varsigma(a(i)) = 1 \tag{3}$$

In the case of nondeterministic users, the definition of anonymity simplifies as follows:

Proposition 3. *A system* (M, I, a, O, Z, p) *in which the choice of* $a(i)$ *(for* $i \in I$*) is nondeterministic in each run, is anonymous if*

$$\forall \varsigma, \vartheta \in Z.\, \forall o \in O.\; p_\varsigma(a) = p_\vartheta(a) = 1 \;\Rightarrow\; p_\varsigma(o) = p_\vartheta(o)$$

Proof. Immediate by instantiating Definition 2 to the case in which (3) holds, and by observing that, if $p_\varsigma(a(i)) = 1$, then $p_\varsigma(o \,|\, a(i)) = p_\varsigma(o)$. □

8 Probabilistic Anonymity for fully probabilistic systems

In this section we investigate how the removal of the nondeterminism influences our definition of anonymity. We consider therefore purely probabilistic systems. This will allow us also to compare our notion with other probabilistic proposals in literature.

Since the system is totally probabilistic, the probability measures do not depend on the choice of the scheduler. To be more precise, there is only one scheduler. So we can eliminate the component Z from the tuple and we can write $p(x)$ instead of $p_\varsigma(x)$. The definition of probabilistic anonymity given in previous section (cfr. Definition 2) simplifies into the following:

Definition 3. *A fully probabilistic system* (M, I, a, O, p) *is anonymous if*

$$\forall i, j \in I.\, \forall o \in O.\; (p(a(i)) > 0 \wedge p(a(j)) > 0) \Rightarrow p(o \,|\, a(i)) = p(o \,|\, a(j))$$

Furthermore, the alternative characterization in Proposition 2 reduces to the following:

Proposition 4. *A fully probabilistic system* (M, I, a, O, p) *is anonymous iff*

$$\forall i \in I.\, \forall o \in O.\; p(a(i)) > 0 \Rightarrow p(o \,|\, a(i)) = p(o \,|\, a)$$

In the fully probabilistic case there are two other notions of anonymity that seem rather natural. The first is based on the intuition that a system is anonymous if the observations do not change the probability of $a(i)$. The probability of $a(i)$ before the observation is called *a priori* probability; the probability of $a(i)$ after the observation is called *a posteriori* probability. So the intuition is that a protocol does not leak information if these probabilities coincide. This is already known in literature as *conditional anonymity* (cfr. [16]). It is interesting to point out the link with Information Theory: the condition means, indeed, that the *entropy* of the random variable associated to the culprit remains the same after the observation; i.e. the observation does not increase the information about the culprit.

The second notion is based on the (similar) idea that observing o rather than o' should not change our knowledge of the probability of $a(i)$.

It is possible to prove that these two notions are equivalent. Furthermore, if we assume that the action a (i.e. the existence of a culprit) is totally visible to the observer, then we can prove that these notions are equivalent to ours. The condition "a is totally visible" means that every observable o indicates unambiguously whether a has taken place or not, i.e. it either implies a, or it implies $\neg a$. In other words this means that either o (as a set) is contained in a (as a set) or in the complement of a. Formally:

Assumption 2 (The existence of a culprit is observable)

$$\forall o \in O.\ \ p(o \cap a) = p(o) \vee p(o \cap \neg a) = p(o)$$

We prove now our claims of equivalence.

Proposition 5. *Under Assumption 2, the following conditions are equivalent to each other and to our condition of anonymity.*

(i) $\forall i \in I.\, \forall o \in O.\ \ p(o \cap a) > 0 \ \Rightarrow\ p(a(i)\,|\,o) = p(a(i))/p(a)$
(ii) $\forall i \in I.\, \forall o, o' \in O.\ \ (p(o \cap a) > 0 \wedge p(o' \cap a) > 0) \ \Rightarrow\ p(a(i)\,|\,o) = p(a(i)\,|\,o')$.

Proof. We first show the equivalence of (i) and our condition of anonymity, using Proposition 4.

Assume (i) and $p(a(i)) > 0$. If $p(o \cap a) = 0$ then $p(o\,|\,a(i)) = p(o\,|\,a) = 0$. Otherwise,

$$\begin{aligned}
p(o\,|\,a(i)) &= \frac{p(a(i)\,|\,o)\ p(o)}{p(a(i))} \\
&= \frac{p(a(i))\ p(o)}{p(a)\ p(a(i))} && \text{(by (i))} \\
&= \frac{p(o)}{p(a)} \\
&= \frac{p(o \cap a)}{p(a)} && \text{(by Assumption 2))} \\
&= p(o\,|\,a)
\end{aligned}$$

Viceversa, assume that the condition in Proposition 4 holds, and $p(o \cap a) > 0$. If $p(a(i)) = 0$, then $p(a(i)\,|\,o) = p(a(i))/p(a) = 0$. Otherwise we derive $p(a(i)\,|\,o) = p(a(i))/p(a)$ from $p(o\,|\,a(i)) = p(o\,|\,a)$, essentially reverting the above derivation.

As for the equivalence of (i) and (ii), we have:

(i) ⇒ (ii)) Let $i \in I$, and $o, o' \in O$ such that $p(o \cap a) > 0$ and $p(o' \cap a) > 0$. By (i) we have $p(a(i)\,|\,o) = p(a(i))/p(a) = p(a(i)\,|\,o')$.

