Weak Probabilistic Anonymity [1]

Yuxin Deng [2]

INRIA Sophia-Antipolis and Université Paris 7

Catuscia Palamidessi Jun Pang

INRIA Futurs and LIX, École Polytechnique

Abstract

Anonymity means that the identity of the user performing a certain action is maintained secret. The protocols for ensuring anonymity often use random mechanisms which can be described probabilistically. In this paper we propose a notion of *weak* probabilistic anonymity, where *weak* refers to the fact that some amount of probabilistic information may be revealed by the protocol. This information can be used by an observer to infer the likeliness that the action has been performed by a certain user. The aim of this work is to study the degree of anonymity that the protocol can still ensure, despite the leakage of information.

We illustrate our ideas by using the example of the dining cryptographers with biased coins. We consider both the cases of nondeterministic and probabilistic users. Correspondingly, we propose two notions of weak anonymity and we investigate their respective dependencies on the biased factor of the coins.

Key words: Anonymity, Probability, Nondeterminism, Dining Cryptographers.

1 Introduction

Anonymity is the property of keeping secret the identity of the user performing a certain action. The need for anonymity may raise in a wide range of situations, like postings on electronic forums, voting, delation, donations, and many others.

The protocols for ensuring anonymity often use random mechanisms. This is the case, for example, of the Dining Cryptographers [3], Crowds [8], Onion Routing [12], and SG-MIX [7].

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

[2] Supported by the EU project PROFUNDIS.

Various notions of probabilistic anonymity have been investigated in the literature [3,8,5,2]. In this paper we propose a notion of *weak* probabilistic anonymity, where *weak* refers to the fact that some amount of probabilistic information may be revealed by the protocol. Typical causes may be either the presence of attackers which interfere with the normal execution the protocol, or some unavoidable imperfection of the internal mechanisms, or may even be inherent to the way the protocol is designed. In any case, the information leaked by the system can be used by an observer to infer the likeliness that the action has been performed by a certain user. The aim of this work is to study the degree of anonymity that the protocol can still ensure, despite the leakage of information.

We illustrate our ideas by using the example of the Dining Cryptographers Problem (DCP). In this protocol, a number of users (cryptographers) cooperate to ensure that the occurrence of a certain action is made visible, while the cryptographer who has performed it remains anonymous. They achieve this goal by executing a certain algorithm which involves coin tossing. In the original formulation of [3] the coins are perfectly fair and no one (except the authorized cryptographers) gets any information about the results of the coins. As a consequence of these assumptions, the protocol ensures strong anonymity in the sense that, from the point of view of an observer, there is no way to infer that a cryptographers is more likely than another to have performed the action.

We consider a more realistic scenario in which some probabilistic information may be leaked by the system. In particular, we consider the case in which this happens due to imperfections in its internal mechanisms. In the case of the DCP, this means to relax the hypothesis of perfect fairness of the coins. It is worth noting that even if an observer does not know a priori whether and how much the coins in the DCP are biased, he may be able to infer it statistically by running the protocol several times [2]. One of the main purposes of this work is to investigate how the biased factor of the coins influences the level of anonymity that the system can still achieve.

An issue to consider when we deal with a probabilistic system is whether or not there is also some nondeterministic choice involved. Nondeterministic means that the choice is completely unpredictable. In anonymity protocols, the user which perform the action may be selected either nondeterministically or probabilistically. In the nondeterministic case, the probabilistic aspect of anonymity can only be relative to the probability of the observables, which derives solely from the randomness of the internal mechanisms of the protocol. The natural notion of anonymity is then that the probability of the observables does not give information about the user.

In the case of probabilistic users, there are two possible points of view under which one can define the notion of anonymity. Namely, we can focus on the probability of the observables, and require that they do not allow to infer information about the probability of the users (similarly to the nondeterministic case), or we can focus on the probability of the users, and require that the system does not allow to infer extra information about it through the observables. Interestingly, in the case of strong anonymity these two notions have been proved equivalent [2].

In this paper we consider both the cases of nondeterministic and probabilistic users, and we propose two notions of weak anonymity corresponding to the two points of view illustrated above. Although, as just said, in the limit case of strong anonymity these two notions are equivalent, their functional dependency on the biased factor of the coins turns out to be totally different.

1.1 Contributions

The main contributions of this work are:

- We propose two notions of weak probabilistic anonymity, for the cases of nondeterministic and probabilistic users, respectively.
- We consider the Dining Cryptographers with biased coins, and we study how the two notions of weak anonymity depend on the biased factor of the coins.
- We show how to code the formulas that expresses weak anonymity in PRISM, so that their validity can be checked automatically on a generic protocol.

1.2 Plan of the paper

In next section we recall some notions which are used in the rest of the paper: the Probabilistic Automata, the Dining Cryptographers Problem, and the framework for anonymity developed in [2]. In Section 3 we propose a notion of weak anonymity for nondeterministic users, and we study the dependency on the biased factor of the coins for the DCP. In Section 4 we do the same for the case of probabilistic users. In Section 5 we code in PRISM the DCP and the notions of anonymity. Finally, in Section 6 we conclude and discuss some related work.

For reasons of space the proofs are omitted. They can be found in [4].

2 Preliminaries

2.1 Nondeterminism and probability

In this paper we consider systems that can perform both probabilistic and nondeterministic choices. Intuitively, a probabilistic choice represents a set of alternative transitions, each of them associated to a certain probability of being selected. The sum of all probabilities on the alternatives of the choice must be 1, i.e. they form a *probability distribution.* Nondeterministic choice is also a set of alternatives, but we have no information on how likely one alternative is selected.

