Probabilistic Anonymity*

Mohit Bhargava†
Indian Institute of Technology Delhi
Hauz Khas New Delhi
110 016 India
csu01122@cse.iitd.ernet.in

Catuscia Palamidessi
INRIA Futurs and LIX
École Polytechnique, rue de Saclay
91128 Palaiseau, France
catuscia@lix.polytechnique.fr

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 mails. One of the formal definitions of this concept given in literature is based on nondeterminism. In this paper, we investigate a notion of anonymity based on probability theory, and we show that it can be regarded as a generalization of the nondeterministic one. We then formulate this definition in terms of observables for processes in the probabilistic π-calculus, and propose a framework to verify the anonymity property. We illustrate the method 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 agent participating to a certain event. There is a wide range of situations in which this property may be needed or desirable; for instance: delation, 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, to not 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 [6].

An important characteristic of anonymity is that it is usually relative to a particular point of view. In general an event can be observed from various viewpoints - differing in the information they give access to, and therefore, the anonymity property depends on the view from which the event is being looked at (that is the exact information available to the observer). For example, in the situation of electronic bulletin boards, 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 and can determine the member who posted the message(s), either directly or indirectly.

In general anonymity may be required for a subset of the agents only. In order to completely define anonymity for a system 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 of the groups is responsible for an event, but the information as to which particular member has performed the event must be hidden. In this paper, however, we do not consider the notion of group anonymity, we leave it for further work.

Various formal definitions and frameworks for analyzing anonymity have been developed in literature. They can be classified into approaches based on process-calculi ([17, 20]), epistemic logic ([19, 6]), and "function views"

*This work has been partially supported by the Project Rossignol of the ACI Sécurité Informatique (Ministère de la recherche et nouvelles technologies).

†This work has been carried out during the visit of Mohit Bhargava at INRIA, which has been supported by the INRIA/DREI programme for International Internship.

([8]). In this paper, we focus on the approach based on process-calculi.

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, [1, 9, 14, 16, 2]. The common denominator is that the various entities involved in the system to verify are specified as concurrent processes and present typically a nondeterministic behavior. In [17, 20], the nondeterministism plays a crucial role to define the concept of anonymity. More precisely, this approach to anonymity is based on the so-called "principle of confusion": a system is anonymous if the set of the possible outcomes is saturated with respect to the intended anonymous users, i.e. if one such user can cause a certain observable trace in one possible computation, then there must be alternative computations in which each other anonymous user can give rise to the same observable trace (modulo the identity of the anonymous users).

The principle of anonymity described above is very elegant and general, however it has a limitation: if the observer has the means to repeat the experiment and perform statistical analysis, he may be able to deduce certain quantitative information on the system. In particular, he may be able to compute the probability of certain observables and from that infer the probability of the relation between users and observables. Now, the situation of perfect anonymity can be only achieved when, for each observable, all the users have the same probability of having produced it, one cannot infer any information on the agent from the observable. However this condition cannot be expressed in the nondeterministic approach, since the latter is based on set-theoretic notions, and it is therefore only able to detect the difference between possible and impossible (which in the finite case correspond to positive and zero probability respectively). Even the case in which one user has probability close to 1 will be considered acceptable by the definition of anonymity based on nondeterminism, provided that all the other users have positive probability.

Probabilistic information allows to classify various notions of anonymity according to their strength. Reiter and Robin propose the following hierarchy ([13]):

Beyond suspicion The actual user (i.e. the user that performed the action) is not more likely (to have performed the action) than every other user

Probable innocence The actual user has probability less than $1/2$

Possible innocence There is a non trivial probability that another used could have performed the action.

For the reasons explained above, in the nondeterministic approach this hierarchy would collapse into the same notion.

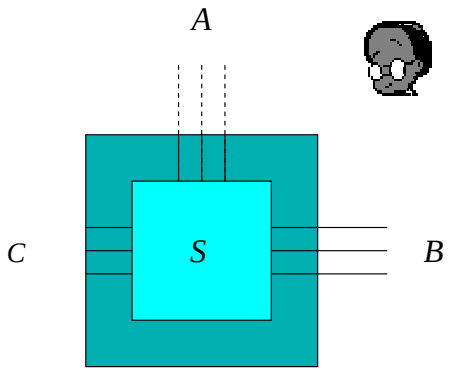

Figure 1. Classification of the actions in an anonymous system

In this paper we explore a notion of anonymity which corresponds to the strongest one above; we leave the others for future investigation.

1.1 Contributions

The contributions of this paper are as follows:

- We propose a new notion of anonymity based on probability theory, and we show that it can be regarded as a generalization of the nondeterministic one given in [17, 20].
- We formulate this definition in terms of observables for processes in the probabilistic π-calculus, and propose a framework to verify anonymity.
- We illustrate the method by using the example of the dining cryptographers.