(ii) ⇒ (i)) Let $i \in I$ and $o \in O$ such that $p(o \cap a) > 0$. We have

$$\begin{aligned}
p(a(i)) &= p(a(i) \cap \textstyle\bigcup_{o' \in O} o') && \text{(by (2))} \\
&= p(\textstyle\bigcup_{o' \in O} (a(i) \cap o')) \\
&= \textstyle\sum_{o' \in O}\ p(a(i) \cap o') && \text{(by (1))} \\
&= \textstyle\sum_{p(o' \cap a) > 0}\ p(a(i) \cap o') \\
&= \textstyle\sum_{p(o' \cap a) > 0}\ p(a(i)\,|\,o')\ p(o') \\
&= p(a(i)\,|\,o)\ \textstyle\sum_{p(o' \cap a) > 0}\ p(o') && \text{(by (ii))} \\
&= p(a(i)\,|\,o)\ p(a) && \text{(by (2) and Assumption 2)}
\end{aligned}$$

□

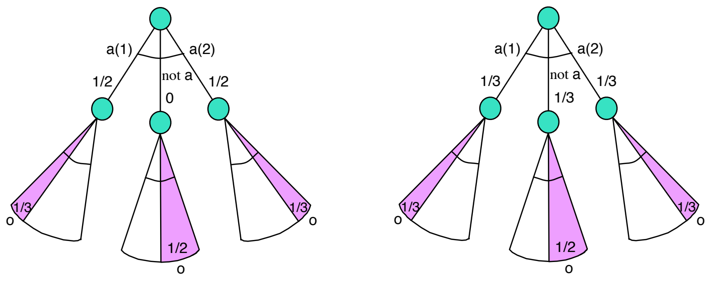

Fig. 8. Illustration of Example 5.

Proposition 5 is based on general properties of probabilistic spaces, independently from the particular setting (in our case, probabilistic automata) which we use to define the probabilities. Similar results have been presented in [15] and in [14] (for the case in which a always occurs, i.e. $p(a) = 1$).

Since the notion of conditional anonymity (as well as the other notion in Proposition 5) is equivalent to ours, we have that also these notions are independent from the probability of the users. On the other hand, one has to keep in mind that the correspondence only holds under the Assumptions 1 (only one culprit) and 2 (the existence of a culprit is observable). Without Assumption 2 conditional anonymity *is not* independent from the probabilities of the users. Without Assumption 1 neither conditional anonymity nor our notion are independent.

Example 5. Fig. 8 shows an example in which Assumption 2 does not hold, and conditional anonymity depends on the probability of the users. In fact the first tree satisfies conditional anonymity, while the second does not. (They both satisfy our notion of anonymity.)

Example 6. Fig. 9 shows an example in which Assumption 1 does not hold, and both conditional anonymity and our notion of anonymity depend on the probability of the users. In fact the first tree satisfies both kinds of anonymity, while the second does not.

9 Conditional anonymity and nondeterminism

It is not clear whether conditional anonymity [16] can be generalized to the case of the users with nondeterministic behavior. The "naive" extensions obtained by introducing the scheduler in the formulae would not work. Let us consider the first characterization, i.e. conditional anonymity (for the other we would follow an analogous reasoning):

$$\forall i \in I.\, \forall o \in O.\;\; p(o \cap a) > 0 \;\Rightarrow\; p(a(i) \,|\, o) = p(a(i))/p(a)$$

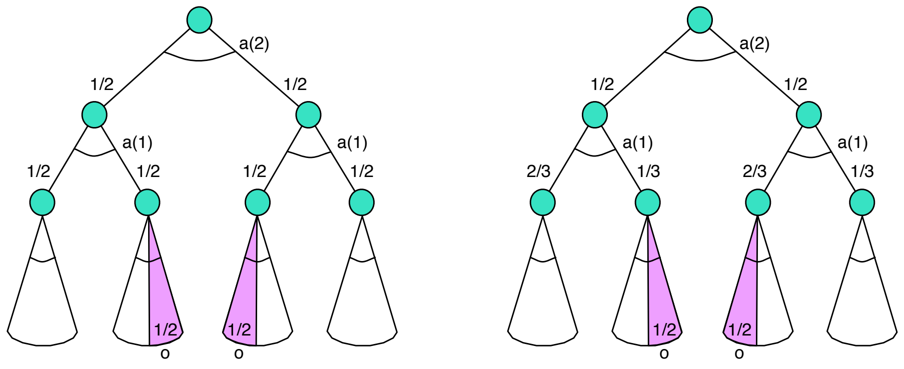

Fig. 9. Illustration of Example 6.

One possible way of reintroducing the notion of scheduler is

$$\forall \varsigma, \vartheta \in Z.\, \forall i \in I.\, \forall o \in O.$$
$$(p_\varsigma(o \cap a) > 0 \wedge p_\vartheta(a) > 0) \Rightarrow\; p_\varsigma(a(i) \,|\, o) = p_\vartheta(a(i))/p_\vartheta(a)$$

However this condition is too strong because it implies that $p_\vartheta(a(i))/p_\vartheta(a)$ is the same for every ϑ, and this is clearly not the case for instance for the nondeterministic and probabilistic master specified in Section 2.

On the other hand, if we weaken the condition by identifying ς and ϑ:

$$\forall \varsigma \in Z.\, \forall i \in I.\, \forall o \in O.\;\; p_\varsigma(o \cap a) > 0 \Rightarrow\; p_\varsigma(a(i) \,|\, o) = p_\varsigma(a(i))/p_\varsigma(a)$$

then the condition would be too weak to ensure anonymity, as shown by the following example:

Example 7. Consider a system in which the master influences the behavior of the coins somehow, in such a way that when $Crypt_i$ is chosen as the payer (say, purely nondeterministically, by ς_i) the result is always $o_0 = \langle d, a, a\rangle$ for $i = 0$, $o_1 = \langle a, d, a\rangle$ for $i = 1$, and $o_2 = \langle a, a, d\rangle$ for $i = 2$. Then we would have $p_{\varsigma_i}(o_j \cap a) > 0$ only if $j = i$, and $p_{\varsigma_i}(a(i) \,|\, o_i) = 1 = p_{\varsigma_i}(a(i))/p_{\varsigma_i}(a)$. Hence the above condition would be satisfied, but the system is not be anonymous at all: whenever we observe $\langle d, a, a\rangle$, for instance, we are sure that $Crypt_0$ is the payer.