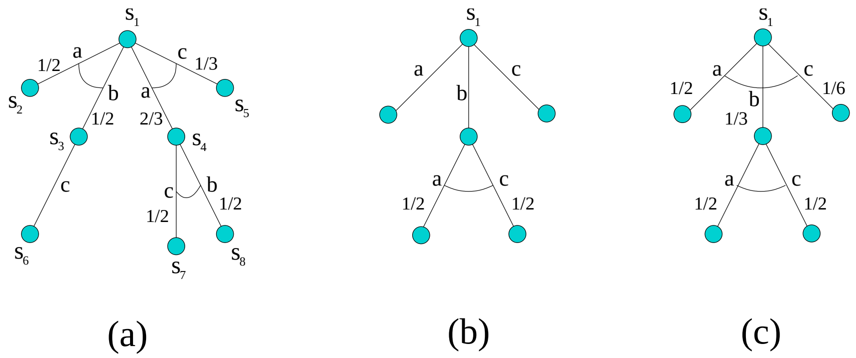


Fig. 1. Examples of probabilistic automata

We take the point of view that a nondeterministic choice *is not a probabilistic choice with unknown probabilities*: in the latter, if we repeatedly run the program, we can infer the probability. For instance, if we have a choice between two transitions and we observe that they are selected with the same frequency, we can infer that the probability is close to 1/2. In the nondeterministic case, this inference would be invalid. Nondeterministic means that the choice is totally unpredictable and that there is no assumption of regularity through time on the mechanisms that determine the selection.

There have been many models proposed in literature that combine both nondeterministic and probabilistic choice. One of the most general is the formalism of *probabilistic automata* proposed in [11]. We give here a brief and informal description of it.

A probabilistic automaton consists in a set of states, and labeled transitions between them. For each node, the outgoing transitions are partitioned in groups called *steps*. Each step represents a probabilistic choice, while the choice between the steps is nondeterministic.

Figure 1 illustrates some examples of probabilistic automata. We represent a step by putting an arc across the member transitions. For instance, in (a), state s_1 has two steps, the first is a probabilistic choice between two transitions with labels a and b, each with probability 1/2. When there is only a transition in a step, like the one from state s_3 to state s_6, the probability is of course 1 and we omit it.

In this paper, we use only a simplified kind of automaton, in which from each node we have either a probabilistic choice or a nondeterministic choice (more precisely, either one step or a set of singleton steps), like in (b). This is not a real restriction since it subsumes the so-called *alternated model*, in which probabilistic and nondeterministic choices alternate, and which is known to have the same expressive power as the full probabilistic automata. In the particular case that the choices are all probabilistic, like in (c), the automaton is called *fully probabilistic*.

Given an automaton M, we denote by $etree(M)$ its unfolding, i.e. the tree of all possible executions of M (in Figure 1 the automata coincide with their unfolding because there is no loop). If M is fully probabilistic, then each execution (maximal branch) of $\mathit{etree}(M)$ has a probability obtained as the product of the probability of the edges along the branch. In the finite case, we can define a probability measure for each set of executions, called *event*, by summing up the probabilities of the elements[3]. Given an event x, we will denote by $p(x)$ the probability of x. For instance, let the event c be the set of all computations in which c occurs. In (c) its probability is $p(c) = 1/3 \times 1/2 + 1/6 = 1/3$.

When nondeterminism is present, the probability can vary, depending on how we *resolve* the nondeterminism. In other words we need to consider a function ς that, each time there is a choice between different steps, selects one of them. By pruning the non-selected steps, we obtain a fully probabilistic execution tree $\mathit{etree}(M, \varsigma)$ on which we can define the probability as before. For historical reasons (i.e. since nondeterminism typically arises from the parallel operator), the function ς is called *scheduler*.

It should then be clear that the probability of an event is relative to the particular scheduler. We will denote by $p_\varsigma(x)$ the probability of the event x under the scheduler ς. For example, consider (a). We have two possible schedulers determined by the choice of the step in s_1. Under one scheduler, the probability of c is $1/2$. Under the other, it is $2/3 \times 1/2 + 1/3 = 2/3$. In (b) we have three possible schedulers under which the probability of c is 0, $1/2$ and 1, respectively.

2.2 The Dining Cryptographers

The general Dining Cryptographers Problem [3] is described as follows: A number of cryptographers, situated in the nodes of a given connected graph, are having a dinner. The representative of their organization (master) may or may not pay the bill of the dinner. If he does not, then he will select exactly one cryptographer and order him to pay the bill. The master will tell secretly each cryptographer whether he has to pay or not. The cryptographers would like to reveal whether the bill is paid by the master or by one of them, but, in the latter case, they wish to keep anonymous the identity of the payer.

A possible solution to this problem, described in [3], is to associate a coin to each edge of the graph, visible only to the adjacent cryptographers. The coins are then tossed, and each cryptographer computes the binary sum of the adjacent coins (counting 0, say, for head and 1 for tail), adds 1 if he is the payer, and outputs the result.

In [3] it is proved that the payer is one of the cryptographers if and only if the binary sum of all the outputs is 1. Furthermore, if the coins are fair, then an external observer cannot identify the payer when it is one of the

[3] In the infinite case things are more complicated: we cannot define a probability measure for all sets of execution, and we need to consider as event space the σ-field generated by the *cones* of $\mathit{etree}(M)$. However, in this paper, we consider only the finite case.

cryptographers.

The DCP will be a running example through the paper.

2.3 Anonymity systems

In this section we recall our approach to anonymity, as developed in [2].