2 The nondeterministic approach to anonymity

In this section we briefly recall the approach in [17, 20]. In these works, the actions of a system S are classified into three sets (see Figure 1):

- A: the actions that are intended to be known anonymously by the observer,
- B: the actions that are intended to be known completely by the observer,
- C: the actions that are intended to be abstracted (hidden) to the observer.

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. The partition of the remaining actions in B and C determines the "point of view".

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 ([4]), $S\backslash C$ 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[1].

Intuitively, the above definition means that for any action sequence $\vec{\alpha} \in A$, if an observable trace t containing $\vec{\alpha}$ 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 Standard Dining Cryptographers' Problem

This problem was described by Chaum ([5]). The standard dining cryptographers' problem involves a situation in which three cryptographers are dining together, at the end of which, each of them is secretly informed by a central agency (master) whether or not she is paying. Thus, either the master itself is paying, or one of the cryptographers is paying. The cryptographers would like to find out whether it is one of them who is paying, or is it the master who is paying; however, if the payer is one among them, they also wish to maintain anonymity over the identity of the payer. Of course, we assume that the master herself 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, as described in [5], is that each cryptographer tosses a coin, which is visible to herself and her neighbor to the right. Each cryptographer observes the two coins that she can see and announces

[1]The definition given here corresponds to that in [17]. In [20] 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.

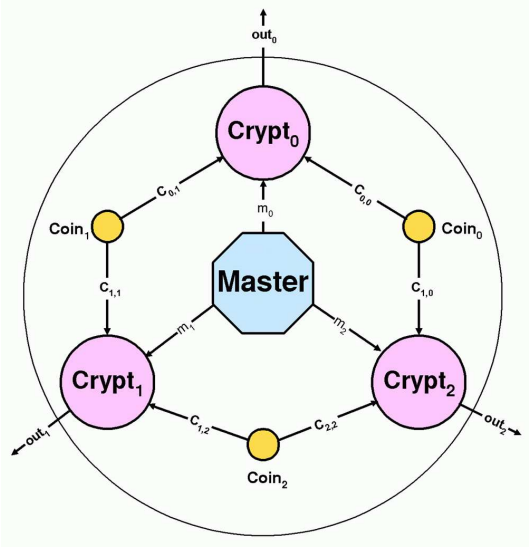

Figure 2. Abstraction of the Standard Dining Cryptographers' Problem

agree or *disagree*. If a cryptographer is not paying, she will announce *agree* if the two sides are same and *disagree* if they are not. However, 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, if one of the cryptographers is paying, the other two cannot identify who exactly has paid from the information available to them.

3.1 Nondeterministic Dining Cryptographers

In this approach to the solution protocol of the Dining Cryptographers' Problem mentioned above, the outcome of the coin tossing and the decision of the master regarding the payment of bill are considered to be nondeterministic.

The nondeterministic specification for the solution protocol of the Dining Cryptographers' Problem can be given in a process calculus style as illustrated below. For the sake of uniformity we use the π-calculus ([11]). We recall that $+$ (Σ) is the nondeterministic sum and $\mid$ (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. More precisely, 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 [10].

$$
\begin{aligned}
Master &= \textstyle\sum_{i=0}^{2} \tau \,.\, \overline{m}_i \mathsf{p} \,.\, \overline{m}_{i\oplus 1} \mathsf{n} \,.\, \overline{m}_{i\oplus 2} \mathsf{n} \,.\, 0 \\
&\quad + \;\; \tau . \overline{m}_0 \mathsf{n} \,.\, \overline{m}_1 \mathsf{n} \,.\, \overline{m}_2 \mathsf{n} \,.\, 0 \\
Crypt_i &= m_i(x) \,.\, c_{i,i}(y) \,.\, c_{i,i\oplus 1}(x) \,. \\
&\qquad \text{if } x = \mathsf{p} \\
&\qquad\quad \text{then } \overline{pay}_i \;. \text{if } y = z \\
&\qquad\qquad \text{then } \overline{out}_i\, agree \\
&\qquad\qquad \text{else } \overline{out}_i\, disagree \\
&\qquad\quad \text{else if } y = z \\
&\qquad\qquad \text{then } \overline{out}_i\, disagree \\
&\qquad\qquad \text{else } \overline{out}_i\, agree \\
Coin_i &= \tau \,.\, Head_i + \tau \,.\, Tail_i \\
Head_i &= \overline{c}_{i,i} head \,.\, \overline{c}_{i\ominus 1,i} head \,.\, 0 \\
Tail_i &= \overline{c}_{i,i} tail \,.\, \overline{c}_{i\ominus 1,i} tail \,.\, 0 \\
DCP &= (\nu \vec{m})(Master \\
&\qquad \mid \; (\nu \vec{c})(\Pi_{i=0}^{2} Crypt_i \;\mid\; \Pi_{i=0}^{2} Coin_i) \;)
\end{aligned}
$$

The actions that are to be hidden (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 (set A) is constituted by the $\overline{pay}_i$ actions, for $i = 0, 1, 2$. The set B is constituted by the actions of the form $\overline{out}_i\, agree$ and $\overline{out}_i\, disagree$, 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 verified