10 Toward an automatic probabilistic checker

We have formulated the notion of anonymity in terms of observables for processes in the probabilistic π-calculus, whose semantics is based on the probabilistic automata of [34]. This opens the way to the automatic verification of the property.

It is worth noting that existing probabilistic model checkers, like PRISM, cannot be used directly to prove anonymity. In fact, PRISM computes only the infimum and the supremum of the probabilities of a given formula with respect to all possible schedulers. The anonymity property, on the contrary, requires computing the exact probability for each scheduler.

We are currently developing a model checker using the probabilistic π-calculus. In order to achieve this goal, several practical issues have to be fixed.

An automatic checker for our probabilistic notion of anonymity needs a full evaluation of the execution. This differs from the model checkers, where the process is checked against a single property. In our case, we want to collect all the various outcomes of the protocol and relate them to the inputs.

The natural language for an implementation is the asynchronous version of the π-calculus. In the asynchronous variant of the π-calculus, only internal (blind) choice is allowed and message sending is limited. The sending construct of the synchronous language, $\overline{x}y.P$ is replaced by an atomic sending process $\overline{x}y$. In [4], this limitation is shown to be equivalent to using an unordered buffered communication medium.

A process in the asynchronous π-calculus cannot natively send a message and wait until it is received to continue its execution. Encodings exist that mostly consist in an acknowledgment of the reception, much like for the TCP protocol.

For a majority of applications, the asynchronous π-calculus is the natural language for expressing a protocol or an algorithm, since most real-life communications and modern electronic networks use asynchronous communication mediums. For instance, the implementation of the the dining cryptographers proposed in this paper is purely asynchronous. However, asynchronous communications imply algorithmic limitations, such as for implementing a distributed consensus, as studied in [26].

A naive evaluation of the probabilistic executions can be very inefficient. When communicating asynchronously, most communications happen without a particular order. This generates interleavings, where several sequence of actions can be observed in any order, but the overall result is the same. When evaluating this kind of executions, it is much more efficient to constrain the evaluation to only one of these possible interleavings. This leads to the definition of restricted executions that we call *focused*.

Such focused evaluations have been proved to maintain all relevant informations on the evaluated process. The may testing semantics is equivalent on focused and original executions. Furthermore, two bisimilar processes for the focused executions are bisimilar for the original executions. The implementation of such focused evaluator for the asynchronous π-calculus is currently under development.

11 Related work

The work [19] presents a modular framework to formalize a range of properties (including numerous flavors anonymity and privacy) of computer systems in which an observer has only partial information about system behavior, thereby combining the benefits of the knowledge-based approach (natural specification of information-hiding) and the algebra-based approach (natural specification of system behavior). It proposes the notion of *function view* to represent a mathematical abstraction of partial knowledge of a function. The logical formulae describing a property are characterized as *opaqueness* of certain function views, converted into predicates over observational equivalence classes, and verified, when possible, using the proof techniques of the chosen process formalism.

In [16, 37] epistemic logic is used to characterize a number of information-hiding requirements (including anonymity). In particular, [37] introduces the notion of a *group principal* and an associated model, language and logic to axiomatize anonymity. The main advantage of modal logic is that even fairly complex properties can be stated directly as formulae in the logic. On the other hand, [16] uses a completely semantic approach and provides an appropriate semantic framework in which to consider anonymity. It also propose notions of probabilistic anonymity in a purely probabilistic framework. In particular, it propose a notion of conditional probability (cfr. Definition 4.4 in [16]) which is similar to the first characterization in Proposition 5, if we interpret the formula φ in [16] as the occurrence of the event a. Another notion defined in [16] is that of (strong) Probabilistic Anonymity, which is expressed as the requirement that the a posteriory probabilities of two different anonymous events are the same under every observable, i.e.

$$\forall i, j \in I. \, \forall o \in O. \;\; p(o) > 0 \;\Rightarrow\; p(a(i) \,|\, o) = p(a(j) \,|\, o)$$

This condition seems quite natural at a first sight; however a more careful analysis reveals that it is too strong. In [4] it was indeed proved (see also Appendix A.3) that it is equivalent to our notion of anonymity (for fully probabilistic users) *plus* the condition that the a priori probabilities of two anonymous events are the same, namely:

$$\forall i, j \in I. \;\; p(a(i)) = p(a(j))$$

So Probabilistic Anonymity in the sense of [16] requires a uniform distribution on the anonymous events, which, in our opinion, is too much of a restriction for the users.

The first characterization in Proposition 5 was also implicitly used by Chaum in [11] (in which he considered a purely probabilistic system) as definition of anonymity. The factor $p(a)$ is not present in the formula of Chaum, but that may be a typo, because in the informal explanation he gives, that factor is taken into account.

In literature there have been other works involving the use of variants of the π-calculus for formalizing protocols providing anonymity or similar properties. See for example [1, 20].

The research on the probabilistic foundations of anonymity (and more in general, information-hiding, and the related field of information flow) has recently evolved towards the use of Information Theory [13, 35, 22, 9, 23] and Hypothesis Testing [10, 36]. The advantage in the use of these frameworks is that they allow to define quantitative degrees of anonymity. On the other hand, these framework are purely probabilistic, i.e. assume that the behavior of the user is probabilistic, and do not consider the existence of internal nondeterminism in the protocol.