We model the anonymity protocol as a probabilistic automaton M. The concept of anonymity is relative to the set of anonymous users and to what is visible to the observer. Hence, following [10,9] we classify the actions of M into the three sets A, B and C as follows:

- A is the set of the anonymous actions $A = \{a(i) \,|\, i \in I\}$ 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*.
- B is the set of the observable actions. We will use b, b', $\ldots$ to denote the elements of this set.
- C is the set of the remaining actions (which are unobservable).

Note that the actions in A normally are not visible to the observer, or at least, not for the part that depends on the identity i. However, for the purpose of defining and verifying anonymity we model the elements of A as visible outcomes of the system.

Definition 2.1 An anonymity system is a tuple (M, I, a, B, Z, p), where M is a probabilistic automaton, (I, a) is an anonymous action generator, B is a set of observable actions, Z is the set of all possible schedulers for M, and for every $\varsigma \in Z$, p_ς is the probability measure on the event space generated by $etree(M, \varsigma)$.

If the system is fully probabilistic, then Z is a singleton and we omit it.

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 as their maximal sequence of observable actions the sequence o (where o is of the form $b_1 b_2 \ldots b_n$ for some $b_1, b_2, \ldots, b_n \in B$). We denote by O (*observables*) the set of all such o's.

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

We wish to keep the notion of observables as general as possible, but we still need to make some assumptions on them. First, we want the observables to be disjoint events. Second, they must cover all possible outcomes. Third, an observable o must indicate unambiguously whether a has taken place or not, i.e. it either implies a, or it implies $\neg a$. In set-theoretic terms it means that either o is a subset of a or of the complement of a. Formally:

Assumption 1 (on the observables)

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

(ii) $\forall \varsigma \in Z.\ \ p_\varsigma(O) = 1$

(iii) $\forall \varsigma \in Z.\ \forall o \in O.\ \ (p_\varsigma(o \cap a) = p_\varsigma(o)) \ \ \vee \ \ p_\varsigma(o \cap \neg a) = p_\varsigma(o)$

Analogously, we need to make some assumption on the anonymous actions. We consider first the conditions tailored for the nondeterministic users: each scheduler determines completely whether an action of the form $a(i)$ takes place or not, and in the positive case, there is only one such i. Formally:

Assumption 2 (on the anonymous actions, for nondeterministic users)

$$\forall \varsigma \in Z.\ \ p_\varsigma(a) = 0 \vee (\exists i \in I.\ (p_\varsigma(a(i)) = 1 \ \wedge\ \forall j \in I.\ j \neq i \Rightarrow p_\varsigma(a(j)) = 0))$$

In [2] the following strong notion of anonymity was proposed. Intuitively, given two schedulers ς and ϑ that both choose a (say $a(i)$ and $a(j)$, respectively), it should not be possible to detect from the probabilistic measure of the observables whether the scheduler was ς or ϑ (i.e. whether the selected user was i or j).

Definition 2.2 [(Strong) anonymity for nondeterministic users] A system (M, I, a, B, Z, p) is anonymous if

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

We now consider the case in which the users are fully probabilistic. The assumption on the anonymous actions in this case is much weaker: we only require that there be at most one user that performs a, i.e. $a(i)$ and $a(j)$ must be disjoint for $i \neq j$. Formally:

Assumption 3 (on the anonymous actions, for probabilistic users)

$$\forall i, j \in I.\ \ i \neq j \ \Rightarrow\ p(a(i) \cup a(j)) = p(a(i)) + p(a(j))$$

The probabilistic counterpart of Definition 2.2 can be formalized using 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 $p(x \mid y)$, is equal to $p(x \cap y)/p(y)$.

Definition 2.3 A fully probabilistic system (M, I, a, B, p) is anonymous if

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

The notions of anonymity illustrated so far focus on the probability of the observables. In the case of probabilistic users, however, one can also approach the concept of anonymity from the point of view of the probabilistic information associated to the users. This is the perspective adopted in [5] to define what they call *conditional anonymity*. The idea is that a system is anonymous if the observations do not change the probability of the $a(i)$'s. In other words, we may know the probability of $a(i)$ by some means external to the system, but the system should not increase our knowledge about it. The same notion was proposed, implicitly, in [3]. This concept can be formulated in our framework as follows:

Definition 2.4 [(Strong) anonymity for probabilistic users] A fully probabilistic system (M, I, a, B, p) is anonymous if

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

Despite Definitions 2.3 and 2.4 are based on conceptually different interpretations of anonymity, it has been shown that they are equivalent (see [2]).

The definitions of anonymity illustrated in this section are satisfied by the DCP only if the coins are fair. In next sections we propose weak versions of these definitions, which may be satisfied also when the coins are biased, depending on the biased factor.

3 Weak anonymity for nondeterministic users

In this section we propose a weak variant of Definition 2.2 and we study, in the particular case of the DCP, how this property depends on the biased factor of the coins.

Intuitively, the weakening consists in relaxing the constraint that the probability of an observer implying a is the same under every scheduler. Instead, we require that the difference between any two such probabilities does not exceed a certain parameter α. Formally:

Definition 3.1 [α-anonymity for nondeterministic users] Given $\alpha \in [0, 1]$, a system (M, I, a, B, Z, p) is α-anonymous if

$$max\{\, p_\varsigma(o) - p_\vartheta(o) \;\;|\, \varsigma, \vartheta \in Z,\; o \in O,\; p_\varsigma(o \cap a) = p_\varsigma(o),\; p_\vartheta(o \cap a) = p_\vartheta(o)\} = \alpha$$

Intuitively, $p_\varsigma(o) - p_\vartheta(o) \;=\; \alpha$ means that, whenever we observe o, we suspect that user i is more likely than user j to have performed the action by an additive factor α (where i and j represent the users selected by ς and ϑ, respectively).