3.1.1 Limitations of the nondeterministic approach

As a result of the nondeterminism, we cannot differentiate between a fair coin and an unfair one. However, it is evident that fairness of coins is essential to ensure the anonymity property in the system. For example, if a cryptographer knows that the coin that is not visible to her is strongly biased to be *head* and since she knows about two coins and can predict the third one with high probability, she can also guess the payer out of the other two (if any) with very high probability. For instance if she observes (say) *head* in both the coins visible to her as well, then the cryptographer who disagrees is most likely to be the payer. As another example, if it is known that two coins are strongly biased to be *head*, and the cryptographer who sees these two coins declares *disagree*, then she in high probability is the one who is paying the bill. This is a *limitation* of the nondeterministic approach. The problem raises from the fact that it can only express whether or not it is possible to have a particular trace, but cannot express whether one trace is more probable than the other.

3.2 The probabilistic π-calculus

In this section we recall the definition of the probabilistic π-calculus, π_p, which was introduced in [7]. This calculus was used in [12] 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.

3.2.1 Probabilistic Automata

The π_p-calculus is based on the model of probabilistic automata of Segala and Lynch ([18]), which are able to express both probabilistic and nondeterministic behaviors.

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

$$
Prob(Y) = \{(X, pb) \mid \begin{array}{l} X \subseteq Y \text{ and } (X, pb) \text{ is} \\ \text{a discrete probabilistic space } \}. \end{array}
$$

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 $\mathcal{T} \subseteq S \times Prob(A \times S)$. We call the elements of $\mathcal{T}$ *transition groups* (in [18] they are called *steps*). The idea is that the choice between two different groups is made nondeterministically, while the transition within the same group is chosen probabilistically.

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 $\mathcal{T} \subseteq S \times Prob(A \times S)$. We call the elements of $\mathcal{T}$ *transition groups* (in [18] 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 [18]. 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 entity playing "against" the process. A process is *robust* with respect to a certain class of adversaries if it achieves its intended result for each possible scheduling imposed by an adversary in the class. Clearly, the reliability of an algorithm depends on how "smart" the adversaries of this class can be. We will assume that an adversary 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 $(s, (X, pb)) \in \mathcal{T}$, 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 $pb(\mu, s')$. We will denote by $nodes(M)$ the set of nodes in $tree(M)$, and by $state(n)$ the state labeling a node n.

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

The *execution tree* of an automaton $M = (S, \mathcal{T}, s_0)$ under an adversary ζ, denoted by $etree(M, \zeta)$, is the tree obtained from $tree(M)$ by pruning all the arcs corresponding to transitions which are not in the group selected by ζ. More formally, $etree(M, \zeta)$ is a fully probabilistic automaton $(S', \mathcal{T}', n_0)$, where $S' \subseteq nodes(M)$, n_0 is the root of $tree(M)$, and $(n, (X', pb')) \in \mathcal{T}'$ iff $X' = \{(\mu, n') \mid (\mu, state(n')) \in X$ and n' is a child of n in $tree(M)\}$ and $pb'(\mu, n') = pb(\mu, state(n'))$, where $(X, pb) = \zeta(n)$. If $(n, (X', pb')) \in \mathcal{T}'$, $(\mu, n') \in X'$, and $pb'(\mu, n') = p$, 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 $etree(M, \zeta)$. 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 $pb(\xi) = \prod_i p_i$. If ξ is maximal, then it is called *execution*. We denote by $exec(M, \zeta)$ the set of all executions in $etree(M, \zeta)$.

We define now a probability on certain sets of executions, following a standard construction of Measure Theory. Given an execution fragment ξ, let $C_\xi = \{\xi' \in exec(M, \zeta) \mid \xi \leq \xi'\}$ (*cone* with prefix ξ). Define $pb(C_\xi) = pb(\xi)$. 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 $pb(\bigcup_{i \in I} C_i) = \sum_{i \in I} pb(C_i)$. Two countable sets of disjoint cones with the same union produce the same result for pb, so pb 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.

3.2.2 Syntax and transition system of the 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.

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 $(s, (\{(\mu_i, s_i) \mid i \in I\}, pb)) \in \mathcal{T}$ and $\forall i \in I\ p_i = pb(\mu_i, s_i)$, 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 1.

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

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

OPEN $\quad \dfrac{P \ \{\xrightarrow[1]{\overline{x}y} P'\}}{\nu y P \ \{\xrightarrow[1]{\overline{x}(y)} P'\}} \quad x \neq y$

RES $\quad \dfrac{P \ \{\xrightarrow[p_i]{\mu_i} P_i\}_i}{\nu y P \ \{\xrightarrow[p'_i]{\mu_i} \nu y P_i\}_{i:y \notin fn(\mu_i)}} \qquad \exists i.\ y \notin fn(\mu_i)$ and $\forall i.\ p'_i = p_i / \sum_{j:y \notin fn(\mu_j)} p_j$

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

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

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

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

Table 1. The probabilistic transition system of the π_p-calculus.