With respect to the protocol's nondeterminism, our definition of anonymity is quite strict, because it requires that there is no leakage of information under any scheduler. But in the majority of cases, we can define a scheduler that follows a different policy depending on the choice of the anonymous event, thus leaking the information about the anonymous event. In the case of our example of the Dining Cryptographers we do not have such problem, but we would have it if the declarations of the cryptographers were observed as a sequence instead of a tuple. In fact, the scheduler could for instance always schedule the declaration of the payer at last, thus revealing entirely who the payer is. The problem of the scheduler in security, and a proposal for its solution, are discussed in [8] and [7].

12 Conclusion and future work

We have proposed a notion of anonymity based on a model which combines probability and nondeterminism, and we have shown that under certain assumptions it can be regarded as a generalization of conditional anonymity [16].

We have formulated the notion of anonymity in terms of observables for processes in the probabilistic π-calculus, whose semantics is based on the probabilistic automata of [34]. This opens the way to the automatic verification of the property. We are currently developing a model checker for the probabilistic π-calculus.

References

1. Martín Abadi and Cédric Fournet. Private authentication. *Theoretical Computer Science*, 322(3):427–476, 2004.
2. Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 10January 1999.
3. Roberto M. Amadio and Denis Lugiez. On the reachability problem in cryptographic protocols. In *Proceedings of CONCUR 00*, volume 1877 of *Lecture Notes in Computer Science*. Springer, 2000. INRIA Research Report 3915, march 2000.
4. Romain Beauxis, Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Formal approaches to information-hiding (tutorial). In Gilles Barthe and Cédric Fournet, editors, *Proceedings of the Third Symposium on Trustworthy Global Computing (TGC 2007)*, volume 4912 of *Lecture Notes in Computer Science*, pages 347–362. Springer, 2008.
5. Mohit Bhargava and Catuscia Palamidessi. Probabilistic anonymity. In Martín Abadi and Luca de Alfaro, editors, *Proceedings of CONCUR*, volume 3653 of *Lecture Notes in Computer Science*, pages 171–185. Springer, 2005. `http://www.lix.polytechnique.fr/~catuscia/papers/Anonymity/concur.pdf`.
6. Michele Boreale and Davide Sangiorgi. Some congruence properties for π-calculus bisimilarities. *Theoretical Computer Science*, 198(1,2):159–176, 1998.
7. Konstantinon Chatzikokolakis, Gethin Norman, and David Parker. Bisimulation for demonic schedulers. In Luca De Alfaro, editor, *Proceedings of the Twelfth International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2009)*, York, UK, March 2009 2009. To appear.
8. Konstantinos Chatzikokolakis and Catuscia Palamidessi. Making random choices invisible to the scheduler. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *Proceedings of CONCUR'07*, volume 4703 of *Lecture Notes in Computer Science*, pages 42–58. Springer, 2007. `http://www.lix.polytechnique.fr/~catuscia/papers/Scheduler/report.pdf`.

9. Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. *Information and Computation*, 206(2–4):378–401, 2008. `http://www.lix.polytechnique.fr/~catuscia/papers/Anonymity/Channels/full.pdf`.
10. Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. On the bayes risk in information-hiding protocols. *Journal of Computer Security*, 16(5):531–571, 2008.
11. David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.
12. Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *Lecture Notes in Computer Science*, pages 44–66. Springer, 2000.
13. Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In Roger Dingledine and Paul F. Syverson, editors, *Proceedings of the workshop on Privacy Enhancing Technologies (PET) 2002*, volume 2482 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 2002.
14. R.D. Gill, M. van der Laan, and J. Robins. Coarsening at random: Characterizations, conjectures and counterexamples. In D.Y. Lin and T.R. Fleming, editors, *Proceedings of the First Seattle Symposium in Biostatistics*, Lecture Notes in Statistics, pages 255–294. Springer, 1997.
15. P. D. Grunwald and J. Y. Halpern. Updating probabilities. *Journal of Artificial Intelligence Research*, 19:243–278, 2003.
16. Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483–512, 2005.
17. Oltea Mihaela Herescu and Catuscia Palamidessi. Probabilistic asynchronous π-calculus. In Jerzy Tiuryn, editor, *Proceedings of FOSSACS 2000 (Part of ETAPS 2000)*, volume 1784 of *Lecture Notes in Computer Science*, pages 146–160. Springer, 2000. `http://www.lix.polytechnique.fr/~catuscia/papers/Prob_asy_pi/fossacs.ps`.
18. C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
19. Dominic Hughes and Vitaly Shmatikov. Information hiding, anonymity and privacy: a modular approach. *Journal of Computer Security*, 12(1):3–36, 2004.
20. Steve Kremer and Mark D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In Mooly Sagiv, editor, *Programming Languages and Systems – Proceedings of the 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200, Edinburgh, U.K., April 2005. Springer.
21. Gavin Lowe. Casper: A compiler for the analysis of security protocols. In *Proceedings of 10th IEEE Computer Security Foundations Workshop*, 1997. Also in Journal of Computer Security, Volume 6, pages 53-84, 1998.
22. Pasquale Malacaria. Assessing security threats of looping constructs. In Martin Hofmann and Matthias Felleisen, editors, *Proceedings of the 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2007, Nice, France, January 17-19, 2007*, pages 225–235. ACM, 2007.
23. Pasquale Malacaria and Han Chen. Lagrange multipliers and maximum information leakage in different observational models. In Úlfar Erlingsson and Marco Pistoia, editor, *Proceedings of the 2008 Workshop on Programming Languages and Analysis for Security (PLAS 2008)*, pages 135–146, Tucson, AZ, USA, June 8, 2008 2008. ACM.
24. R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.
25. Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–40 & 41–77, 1992. A preliminary version appeared as Technical Reports ECF-LFCS-89-85 and -86, University of Edinburgh, 1989.