Let us consider the DCP on a linear graph consisting of three nodes, i.e. three cryptographers $Crypt_0$, $Crypt_1$, and $Crypt_2$, and two edges, $Coin_0$ between $Crypt_0$ and $Crypt_1$, and $Coin_1$, between $Crypt_1$ and $Crypt_2$.

In case one of the cryptographers pays (event a), the possible observables are

$$o_1 = 111 \qquad o_2 = 100 \qquad o_3 = 010 \qquad o_4 = 001$$

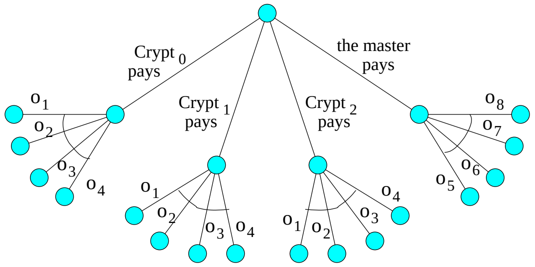


Fig. 2. The DCP with three cryptographers and nondeterministic master.

where $b_0b_1b_2$ refers to the outputs of $Crypt_0$, $Crypt_1$ and $Crypt_2$, respectively. For instance, if $Crypt_1$ is the payer, then o_1 is obtained when both the two coins give 1, o_2 is obtained when $Coin_0$ gives 1 and $Coin_1$ gives 0, etc. In case the master pays, then the possible observables are $o_5 = 110, o_6 = 101, o_7 = 011, o_8 = 000$. For instance, $o_5 = 110$ is obtained when $Coin_0$ gives 1 and $Coin_1$ gives 0.

The probabilistic automaton corresponding to this situation is illustrated in Figure 2. For simplicity, we have drawn only the "big-step-transitions" corresponding to the observables o_1, o_2 etc. They represents sequences of "small-step-transitions" where each coin is flipped, then each cryptographer in turn reads the coins, then it computes and output the results.

It is important to note that we will consider only one form of nondeterminism: that associated to the choice of the master (*nondeterministic master*, which in the DCP is synonymous of nondeterministic users). In general in a system there is also the nondeterminism caused by the different possible interleaving of the various components of the system, but here, for simplicity, we will assume that the order in which the various components of the system (master, cryptographers, coins) execute their operations is fixed. In any case, it can be shown that this latter form of nondeterminism would not affect the properties of the DCP with respect to anonymity.

Let us represent by β_i the "biased factor" of $Coin_i$, i.e. the probability that $Coin_i$ gives 0. We want to determine how the parameter α of anonimity (Definition 3.1) depends on β_0 and β_1.

Consider, for each scheduler that selects a payer among the cryptographers, the possible observables and their probability measure. A simple calculation gives the figures shown in Table 1. Then by case analysis, we obtain:

$$\alpha = \begin{cases} |1-(\beta_0+\beta_1)| & \text{if } (\beta_0, \beta_1 \leq 0.5) \text{ or } (\beta_0, \beta_1 \geq 0.5); \\ |\beta_0 - \beta_1| & \text{if } (\beta_0 > 0.5 \text{ and } \beta_1 < 0.5) \text{ or } (\beta_0 < 0.5 \text{ and } \beta_1 > 0.5); \end{cases}$$

Figure 3 shows the graph of α as a function of β_0 and β_1.

The above analysis can be extended to the general case of linear graphs with any number of nodes.

Theorem 3.2 *In the DCP on a linear graph with n nodes the α in Defini-*

Observables: $o_1 = 111$, $o_2 = 100$, $o_3 = 010$, $o_4 = 001$			
	$Crypt_0$ pays	$Crypt_1$ pays	$Crypt_2$ pays
$p(o_1)$	$\beta_0(1-\beta_1)$	$(1-\beta_0)(1-\beta_1)$	$(1-\beta_0)\beta_1$
$p(o_2)$	$\beta_0\beta_1$	$(1-\beta_0)\beta_1$	$(1-\beta_0)(1-\beta_1)$
$p(o_3)$	$(1-\beta_0)\beta_1$	$\beta_0\beta_1$	$\beta_0(1-\beta_1)$
$p(o_4)$	$(1-\beta_0)(1-\beta_1)$	$\beta_0(1-\beta_1)$	$\beta_0\beta_1$

Table 1
Probabilities of the observables in the case of 3 cryptographers on a linear graph.

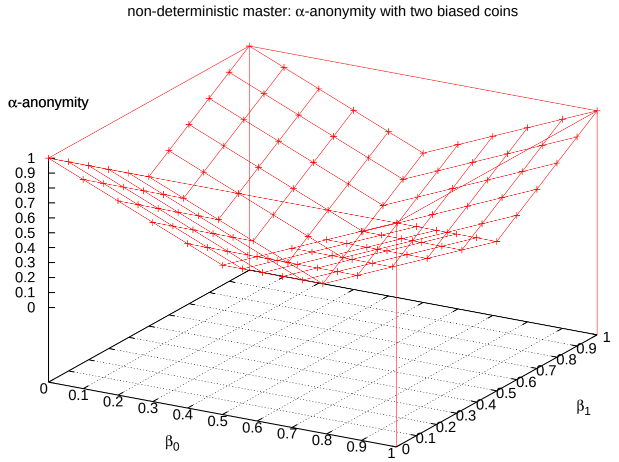


Fig. 3. The dependency of α-anonymity on β_0 and β_1 in the case of three cryptographers.

tion 3.1 depends on the β_i's as follows:

$$\alpha = \prod_{\beta_i \geq 0.5} \beta_i \prod_{\beta_j < 0.5} (1-\beta_j) \quad - \quad \prod_{\beta_i \geq 0.5} (1-\beta_i) \prod_{\beta_j < 0.5} \beta_j$$

It is possible to show that the above theorem holds also when the topology is a ring. On the other hand, it does not hold for graphs which contain one or more nodes with an odd number of adjacent edges.