The following is an informal explanation of the rules in Table 1.

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 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.

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.

3.2.3 Probabilistic Dining Cryptographers

We show here how to express in π_p the probabilistic version of the dining cryptographers, which can be obtained from the nondeterministic one by attaching probabilities to the the outcome of the coin tossing and the decision of the master. The probability on actions will induce probabilities on the traces. Thus, we can not only check whether a trace is possible or not, but can also compare their probabilities.

In the specification which follows, the $p_0, \ldots, p_3$ represent the probabilities of the various decisions of the master, while p_h and p_t represent the probabilities of the outcome of the coin tossing.

We remark that, also in this case, we do not use the full power of the π_p-calculus. In particular, we only use probabilistic internal choice (because all the branches are prefixed by τ).

$$
\begin{array}{rcl}
Master & = & \sum_{i=0}^{2} p_i \; \tau \,.\, \overline{m}_i \mathsf{p} \,.\, \overline{m}_{i\oplus 1} \mathsf{n} \,.\, \overline{m}_{i\oplus 2} \mathsf{n} \,.\, 0 \\
 & & \quad + \;\; p_3 \tau . \overline{m}_0 \mathsf{n} \,.\, \overline{m}_1 \mathsf{n} \,.\, \overline{m}_2 \mathsf{n} \,.\, 0 \\
Crypt_i & = & m_i(x) \,.\, c_{i,i}(y) \,.\, c_{i,i\oplus 1}(x) \,. \\
 & & \quad \text{if } x = \mathsf{p} \\
 & & \quad\quad \text{then } \overline{pay}_i \;. \text{if } y = z \\
 & & \quad\quad\quad \text{then } \overline{out}_i agree \\
 & & \quad\quad\quad \text{else } \overline{out}_i disagree \\
 & & \quad\quad \text{else if } y = z \\
 & & \quad\quad\quad \text{then } \overline{out}_i disagree \\
 & & \quad\quad\quad \text{else } \overline{out}_i agree \\
Coin_i & = & p_h \tau \,.\, Head_i + p_t \tau \,.\, Tail_i \\
Head_i & = & \overline{c}_{i,i} head \,.\, \overline{c}_{i\ominus 1,i} head \,.\, 0 \\
Tail_i & = & \overline{c}_{i,i} tail \,.\, \overline{c}_{i\ominus 1,i} tail \,.\, 0 \\
DCP & = & (\nu \vec{m})(Master \\
 & & \quad \mid \; (\nu \vec{c})(\Pi_{i=0}^{2} Crypt_i \;\mid\; \Pi_{i=0}^{2} Coin_i) \;)
\end{array}
$$

4 Probabilistic Anonymity

In this section we propose our notion of probabilistic anonymity.

Also in this case we classify the actions of a system S into the three sets A, B and C of Section 2. As before, these three sets are determined by the set of the anonymous users, the specific type of action on which we want anonymity, and the observer. We only change notation slightly:

- 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*. We also call the pair (I, a) *anonymous action generator*.

- The set of the actions that we observe entirely, B. We will use b, b', ... to denote the elements of this set.

- The set of the hidden actions C.

It should be remarked the the term "observable" here is relative: we assume that the observer can observe only B and a, but, to the purpose of defining anonymity and checking whether a system is anonymous, we need to let the elements of A by visible outcomes of the system. We will come back to this point in Section 4.1.

We assume that a probability p is assigned to the actions of the set A and O.

Definition 1 *An anonymity structure is a tuple (I, a, B, Ξ, p) where (I, a) is an anonymous action generator, B is a set of actions (the observables), Ξ is the set of all possible executions of the system restricted on C (i.e. abstracted from the hidden actions), and p is a probability measure on the event space generated by Ξ (i.e. the sigma space generated by the cones in Ξ, see Section 3.2.1).*

Note that as expressed by the above definition, an *event* is a set of executions. For the sake of simplicity, we will use the following notation to represent some events:

- $a(i)$: the event consisting of all the executions containing the action $a(i)$
- o: the event consisting of all the executions containing as their maximal sequence of observable actions the sequence $o = b_1, b_2, \ldots, b_n$. We will represent by O the set of all the maximal sequences of observable actions.

We will use the symbols $\vee$ and $\wedge$ to represent the union and the intersection of events, respectively. We will say that two events x and y are *incompatible* if $p(x \wedge y) = 0$. We will say that they are *compatible* otherwise.

Since O contains all the maximal sequences of all possible computations, and the o's are disjoint, we have the following result:

Proposition 1

1. $p(O) = 1$
2. $\forall o, o' \in O$, *if* $o \neq o'$ *then* $p(o \vee o') = p(o) + p(o')$.