26. Catuscia Palamidessi. Comparing the expressive power of the synchronous and the asynchronous π-calculus. In *Conference Record of POPL '97: The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 256–265, Paris, France, 1997.
27. Catuscia Palamidessi and Oltea M. Herescu. A randomized encoding of the π-calculus with mixed choice. *Theoretical Computer Science*, 335(2-3):373–404, 2005. `http://www.lix.polytechnique.fr/~catuscia/papers/prob_enc/report.pdf`.
28. Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
29. A. W. Roscoe. Modelling and verifying key-exchange protocols using CSP and FDR. In *Proceedings of the 8th IEEE Computer Security Foundations Workshop*, pages 98–107. IEEE Computer Soc Press, 1995.
30. Peter Y. Ryan and Steve Schneider. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001.
31. Davide Sangiorgi. π-calculus, internal mobility and agent-passing calculi. *Theoretical Computer Science*, 167(1,2):235–274, 1996.
32. S. Schneider. Security properties and CSP. In *Proceedings of the IEEE Symposium Security and Privacy*, 1996.
33. Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *Proc. of the European Symposium on Research in Computer Security (ESORICS)*, volume 1146 of *Lecture Notes in Computer Science*, pages 198–218. Springer, 1996.
34. Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995. An extended abstract appeared in *Proceedings of CONCUR '94*, LNCS 836: 481-496.
35. Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Roger Dingledine and Paul F. Syverson, editors, *Proceedings of the workshop on Privacy Enhancing Technologies (PET) 2002*, volume 2482 of *Lecture Notes in Computer Science*, pages 41–53. Springer, 2002.
36. Geoffrey Smith. On the foundations of quantitative information flow. In Luca De Alfaro, editor, *Proceedings of the Twelfth International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2009)*, volume 5504 of *Lecture Notes in Computer Science*, pages 288–302, York, UK, March 2009 2009. Springer.
37. Paul F. Syverson and Stuart G. Stubblebine. Group principals and the formalization of anonymity. In *World Congress on Formal Methods (1)*, pages 814–833, 1999.
38. P.F. Syverson, D.M. Goldschlag, and M.G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 1997.

A Appendix

A.1 The π-calculus

We recall here the basic notions about the π-calculus. We choose the variant used in [6, 31], which differs from the standard one because is has a guarded choice instead of the free choice. This is convenient because it will allow to introduce the probabilistic π-calculus, in the next section, in a smoother way.

Let $\mathcal{N}$ be a countable set of *names*, $x, y, \ldots$. The set of prefixes, $\alpha, \beta, \ldots$, and the set of π-calculus processes, $P, Q, \ldots$, are defined by the following abstract syntax:

$$\textit{Prefixes}\ \alpha\ ::=\ x(y)\ \mid\ \bar{x}y\ \mid\ \tau$$

$$\begin{aligned}\textit{Processes}\ P\ ::=\ & \textstyle\sum_i \alpha_i.P_i\ \mid\ \nu x P\ \mid\ P|P \\ & \mid\ !\,P\ \mid\ [x=y]\,P\ \mid\ [x\neq y]\,P\end{aligned}$$

Prefixes represent the basic actions of processes: $x(y)$ is the *input* of the (formal) name y from channel x; $\bar{x}y$ is the *output* of the name y on channel x; τ stands for any silent (non-communication) action.

The process $\sum_i \alpha_i.P_i$ represents guarded (global) choice and it is usually assumed to be finite. We will use the abbreviations **0** (*inaction*) to represent the empty sum, $\alpha.P$ (*prefix*) to represent sum on one element only, and $P + Q$ for the binary sum. The symbols νx, $|$, and ! are the *restriction*, the *parallel*, and the *replication* operator, respectively.

To indicate the structure of a process expression we will use the following conventions: $P_0 \,|\, P_1 \,|\, P_2 \,|\, \ldots \,|\, P_{k-1}$ stands for $(\ldots((P_0 \,|\, P_1) \,|\, P_2) \,|\, \ldots \,|\, P_{k-1})$, i.e. the parallel operator is left associative, and $\alpha_1.P_1 \,|\, \alpha_2.P_2$ stands for $(\alpha_1.P_1)|(\alpha_2.P_2)$, i.e. the prefix operator has precedence over $|$. In all other cases of ambiguity we will use parentheses.

The operators νx and $y(x)$ are x-*binders*, i.e. in the processes $\nu x P$ and $y(x).P$ the occurrences of x in P are considered *bound*, with the usual rules of scoping. The set of the *free names* of P, i.e. those names which do not occur in the scope of any binder, is denoted by $fn(P)$. The *alpha-conversion* of bound names is defined as usual, and the renaming (or substitution) $P\{y/x\}$ is defined as the result of replacing all occurrences of x in P by y, possibly applying alpha-conversion to avoid capture.

In the paper we use also the construct

$$\text{if } x = y \text{ then } P \text{ else } Q$$

This expression is syntactic sugar standing for the process $[x = y]\,P \,|\, [x \neq y]\,Q$.

The operational semantics is specified via a transition system labeled by *actions* $\mu, \mu' \ldots$. These are given by the following grammar:

$$\textit{Actions}\ \mu\ ::=\ xy\ \mid\ \bar{x}y\ \mid\ \bar{x}(y)\ \mid\ \tau$$

Action xy corresponds to the input prefix $x(z)$, where the formal parameter z is instantiated to the actual parameter y (see Rule I-SUM in Table 1). Action $\bar{x}y$ correspond to the output of a free name. The *bound output* $\bar{x}(y)$ is introduced to model *scope extrusion*, i.e. the result of sending to another process a private (ν-bound) name. The bound names of an action μ, $bn(\mu)$, are defined as follows: $bn(\bar{x}(y)) = \{y\}$; $bn(xy) = bn(\bar{x}y) = bn(\tau) = \emptyset$. Furthermore, we will indicate by $n(\mu)$ all the *names* which occur in μ.