Figure 4 illustrates the dependency of α on β for three to six cryptographers, where for all i, $\beta_i = \beta$ (uniform coins). We note that the anonymity level increases (i.e. α decreases) as the number of cryptographers increases. If the coins are fair ($\beta = 0.5$), then we have strong anonymity, i.e. $\alpha = 0$. In the two extreme cases of $\beta = 0$ or $\beta = 1$, the α-anonymity is always 1, which is maximal. It is also possible to show that α is expressed by a polynomial on β whose degree is $n-1$ if n is even, and $n-2$ if n is odd.

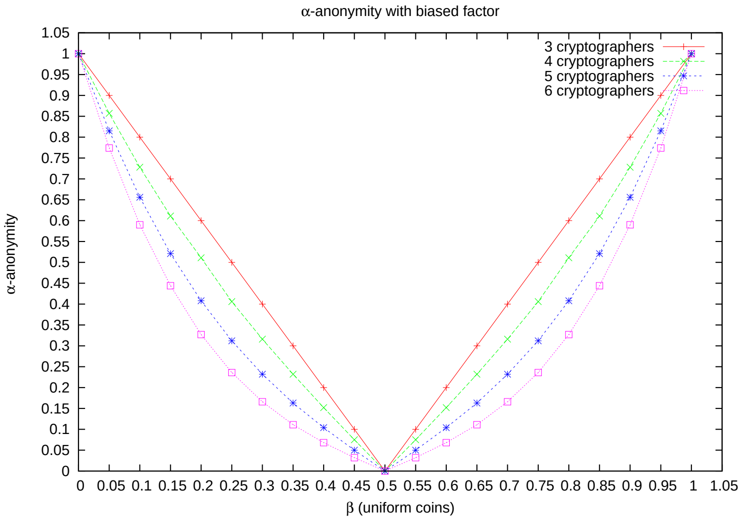


Fig. 4. The dependency of α-anonymity on β's in the case of 3-6 cryptographers.

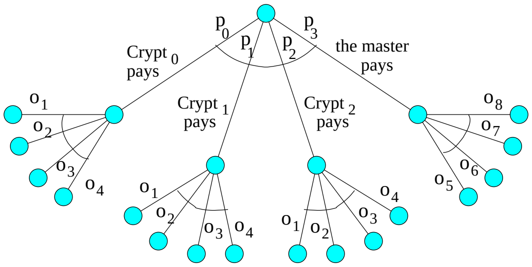


Fig. 5. The DCP with three cryptographers and probabilistic master.

4 Weak anonymity for probabilistic users

In this section we consider the case in which the user is selected according to a certain probability distribution. Since we assume that we have no other source of nondeterminism, the automata that we consider in this section are fully probabilistic. For example, in the case of the DCP with three cryptographers, we have the automaton represented in Figure 5.

4.1 Focusing on the probabilities of the observables

The fully probabilistic version of Definition 3.1, corresponding also to the weak version of Definition 2.3, is the following:

Definition 4.1 [α-anonymity for probabilisitic users] Given $\alpha \in [0,1]$, a fully probabilistic system (M, I, a, B, p) is α-anonymous if

$$max\{\ p(o \mid a(i)) - p(o \mid a(j))\ \mid i,j \in I,\ o \in O,\ p(a(i)) > 0,\ p(a(j)) > 0\ \} = \alpha$$

Like Definition 3.1, this notion focuses on the probability of the observables.

It is possible to prove that for the DCP the α of the above definition depends on the β_i's exactly like the α for nondeterministic users (Definition 3.1). In fact consider, in the case of nondeterministic users, a scheduler ς that selects i, i.e. $p_\varsigma(a(i)) = 1$. Then assume, in the case of probabilistic users, that $p_\varsigma(a(i)) > 0$. It is easy to see that $p_\varsigma(o) = p(o \mid a(i))$.

So, in a sense, the notion of anonymity proposed in Definition 4.1 does not seem to introduce any new technical challenge with respect to the study done for the case of nondeterministic users.

In next section we investigate, instead, the weak version of the alternative notion of anonymity given in Definition 2.4.

4.2 Focusing on the probabilities of the users

We take here the point of view that anonymity means to preserve the probability of the users, like in [3] and [5].

Definition 4.2 [α-anonymity for probabilisitic users – alternative notion] Given $\alpha \in [0,1]$, a fully probabilistic system (M, I, a, B, p) is α-anonymous if

$$max\{\ p(a(i) \mid o) - p(a(i) \mid a) \ \mid i \in I,\ o \in O,\ p(o \cap a) > 0\} = \alpha$$

Intuitively, $p(a(i) \mid o) - p(a(i) \mid a) \ = \ \alpha$ means that, after observing o, the probability we attribute to i as the performer of the action, has increased by an additive factor α.