In order to formalize the notion of anonymity, we will use the concept of *conditional probability*. Recall that, given two events x and y, the *probability* of x given y, denoted by $p(x \mid y)$, is equal to the probability of x *and* y, divided by the probability of y:

$$p(x \mid y) = \frac{p(x \wedge y)}{p(y)}$$

In an anonymous system, it must not be possible to derive any probabilistic information on the anonymous users from the observables of O. This means that, 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 $j \in I$.

Definition 2 *An anonymity structure (I, a, B, Ξ, p) satisfies anonymity (i.e. the system is anonymous wrt the corresponding observer and anonymous users) iff*

$$\forall i, j \in I, \forall o \in O: \quad p(o \mid a(i)) = p(o \mid a(j))$$

This definition of probability may seem a bit surprising at first: one would have expected the roles of $a(i)$ and 0 reversed. I.e.: a system is anonymous if, for every o, the probability of $a(i)$ given o is the same as the probability of $a(j)$ given o. Formally: $\forall i, j \in I, \forall o \in O: \quad p(a(i) \mid o) = p(a(j) \mid o)$. In fact, we had tried this definition first, but we soon realized that it did not work. On the example of the dining cryptographers, for instance, $a(i)$ is $\overline{pay}_i$, and its probability is decided by the master. So in general, independently from the observables, $p(a(i))$ and $p(a(j))$ are different.

Note the analogy between Definition 2 and the definition in Section 2 for the non-deterministic case. The latter claims that for a system to be anonymous (in the non-deterministic sense) over events $a(i) \in A$, if, whenever there is a trace containing $o \in O$ and $a(i)$, then there should be a trace with the same observable part o and an action $a(j)$ for every other anonymous actions $a(j) \in A$. If we include probabilities to the events in this definition, then we have two possible extensions, that are the one in Definition 2 and the (wrong) one in the paragraph which follows the definition. .

We give now some sufficient criterion to prove anonymity:

Proposition 2 *Given an anonymity structure (I, a, B, Ξ, p), the following three properties are equivalent, and each of them implies anonymity:*

1. $\forall i \in I, \forall o, o' \in O: \quad p(a(i) \mid o) = p(a(i) \mid o')$.
2. $\forall i \in I, \forall o \in O: \quad p(a(i) \mid o) = p(a(i))$.
3. $\forall i \in I, \forall o, o' \in O: \quad p(a(i) \wedge o) = p(a(i))p(o)$.

Proof: *We show that* 1 *implies* 2. *The other statements are immediate.*

$$\begin{array}{lcll} p(a(i)) & = & p(a(i) \wedge O) & \textit{(Proposition 1.1)} \\ & = & \sum_{o \in O} p(a(i) \wedge o) & \textit{(Proposition 1.2)} \\ & = & \sum_{o \in O} p(a(i) \mid o)p(o) & \textit{(def of cond. prob.)} \\ & = & p(a(i) \mid o) \sum_{o \in O} p(o) & \textit{(hypothesis 1)} \\ & = & p(a(i) \mid o) & \textit{(Proposition 1.1)} \end{array}$$

□

4.1 Application to Dining Cryptographers' Problem

In the probabilistic Dining Cryptographers described in Section 3.2.3, the events $a(i)$'s are the actions $\overline{pay}_i$. The observable events in B are $\overline{out}_i agree$ and $\overline{out}_i disagree$, and the sequences in O are all those of the form $\alpha_0, \alpha_1, \alpha_2$ where $\alpha_i = \overline{out}_i agree$ or $\alpha_i = \overline{out}_i disagree$.

It is worth noting that this system is both nondeterministic and probabilistic. However, the nondeterministic component here does not play any role, because for each schedule we get always the same result (traces), modulo the order of the visible actions. Hence we will simply assume that we have fixed a scheduler, and we will cope with the system as if it were a purely probabilistic automaton.

In general, for analyzing systems which are both nondeterministic and probabilistic, one should enrich the definition of anonymity given in previous section by requiring that the same property holds with respect to any scheduling policy.

We now prove that the probabilistic Dining Cryptographers satisfy anonymity. According to the definition in previous section, in order to determine the anonymity of a given system, it is required to compute $p(o \mid a(i))$. If this value is same $\forall i \in I$, then the system is anonymous. We could do it by examining the probability of all the computations, but we prefer to introduce a new method of proof which will turn out to be useful also in the rest of the paper.

In the dining cryptographers' problem, we can observe a relationship between a trace consisting of $a(i)$ and o's and the set of results of the coins. We will call the latter *key-configuration*. More precisely, a *key -configuration* is a particular assignment of values (0 or 1) to all the edges in the graph (keys). Given $i \in I, o \in O$, $C(i,o)$ represents the set of all key-configurations which are compatible with the simultaneous occurrences of $a(i)$ and o.

It turns out that the value of $p(a(i)|o)$ is determined by the key-configurations:

Lemma 1 $\forall i \in I, \forall o \in O: \quad p(a(i)|o) = p(C(i,o)|o)$

Proof: In the general dining cryptographers' problem (graph) with n cryptographers (nodes) and m keys (edges), the total number of key configurations is 2^m. If an observable set of events is known to the observer (o), then some of these configurations can be ruled out. For examples, some key-configurations would correspond to the case in which more than one cryptographer inverts her output - these can be eliminated immediately.