In literature there are two definitions for the transition system of the π-calculus which induce the so-called *early* and *late* bisimulation semantics respectively. Here we choose to present the first one. There is no difference between the two for the purposes of our paper.

The rules for the early semantics are given in Table 1. The symbol $\equiv$ used in Rule CONG stands for *structural congruence*, a form of equivalence which identifies "statically" two processes. Again, there are several definition of this relation in literature. For our purposes we do not need a very rich notion, we will just use it to simplify the presentation. Hence we only assume this congruence to satisfy the following:

(i) $P \equiv Q$ if Q can be obtained from P by alpha-renaming, notation $P \equiv_\alpha Q$,
(ii) $P|Q \equiv Q|P$,
(iii) $(P|Q)|R \equiv P|(Q|R)$,
(iv) $(\nu x P)|Q \equiv \nu x(P|Q)$ if $x \notin fv(Q)$,
(v) $!\,P \equiv P\,|\,!\,P$,
(vi) $[x = x]\,P \equiv P$,
(vii) $[x \neq y]\,P \equiv P$, if x is syntactically different from y.

A.2 The probabilistic π-calculus

In this section we recall the definition of the probabilistic π-calculus, π_p, which was introduced in [17]. This calculus was used in [27] to express various randomized algorithms, notably the distributed implementation of the π-calculus with mixed choice. In this paper, we are going to use it as a formalism to express systems of probabilistic anonymous agents.

Probabilistic automata The π_p-calculus is based on the model of probabilistic automata of Segala and Lynch [34], which are able to express both probabilistic and nondeterministic behaviors.

A discrete probabilistic space is a pair (X, p) where X is a finite or countable set and p is a function $p : X \to (0, 1]$ such that $\sum_{x \in X} p(x) = 1$. Given a set Y, we define the sets of all probabilistic spaces on Y as

$$\mathit{Prob}(Y) = \{(X, p) \mid X \subseteq Y \text{ and } (X, p) \text{ is a discrete probabilistic space }\}.$$

Given a set of states S and a set of actions A, a *probabilistic automaton* on S and A is a triple $(S, \mathcal{T}, s_0)$ where $s_0 \in S$ (initial state) and $\mathcal{T} \subseteq S \times \mathit{Prob}(A \times S)$. We call the elements of $\mathcal{T}$ *transition groups* (in [34] they are called *steps*). The idea behind this model is that the choice between two different groups is made nondeterministically and possibly controlled by an external agent, e.g. a scheduler, while the transition within the same group is chosen probabilistically and it is controlled internally (e.g. by a probabilistic choice operator). An automaton in which there is at most one transition group for each state is called *fully probabilistic*.

We define now the notion of execution of an automaton under a *scheduler*, by adapting and simplifying the corresponding notion given in [34]. A scheduler can be seen as a function that solves the nondeterminism of the automaton by selecting, at each moment of the computation, a transition group among all the ones allowed in the present state. Schedulers are sometimes called *adversaries*, thus conveying the idea of an external

I-Sum	$\sum_i \alpha_i.P_i \xrightarrow{x(z)} P_j[z/y]$	$\alpha_j = x(y)$
O/τ-Sum	$\sum_i \alpha_i.P_i \xrightarrow{\alpha_j} P_j$	$\alpha_j = \bar{x}y$ or $\alpha_j = \tau$
Open	$\dfrac{P \xrightarrow{\bar{x}y} P'}{\nu y P \xrightarrow{\bar{x}(y)} P'}$	$x \neq y$
Res	$\dfrac{P \xrightarrow{\mu} P'}{\nu y P \xrightarrow{\mu} \nu y P'}$	$y \notin n(\mu)$
Par	$\dfrac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}$	$bn(\mu) \cap fn(Q) = \emptyset$
Com	$\dfrac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}y} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$	
Close	$\dfrac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}(y)} Q'}{P \mid Q \xrightarrow{\tau} \nu y(P' \mid Q')}$	
Cong	$\dfrac{P \equiv P' \quad P' \xrightarrow{\mu} Q' \quad Q' \equiv Q}{P \xrightarrow{\mu} Q}$	

Table 1. The transition system of the π-calculus.

entity playing “against” the process. For the purpose of this paper, however, we stick to the term “scheduler” in order to avoid confusion with the notion of adversary used in security. We will assume that a scheduler can decide the next transition group depending not only on the current state, but also on the whole history of the computation till that moment, including the random choices made by the automaton.

Given a probabilistic automaton $M = (S, \mathcal{T}, s_0)$, define $tree(M)$ as the tree obtained by unfolding the transition system, i.e. the tree with a root n_0 labeled by s_0, and such that, for each node n, if $s \in S$ is the label of n, then for each $(s, (X, p)) \in \mathcal{T}$, and for each $(\mu, s') \in X$, there is a node n' child of n labeled by s', and the arc from n to n' is labeled by μ and $p(\mu, s')$. We will denote by $nodes(M)$ the set of nodes in $tree(M)$, and by $state(n)$ the state labeling a node n.

A *scheduler* for M is a function ς that associates to each node n of $tree(M)$ a transition group among those which are allowed in $state(n)$. More formally, $\varsigma : nodes(M) \rightarrow Prob(A \times S)$ such that $\varsigma(n) = (X, p)$ implies $(state(n), (X, p)) \in \mathcal{T}$.