We study now the dependency of α on the β_i's in the case of the DCP with n cryptographers on a linear graph. We need to introduce some definitions: Let p_i be the probability that $Crypt_i$ is the payer. Of course, the probability that one of the cryptographers is the payer is then $\sum_{i=0}^{n-1} p_i$. Let k be the index of the cryptographer with the highest probability, i.e.

$$p_k = max\{p_i \mid i \in [0, n-1]\}$$

For $i \in [0, n-2]$, define

$$\gamma_i = \begin{cases} \beta_i & \text{if } \beta_i \geq 0.5 \\ 1 - \beta_i & \text{otherwise} \end{cases}$$

Finally, for an arbitrary $j \in [0, n-1]$, define

$$q_j = \begin{cases} \prod_{i=0}^{j-1} \gamma_i \prod_{i=j}^{k-1} (1-\gamma_i) \prod_{i=k}^{n-2} \gamma_i & \text{if } j \leq k \\ \prod_{i=0}^{k-1} \gamma_i \prod_{i=k}^{j-1} (1-\gamma_i) \prod_{i=j}^{n-2} \gamma_i & \text{otherwise} \end{cases}$$

We are now ready to show how α depends on the β_i's:

Theorem 4.3 *In the DCP on a linear graph with n nodes the α in Definition 4.2 depends on the β_i's (and on the p_i's) as follows:*

$$\alpha = \frac{q_k\ p_k}{\sum_{j=0}^{n-1} q_j\ p_j} - \frac{p_k}{\sum_{j=0}^{n-1} p_j} \tag{1}$$

Figure 6 shows the dependency of α on the β_i's in the case of three cryptographers. The various graphs refer to different probability distributions for the payer. It is worth noting that, in contrast to the notion of α-anonymity given in Definition 4.1, the version presented in this section depends not only the β_i's, but also on the $p(a(i))$'s. On the other hand, in the limit case of strong anonymity, the two notions are equivalent, as explained in Section 2.3.

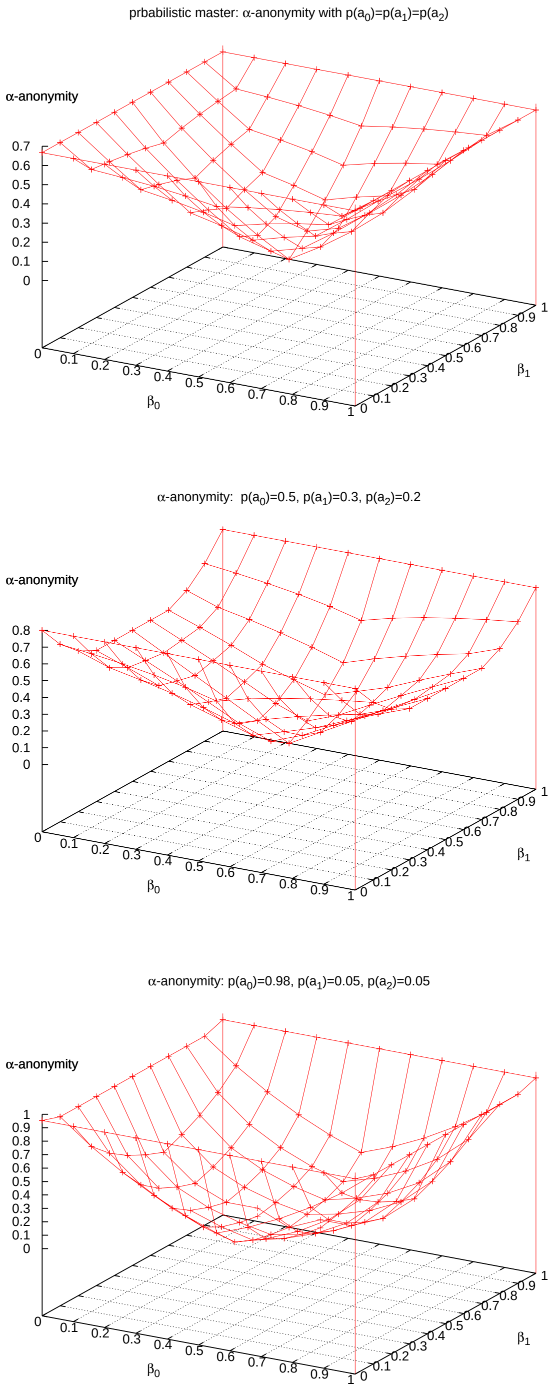
5 Automatic Analysis

In the case of a very simple topology (linear graphs) we have been able to express the dependency of α on the β_i's with a mathematical formula. In this way, if we have a system whose internal bias are known, it is immediate to check whether it satisfies weak anonymity (for a given α) or not. It is possible to extend the method also to rings, but as the graphs get more complicated, it is not clear how to proceed to find the formula that express the dependency. This is a typical situation for most real-life systems: the symbolic analysis is often unfeasible, and we have to resort to automatic tools supported by computers.

In this section, we describe how to use the probabilistic model checker PRISM to check the property of the α-anonymity for the DCP. We consider both nondeterministic and probabilistic masters (recall that in the DCP nondeterministic/probabilistic master is synonymous of nondeterministic/probabilistic users). We model the DCP as a discrete-time Markov chain (DTMC) in the case of a probabilistic master, and as Markov decision process (MDP) in the case of a nondeterministic master[4]. The PRISM input language is a simple, state-based language, based on the Reactive Modules formalism of Alur and Henzinger [1]. The events are formalized using the temporal probabilistic logic PCTL [6]. Once this translation is done, we can use PRISM to compute the probabilities of the relevant events so to check α-anonymity. A brief overview of PRISM and PCTL can be found in [4].