If the observable set of events o (a set of outputs from all the cryptographers) is known, such that it is determined by their binary sum that one of them is paying, then by assuming one of the cryptographers to be the payer, we find the possible key-configuration(s). We combine all the satisfying key-configuration(s) for each potential payer and obtain a one-to-one relationship between a set of key-configuration(s) and a potential payer. This implies that, given the observable set of events o , whenever $a(i)$ occurs, $C(i,o)$ also occurs, and vice versa. In fact, this is how we defined $C(i,o)$. Thus, $p(a(i)|o)$ will be equal to $p(C(i,o)|o)$. □

Extension 1: Since at most one cryptographer inverts his output (as the situation can have only one payer), each possible key-configuration can correspond to at most one payer. This implies the following:
$\forall i,k \in I, \forall o \in O$, if $i \neq k$ then $C(i,o)$ and $C(k,o)$ are disjoint.

Extension 2: If $C(i,o)$ and o occurs then $a(i)$ must also occur. However, $a(i)$ can occur for other values of $C(i,o)$ or o as well. In order to equate the probabilities we would need to consider $a(i)$ occurring either with $C(i,o)$ or o. Thus, we get relations: $p(C(i,o) \wedge o) = p(C(i,o) \wedge e_i)$ or $p(C(i,o) \wedge o) = p(a(i) \wedge o)$. It can be noted that these relations also verify the main lemma result that $p(a(i) \wedge o) = p(C(i,o) \wedge o)$.

4.1.1 Determination of Anonymity

To determine anonymity in the general dining cryptographers' problem with possibly unfair coins, we needed to evaluate $\forall i \in I, \forall o \in O: \quad p(o|a(i))$. However, from the above lemma and since the probability of each key-configuration can be computed using (known) information on the probability of individual keys, we have shifted the task of evaluating $p(a(i)\,|o)$ to determining the set of key-configuration(s) corresponding to the particular payer $(a(i))$, given that o occurs.

This can be achieved by first selecting a node to be the potential payer; and then making cuts on the edges of the graph (dividing it into 2 parts) and resolving the 2 subgraphs, for all the permutations of the values on the cut-edges. This is done recursively, till complete keys-configuration(s) have been evaluated. It must be noted that, based on the overall sum of the outputs and the sum of the keys on the cut-edges, it can be determined in which of the 2 subgraph the payer lies in. Thus in each cut, we select the key-values of the cut-edges such that their sum indicates the payer to be in the same subgraph as that of the node which is considered to be the potential payer.

Therefore, for the dining cryptographers' problem, the values of $p(a(i)|o)$ can be computed as described in the forgoing. If this value is same $\forall o \in O$, then according to Proposition 2 in section 4, the system is anonymous.

4.1.2 Case of Fair Coins

The previous section provides a generic procedure to determine anonymity in the general dining cryptographers' problem. However, if we consider the coins (keys) to be fair, then we observe some useful properties.

Lemma 2 *If the coins (keys) are fair, then considering the number of cryptographers is* n $\forall i \in I, \forall o \in O$: $p(C(i,o)|C(o)) = 1/n$
where $C(o)$ *represents a set of key-configurations as a function of* $o \in O$*. such that all the elements of the set are compatible with o.*

Proof: First consider a simple case in which the cryptographers are arranged such that the corresponding nodes are in the form of a tree (that is, the corresponding graph is acyclic). If the outputs for each cryptographer are known to the observer (such that one of them is the payer), then there is exactly one configuration of keys possible for each potential payer. Also, since all the coins (keys) are known to be fair, the probability of each key-configuration is the same. Thus the probability of configuration(s) corresponding to each potential payer is equal and we get $p(C(i,o)|C(o)) = 1/n$. □

To illustrate the fact that given the outputs from all the cryptographers, there is only one configuration possible for each potential payer in an acyclic arrangement (with fair coins), consider the example in figure 3. The outputs from each node (o) are known. Let us evaluate the possible configurations for E $(a(i))$ to be the payer.

We start the analysis from the leaf-nodes of the tree. A leaf has one unknown incident edge - while the output of the node, and whether or not she is the payer is known. Consider *inversion* of a node to be 1 if it is the payer, else 0; and *edge-value* to be 1 if the corresponding coin is head, else 0 if tail. Thus, we can evaluate the unknown incident edge from the rule:

$output = (\sum \textit{edge-values} + inversion) mod\, 2$

After evaluating the incident edges on all the leaves $(D, E, F$ in figure 3, with incident edges $0, 0, 1$ respectively), we exclude them from further analysis. We move toward the root from each of these nodes and consider the next set of nodes (B, C in our example). Again, for each of these nodes, the incident edges is not known, but the originating edges (to its children) and whether or not she is the payer is known. We use the same rule and determine the incident edge ($1, 1$ for nodes B, C respectively). We thus continue moving upward, each time determining the incident edge on the nodes, till we reach the root of the tree. At the root, we can verify that the rule holds (as in our example, output of A is 0, and the sum of its coins (keys) and inversion is also 0).