The *execution tree* of an automaton $M = (S, \mathcal{T}, s_0)$ under a scheduler ς, denoted by $etree(M, \varsigma)$, is the tree obtained from $tree(M)$ by pruning all the arcs corresponding to transitions which are not in the group selected by ς. More formally, $etree(M, \varsigma)$ is a fully probabilistic automaton $(S', \mathcal{T}', n_0)$, where $S' \subseteq nodes(M)$, n_0 is the root of $tree(M)$, and $(n, (X', p')) \in \mathcal{T}'$ iff $X' = \{(\mu, n') \mid (\mu, state(n')) \in X$ and n' is a child of n in $tree(M)\}$ and $p'(\mu, n') = p(\mu, state(n'))$, where $(X, p) = \varsigma(n)$. If $(n, (X', p')) \in \mathcal{T}'$, $(\mu, n') \in X'$, and $p'(\mu, n') = p$, we will use sometime the notation $n \xrightarrow[p]{\mu} n'$.

An *execution fragment* ξ is any path (finite or infinite) from the root of $etree(M, \varsigma)$. The notation $\xi \leq \xi'$ means that ξ is a prefix of ξ'. If ξ is $n_0 \xrightarrow[p_0]{\mu_0} n_1 \xrightarrow[p_1]{\mu_1} n_2 \xrightarrow[p_2]{\mu_2} \ldots$, the *probability* of ξ is defined as $p(\xi) = \prod_i p_i$. If ξ is maximal, then it is called *execution*. We denote by $exec(M, \varsigma)$ the set of all executions in $etree(M, \varsigma)$.

We define now a probability on certain sets of executions, following a standard construction of Measure Theory. Given an execution fragment ξ, let $C_\xi = \{\xi' \in exec(M, \varsigma) \mid \xi \leq \xi'\}$ (*cone* with prefix ξ). Define $p(C_\xi) = p(\xi)$. Let $\{C_i\}_{i \in I}$ be a countable set of disjoint cones (i.e. I is countable, and $\forall i, j.\ i \neq j \Rightarrow C_i \cap C_j = \emptyset$). Then define $p(\bigcup_{i \in I} C_i) = \sum_{i \in I} p(C_i)$. Two countable sets of disjoint cones with the same union produce the same result for p, so p is well defined. Further, we define the probability of an empty set of executions as 0, and the probability of the complement of a certain set of executions, with respect to the all executions as the complement with respect to 1 of the probability of the set. The closure of the cones (plus the empty set) under countable unions and complementation generates what in Measure Theory is known as a σ-field.

Syntax and transition system of the π_p-calculus We will now illustrate the π_p-calculus. Syntactically, the only difference with respect to the π-calculus is that we do not have the free choice (or mixed guarded choice depending on the presentation), and we have instead the output prefix

$$\bar{x}y.P$$

and the following *probabilistic non-output choice operator*

$$\sum_i p_i \alpha_i . P_i$$

where the p_i's represents positive probabilities, i.e. they satisfy $p_i \in (0, 1]$ and $\sum_i p_i = 1$, and the α_i's are non-output prefixes, i.e. either input or silent prefixes.

Note that the nondeterministic blind choice $\tau.P + \tau.Q$ can be obtained in this calculus by using the parallel operator: it is in fact equivalent to $(\nu x)(\bar{x} \parallel x.P \parallel x.Q)$.

In order to give the formal definition of the probabilistic model for π_p, it is convenient to introduce the following notation: given a probabilistic automaton $(S, \mathcal{T}, s_0)$ and $s \in S$, we write

$$s \, \{ \xrightarrow[p_i]{\mu_i} s_i \mid i \in I \}$$

iff $(s, (\{(\mu_i, s_i) \mid i \in I\}, p)) \in \mathcal{T}$ and $\forall i \in I \; p_i = p(\mu_i, s_i)$, where I is an index set. When I is not relevant, we will use the simpler notation $s \, \{ \xrightarrow[p_i]{\mu_i} s_i \}_i$. We will also use the notation $s \, \{ \xrightarrow[p_i]{\mu_i} s_i \}_{i:\phi(i)}$, where $\phi(i)$ is a logical formula depending on i, for the set $s \, \{ \xrightarrow[p_i]{\mu_i} s_i \mid i \in I \text{ and } \phi(i) \}$.

The operational semantics of a π_p process P is a probabilistic automaton whose states are the processes reachable from P and the $\mathcal{T}$ relation is defined by the rules in Table 2.

The following is an informal explanation of the rules in Table 2.

SUM**:** This rule models the behavior of a choice process: each transition corresponds to the possible execution of an enabled guard α_i and the consequent commitment to the branch P_i. Note that all possible transitions belong to the same group, meaning that the transition is chosen probabilistically by the process itself.

OUT**:** This rule expresses the fact that an output prefix process $\alpha.P$ simply performs the action, and then continues with P.

RES**:** This rule models restriction on channel y: only the actions on channels different from y can be performed and possibly synchronize with an external process. The probability is redistributed among these actions.

OPEN**:** This rule works in combination with CLOSE by signaling that the send action labeling the transition is on a name which is private to the sender.

PAR**:** This rule represents the interleaving of parallel processes. All the transitions of the processes involved are made possible, and they are kept separated in the original groups. In this way we model the fact that the selection of the process for the next computation step is determined by a scheduler. In fact, choosing a group corresponds to choosing a process.

COM**:** This rule models communication by handshaking. The output action synchronizes with all matching input actions of a partner, with the same probability of the input action. The other possible transitions of the partner are kept with the original probability as well. Note that the side condition ensure that all matching inputs are considered. Thanks to alpha-conversion, we can always rewrite a process so that this condition is met.