The following code is for three cryptographers and two coins arranged in a line. It can be easily generalized to more cryptographers and a different

[4] DTMC and MDP, which are the formats accepted by PRISM, can be seen as special cases of probabilistic automata.


prbabilistic master: α-anonymity with p(a0)=p(a1)=p(a2)

α-anonymity: p(a0)=0.5, p(a1)=0.3, p(a2)=0.2

α-anonymity: p(a0)=0.98, p(a1)=0.05, p(a2)=0.05

Fig. 6. The dependency of α-anonymity on the β_i's in the case of three cryptographers.

graph structure. First we describe the variables we use in the model. N is the number of cryptographers, and, since the topology is a line, there are N-1 coins in the model. The probabilities of each coin of showing head are defined as beta0, beta1, etc. We define three more state variables: s_master:

[0..2] for the master, s_coin: [0..N-1] to indicate how many coins have been flipped, and s_crypt: [0..N] to indicates how many cryptographers have decided their outputs. payerid=N indicates that either no cryptographer will pay or the master hasn't decided yet.. Initially, they are all 0. Once the execution terminates, we will have s_master=2, s_coin=N-1, and s_crypt=N. The variable payerid: [0..N] init N is used to record who is the payer. The variable toss: bool init false is used to let the coins to be flipped after the master has made his decision.

In the following, we consider the case N = 3. We use the variables crypt0, crypt1, crypt2 to record the values computed by each cryptographer, that are either 0 or 1 and depend on whether the cryptographer is paying and on the sides of the coins the cryptographer can see. Initially, their values are 0. In the model, there are two coins coin0 and coin1. The first is shared shared by Cryptographers 0 and 1, the second is shared by Cryptographers 1 and 2. We use 0 for head, and 1 for tail.

Next, we describe the behavior of the master, the coins and the cryptographers. If the master is nondeterministic, he will decide nondeterministically the payer: one of the cryptographers (payerid = 0, 1, or 2) or himself (payerid = 3). Once he has made the decision, the value of toss is set to true, in order to let the coins to be flipped.

```
[] (s_master=0) → (s_master'=1) & (payerid'=0);
[] (s_master=0) → (s_master'=1) & (payerid'=1);
[] (s_master=0) → (s_master'=1) & (payerid'=2);
[] (s_master=0) → (s_master'=1) & (payerid'=3);
[] (s_master=1) & (!toss) → (s_master'=2) & (toss'=true);
```

If the master is probabilistic, then the choice of the payer is based on a probability distribution. For instance:

```
[] (s_master=0) →
                0.5: (s_master'=1) & (payerid'=0) +
                0.3: (s_master'=1) & (payerid'=1) +
                0.1: (s_master'=1) & (payerid'=2) +
                0.1: (s_master'=1) & (payerid'=3);
[] (s_master=1) & (!toss) → (s_master'=2) & (toss'=true);
```

Once toss becomes true, the coins start to flip. With probabilities beta0 and beta1, the side of the coins will be head. With probabilities 1-beta0 and 1-beta1, the side of the coins will be tail. Each time when a coin is flipped, the value of s_coin is increased by one.

```
[] (s_coin=0) & (toss) →
                beta0: (coin0'=0) & (s_coin'=s_coin+1) +
                (1-beta0): (coin0'=1) & (s_coin'=s_coin+1);
[] (s_coin=1) & (toss) →
                beta1: (coin1'=0) & (s_coin'=s_coin+1) +
                (1-beta1): (coin1'=1) & (s_coin'=s_coin+1);
```

After all the coins have been flipped (s_coin=N-1), the cryptographers calculate the value of their variable crypt0, crypt1 and crypt2. Once a cryptographer has terminated this calculation, the value of s_crypt is increased by 1. Since the Cryptographers 0 and 2 sit at the two ends of the line, they can only observe one coin: Cryptographer 0 sees Coin 0, and Cryptographer 2 sees Coin 1. If Cryptographer 0 is the payer, he will set the variable crypt0 to 1 if he sees the head of Coin 0, and to 0 otherwise. If Cryptographer 0 is not the payer, he will set crypt0 to 0 if he sees the head of Coin 0, and to 1 otherwise. The code for Cryptographer 2 is similar: just rename crypt0 into crypt2, coin0 into coin1, and s_crypt=0 into s_crypt=2.

```
[] (s_crypt=0) & (s_coin=N-1) & (payerid=0) & (coin0=0) s→
                (crypt0'=1) & (s_crypt'=s_crypt+1);
[] (s_crypt=0) & (s_coin=N-1) & !(payerid=0) & (coin0=0) →
                (crypt0'=0) & (s_crypt'=s_crypt+1);
[] (s_crypt=0) & (s_coin=N-1) & (payerid=0) & (coin0=1) →
                (crypt0'=0) & (s_crypt'=s_crypt+1);
[] (s_crypt=0) & (s_coin=N-1) & !(payerid=0) & (coin0=1) →
                (crypt0'=1) & (s_crypt'=s_crypt+1);
```

The behavior of Cryptographer 1 is slightly different, since he can observe two coins. If he is the payer, he will set the variable crypt1 to 1 if the two coins have the same side, and to 0 otherwise. If he is not the payer, he will set crypt1 to 0 if the two coins have the same side, and to 1 otherwise.

```
[] (s_crypt=1) & (s_coin=N-1) & (payerid=1) & (coin1=coin0) →
                (crypt1'=1) & (s_crypt'=s_crypt+1);
[] (s_crypt=1) & (s_coin=N-1) & !(payerid=1) & (coin1=coin0) →
                (crypt1'=0) & (s_crypt'=s_crypt+1);
[] (s_crypt=1) & (s_coin=N-1) & (payerid=1) & !(coin1=coin0) →
                (crypt1'=0) & (s_crypt'=s_crypt+1);
[] (s_crypt=1) & (s_coin=N-1) & !(payerid=1) & !(coin1=coin0) →
                (crypt1'=1) & (s_crypt'=s_crypt+1);
```

A self-loop is added in the end of the specification to avoid deadlock states[5] .

```
[] (s_coin=N-1) & (s_master=2) & (s_crypt=N) →
                (s_coin'=N-1) & (s_master'=2) & (s_crypt'=N);
```

In the DCP, an external observer can see the values of the variables crypt0, crypt2 and crypt2. Furthermore, the values of the variables in the PRISM model define the states of the system. For example, the following predicate represents the final states in which all cryptographers output 1. We denote it by o_1.