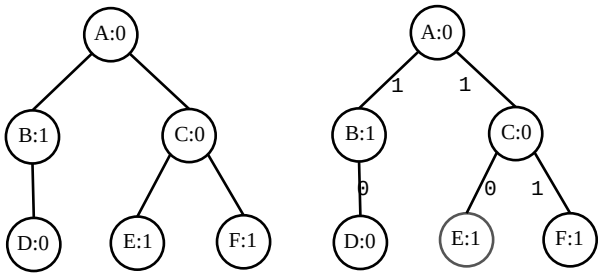

Figure 3. Determination of key-configurations in an acyclic graph

This analysis is more complex in the situation where the cryptographers are arranged in a manner such that the corresponding graph has cycles. However, in order to determine $p(C(i,o) \mid C(o))$ for a particular payer we again evaluate the number of key-configurations, corresponding to the payer, and divide it by the number of configurations for all possible payers.

It is evident that if there is one extra edge, resulting in a single cycle, then by considering specific values (0 and 1) for the edge, we can remove it from the graph (ensuring connectivity of the rest of the graph) and evaluate the possible configurations in the two resulting acyclic subgraphs. That is, if we consider the extra edge to be 0 (tail), then the remaining graph will be acyclic and the analysis (as specified above) for acyclic graphs will return one possible configuration for each node to be the payer. By the same analysis, we will get another (different) configuration for each node, if the extra edge is considered to be 1(head). Thus, the number of configurations for each node to be the payer will double up uniformly. That is, $p(C(i,o)|C(o) = 2/2n = 1/n$.

We can generalize this for k extra edges. k extra edges mean 2^k possible permutations of 1s and 0s, and thus as many acyclic subgraphs (or distinct key-configurations). Here again, $p(C(i,o)|C(o) = 2^k/2^k n = 1/n$. It must be noted that for n connected nodes with no cycles, there will be $n-1$ edges. Since all the coins (keys) are fair, we can randomly declare edges as 'extra' (provided connectivity is maintained) till $n-1$ edges are remaining.

4.1.3 Dependence on Fairness of Coins

Main Result: In the general dining cryptographers' problem, if coins (keys) are fair then any possible arrangement (key sharing) of the diners will maintain anonymity of the payment of bill (from the point of view of a cryptographer, or an external observer) provided the corresponding graph is connected.

Proof: Using lemma 2 (page 10) and the extensions of lemma 1 (page 9) we can show that any arrangement of diners satisfies the sufficient condition of anonymity for the probabilistic case (Proposition 2). This condition states that a system is anonymous iff

$$\forall i \in I, \forall o, o' \in O: \quad p(a(i) \mid o) = p(a(i) \mid o')$$

$\forall i \in I, o, o' \in I_B$, we have the following:

$$\begin{array}{rcll} p(a(i) \mid o) & = & \frac{p(a(i) \wedge o)}{p(o)} & \text{(standard)} \\ & = & \frac{p(a(i) \wedge C(i,o)))}{p(o)} & \text{(extension 2)} \\ & = & \frac{p(a(i)) \cdot p(C(i,o))}{p(o)} & \text{(independence)} \end{array}$$

$$\begin{array}{rcll} p(o_j) & = & \sum_i p(o \wedge a(i)) & \text{(standard)} \\ & = & \sum_i p(C(i,o) \wedge a(i)) & \text{(extension 2)} \\ & = & \sum_i (p(C(i,o)) \cdot p(a(i))) & \text{(independence)} \\ & = & \sum_i (p(C(o)) \cdot \frac{1}{n} \cdot p(a(i))) & \text{(lemma 2)} \\ & = & p(C(o)) \cdot \frac{1}{n} & \text{(standard)} \end{array}$$

$$\begin{array}{rcll} p(a(i) \mid o) & = & \frac{p(a(i)) \cdot p(C(i,o))}{p(C(o)) \cdot \frac{1}{n}} & \text{(from above)} \\ p(a(i) \mid o) & = & p(a(i)) & \text{(lemma 2)} \\ p(a(i) \mid o') & = & p(a(i)) & \text{(from above)} \\ p(a(i) \mid o) & = & p(a(i) \mid o') & \text{(Definition 2)} \end{array}$$

□

We have shown that for fair coins (keys), any possible arrangement (key sharing) of the diners will maintain anonymity provided the corresponding graph is connected. However if, in a given arrangement of diners there is an unfair key, then it may still be possible to directly show that it is anonymous if the edge corresponding to the unfair key forms a part of a cycle. As explained in section 4.1.1, if a key is known to the observer, then it must be removed from the graph, similarly, in the probabilistic case, we remove unfair keys from the graph.