SUM $\sum_i p_i \alpha_i . P_i \;\{\xrightarrow[p_i]{x_i(z_i)} P'_i\}_i$ $\quad \alpha_i = x_i(y_i)$ and $P'_i = P_i[z_i/y_i]$ or $\alpha_i = \tau$ and $P'_i = P_i$

OUT $\overline{x}y.P \;\{\xrightarrow[1]{\overline{x}y} P\}$

OPEN
$$\frac{P \;\{\xrightarrow[1]{\overline{x}y} P'\}}{\nu y P \;\{\xrightarrow[1]{\overline{x}(y)} P'\}} \qquad x \neq y$$

RES
$$\frac{P \;\{\xrightarrow[p_i]{\mu_i} P_i\}_i}{\nu y P \;\{\xrightarrow[p'_i]{\mu_i} \nu y P_i\}_{i : y \notin fn(\mu_i)}} \qquad \begin{array}{l} \exists i.\ y \notin fn(\mu_i) \text{ and} \\ \forall i.\ p'_i = p_i / \sum_{j : y \notin fn(\mu_j)} p_j \end{array}$$

PAR
$$\frac{P \;\{\xrightarrow[p_i]{\mu_i} P_i\}_i}{P \mid Q \;\{\xrightarrow[p_i]{\mu_i} P_i \mid Q\}_i} \qquad bn(\mu) \cap fn(Q) = \emptyset$$

COM
$$\frac{P \;\{\xrightarrow[1]{\overline{x}y} P'\} \qquad Q \;\{\xrightarrow[p_i]{\mu_i} Q_i\}_i}{P \mid Q \;\{\xrightarrow[p_i]{\tau} P' \mid Q_i\}_{i:\mu_i = x(y)} \cup \{\xrightarrow[p_i]{\mu_i} P \mid Q_i\}_{i:\mu_i \neq x(y)}} \qquad \text{if } \mu_i = x(z) \text{ then } z = y$$

CLOSE
$$\frac{P \;\{\xrightarrow[1]{\overline{x}(y)} P'\} \qquad Q \;\{\xrightarrow[p_i]{\mu_i} Q_i\}_i}{P \mid Q \;\{\xrightarrow[p_i]{\tau} \nu y(P' \mid Q_i)\}_{i:\mu_i = x(y)} \cup \{\xrightarrow[p_i]{\mu_i} P \mid Q_i\}_{i:\mu_i \neq x(y)}} \qquad \text{if } \mu_i = x(z) \text{ then } z = y$$

CONG
$$\frac{P \equiv P' \qquad P' \;\{\xrightarrow[p_i]{\mu_i} Q'_i\}_i \qquad \forall i.\ Q'_i \equiv Q_i}{P \;\{\xrightarrow[p_i]{\mu_i} Q_i\}_i}$$

Table 2. The probabilistic transition system of the π_p-calculus.

CLOSE**:** This rule is analogous to COM, the only difference is that the name being transmitted is private (local) to the sender.

CONG**:** This rule rule says that structurally equivalent processes perform the same transitions.

A.3 Relation between notions of anonymity for probabilistic users

In this appendix we prove the claim we made in Section 11 (and already proved in [4]), namely that the notion of probabilistic anonymity of Halpern and O'Neil (cfr. Definition 4.4 in [16]), i.e.

$$\forall i, j \in I.\, \forall o \in O.\; p(o) > 0 \;\Rightarrow\; p(a(i) \,|\, o) = p(a(j) \,|\, o) \tag{4}$$

is equivalent to our notion of anonymity (for fully probabilistic users) *plus* the condition that the a priori probabilities of two anonymous events are the same, that is:

$$\forall i, j \in I.\; p(a(i)) = p(a(j)) \tag{5}$$

In order to prove the above claim, let us recall our definition of anonymity as given in Definition 3:

$$\forall i, j \in I.\, \forall o \in O.\; (p(a(i)) > 0 \wedge p(a(j)) > 0) \Rightarrow p(o \,|\, a(i)) = p(o \,|\, a(j)) \tag{6}$$

Proposition 6. $(4) \Leftrightarrow (5), (6)$

Proof.

(4**)** $\Rightarrow$ **(**5**))**

$$\begin{array}{rcll} p(a(i)) & = & \sum_o p(a(i) \text{ and } o) & \text{by the disjointness of the anonymous actions} \\ & = & \sum_o p(a(i)|o)\, p(o) & \text{by definition of conditional probability} \\ & = & \sum_o p(a(j)|o)\, p(o) & \text{by (4)} \\ & = & \sum_o p(a(j) \text{ and } o) & \\ & = & p(a(j)) & \end{array}$$

(4**)** $\Rightarrow$ **(**6**))**

$$\begin{array}{rcll} p(o|a(i)) & = & \dfrac{p(a(i)|o)\, p(o)}{p(a(i))} & \text{by Bayes theorem} \\[2ex] & = & \dfrac{p(a(j)|o)\, p(o)}{p(a(i))} & \text{by (4)} \\[2ex] & = & \dfrac{p(a(j)|o)\, p(o)}{p(a(j))} & \text{by (5)} \\[2ex] & = & p(o|a(j)) & \text{by Bayes theorem} \end{array}$$

(4) $\Leftarrow$ (5) , (6))

$$\begin{aligned}
p(a(i)|o) &= \frac{p(o|a(i))\,p(a(i))}{p(o)} && \text{by Bayes theorem}\\
&= \frac{p(o|a(j))\,p(a(i))}{p(o)} && \text{by (6)}\\
&= \frac{p(o|a(j))\,p(a(j))}{p(o)} && \text{by (5)}\\
&= p(a(j)|o) && \text{by Bayes theorem}
\end{aligned}$$

□