Thus, if an arrangement of diners in the general dining cryptographers' problem have unfair key(s), the corresponding edges from the graph must be removed. If connectivity of the graph is maintained then the system is anonymous for all cryptographers. Otherwise, the connected components of the graph will represent different anonymous sets of cryptographers.

5 Related work

[8]) developed 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 formulas 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 [6, 19] epistemic logic is used to characterize a number of information-hiding requirements (including anonymity).[19] 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 formulas in the logic. However, to verify whether a given system satisfies a property it is necessary to formalize the behavior of system agents as knowledge-based programs. On the other hand, [6] uses a completely semantic approach and provides an appropriate semantic framework in which to consider anonymity. It also using this framework to derive notions of probabilistic anonymity. The basic notions of probability in anonymity that we have considered is similar to the ones described there, but our definition is based only on probability theory and is free from this framework.

References

[1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 10 Jan. 1999.

[2] R. M. Amadio and D. 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.

[3] M. Boreale and D. Sangiorgi. Some congruence properties for π-calculus bisimilarities. *Theoretical Computer Science*, 198(1,2):159–176, 1998.

[4] S. Brookes, C. Hoare, and A. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, 1984.

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

[6] J. Y. Halpern and K. R. O'Neill. Anonymity and information hiding in multiagent systems. In *Proc. of the 16th*

IEEE Computer Security Foundations Workshop, pages 75–88, 2003.

[7] O. M. Herescu and C. Palamidessi. Probabilistic asynchronous π-calculus. In J. Tiuryn, editor, *Proceedings of FOSSACS 2000 (Part of ETAPS 2000)*, volume 1784 of *Lecture Notes in Computer Science*, pages 146–160. Springer-Verlag, 2000.

[8] D. Hughes and V. Shmatikov. Information hiding, anonymity and privacy: a modular approach. *Journal of Computer Security*, 12(1):3–36, 2004.

[9] G. 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.

[10] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[11] R. Milner, J. Parrow, and D. 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.

[12] C. Palamidessi and O. M. Herescu. A randomized encoding of the π-calculus with mixed choice. *Theoretical Computer Science*, 2004. To appear.

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

[14] 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.

[15] D. Sangiorgi. π-calculus, internal mobility and agent-passing calculi. *Theoretical Computer Science*, 167(1,2):235–274, 1996.

[16] S. Schneider. Security properties and csp. In *Proceedings of the IEEE Symposium Security and Privacy*, 1996.

[17] S. Schneider and A. 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-Verlag, 1996.

[18] R. Segala and N. 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: 22–25.

[19] P. F. Syverson and S. G. Stubblebine. Group principals and the formalization of anonymity. In *World Congress on Formal Methods (1)*, pages 814–833, 1999.

[20] P. Y.Ryan and S. Schneider. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001.

A Appendix

A.1 The π-calculus

We recall here the basic notions about the π-calculus. We choose the variant used in [3, 15], 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:

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

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

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 $\mathbf{0}$ (*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: $P_0 \mid P_1 \mid P_2 \mid \ldots \mid P_{k-1}$ stands for $(\ldots((P_0 \mid P_1) \mid P_2) \mid \ldots \mid P_{k-1})$, i.e. the parallel operator is left associative, and $\alpha_1.P_1 \mid \alpha_2.P_2$ 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 $\mathit{fn}(P)$. 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 $[x=y]\,P \mid [x \neq y]\,P$.

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

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

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 2). 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 2. 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) $!P = P \,|\, !P$,

(vi) $[x = x]\, P = P$,

(vii) $[x \neq y]\, P = P$, if x is syntactically different from y.

I-SUM	$\sum_i \alpha_i.P_i \xrightarrow{x(z)} P_j[z/y] \quad \alpha_j = x(y)$
O/τ-SUM	$\sum_i \alpha_i.P_i \xrightarrow{\alpha_j} P_j \quad \alpha_j = \bar{x}y$ or $\alpha_j = \tau$
OPEN	$\dfrac{P \xrightarrow{\bar{x}y} P'}{\nu y P \xrightarrow{\bar{x}(y)} P'} \quad x \neq y$
RES	$\dfrac{P \xrightarrow{\mu} P'}{\nu y P \xrightarrow{\mu} \nu y P'} \quad y \notin n(\mu)$
PAR	$\dfrac{P \xrightarrow{\mu} P'}{P\|Q \xrightarrow{\mu} P'\|Q} \quad bn(\mu) \cap fn(Q) = \emptyset$
COM	$\dfrac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}y} Q'}{P\|Q \xrightarrow{\tau} P'\|Q'}$
CLOSE	$\dfrac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}(y)} Q'}{P\|Q \xrightarrow{\tau} \nu y(P'\|Q')}$
CONG	$\dfrac{P \equiv P' \quad P' \xrightarrow{\mu} Q' \quad Q' \equiv Q}{P \xrightarrow{\mu} Q}$

Table 2. The transition system of the π-calculus.