[5] This is required by the design of PRISM.

```
(crypt0=1) & (crypt1=1) & (crypt2=1) &
(s_crypt=3) & (s_coin=2) & (s_master=2)
```

For each type of master (nondeterministic or probabilistic) we can describe observables as a PCTL formula by using the $\mathcal{P}$ operator (see [4]). Then, we can use PRISM to compute the probability of each observable for the analysis of α-anonymity.

Nondeterministic master:

If the master is nondeterministic, we can compute the maximum and the minimum probability of each observable, under any possible scheduler that selects one of the cryptographers to pay. Below, we specify the PCTL formulas to compute the probabilities of observable o_1.

$$\mathcal{P}_{max=?}[\mathit{true}\ \mathcal{U}\ o_1] \qquad \text{and} \qquad \mathcal{P}_{min=?}[\mathit{true}\ \mathcal{U}\ o_1]$$

Thus, it is sufficient to use the formulation of α-anonymity given by the following proposition, whose proof is immediate:

Proposition 5.1 *A system (M, I, a, B, Z, p) is α-anonymous (with respect to nondeterministic users) if*

$$\mathit{max}\{\ \begin{array}{l} \mathit{max}\{\ p_\varsigma(o)\ \mid \varsigma \in Z,\ p_\varsigma(o \cap a) = p_\varsigma(o)\ \} \\ - \\ \mathit{min}\{\ p_\vartheta(o)\ \mid \vartheta \in Z,\ p_\vartheta(o \cap a) = p_\vartheta(o)\ \} \end{array} \ \mid\ o \in O\ \} \ = \ \alpha$$

The results in Figure 4 have been checked using PRISM.

Probabilistic master:

When the master is nondeterministic, α-anonymity is defined as

$$\mathit{max}\{\ p(a(i) \mid o) - p(a(i) \mid a)\ | i \in I,\ o \in O,\ p(o \cap a) > 0\ \} = \alpha.$$

Since PRISM does not support the calculation of conditional probability as a primitive, we have to compute each $p(a(i) \mid o)$ using the equivalent expression $p(a(i) \cap o)/p(o)$. As for $p(a(i) \mid a)$, this is the same as $p(a(i))/p(a)$. For example, in case of three cryptographers, $p(a(0) \cap o_1)$ can be computed by using the PCTL formula

$$\mathcal{P}_{=?}[\mathit{true}\ \mathcal{U}\ o_1 \wedge (\mathit{payerid} = 0)]$$

The results presented in Figure 6 have been checked using PRISM.

6 Conclusion

We propose two notions of weak probabilistic anonymity, for the cases of nondeterministic and probabilistic users, respectively. We have applied these two notions to the DCP with biased coins, and we have described the functional dependency of the weakness level on the biased factor of the coins. Furthermore we have coded in PRISM the DCP and the formulas that express weak anonymity.

This paper builds on the framework of probabilistic anonymity proposed in [2] that we have summarized in Section 2.3. The notions that we investigate here represent a generalization of the strong probabilistic anonymity proposed in [2].

To our knowledge, the first notion of probabilistic anonymity was proposed (although not with an explicit definition) in [3]. That notion corresponds to one of the notions of strong anonymity for probabilistic users investigated in [2], and more precisely, to the one recalled in Definition 2.4. This is the notion for which we have given the weak version in Definition 4.2.

In [8] Reiter and Robin have proposed an hierarchy of notions of probabilistic anonymity in the context of Crowds. We recall that Crowds is a system aimed at protecting the identity of users when sending (originating) messages. This is achieved by forwarding the message to another user selected randomly, which in turn forward the message, and so on, until the message reaches its destination. Part of the users may be corrupted (attackers), and one of the main purposes of the protocol is to protect the identity of the originator of the message from those attackers.

The following is Reiter and Robin's description of the hierarchy. Here the *sender* stands for the user that forwards the message to the attacker.

Beyond suspicion From the attacker's point of view, the sender appears no more likely to be the originator of the message than any other potential sender in the system.

Probable innocence From the attacker's point of view, the sender appears no more likely to be the originator of the message than to not be the originator.

Possible innocence From the attacker's point of view, there is a nontrivial probability that the real sender is someone else.

These notions were only given informally in [8] and we are not sure how to interpret them formally. However, the property of anonymity which is actually proved in [8] for the system Crowds (and which the authors call "probable innocence") is described formally, and says that the probability that the originator forwards the message to an attacker is not greater than 1/2. Equivalently, the probability that an attacker receives the message from user i (observable event), given that i is the originator of the message (event $a(i)$), is not greater than 1/2. A notion of probable innocence in that sense corresponds to our Definitions 3.1 and 4.1 (for nondeterministic and probabilistic users, respectively) with $\alpha \leq 1/2$.

Halpern and O'Neill have proposed in [5] various notions of probabilistic anonymity, focusing on the probability of the users. Their principal notion is based on epistemic logic and is formulated as a requirement on the knowledge of the observer about the probability of the user. They have given both strong and weak version of this notion, proposing a formal interpretation the three levels of the hierarchy proposed by [8] (see above). These notions do not seem directly related to the ones we investigate in this paper. In particular, those in [8] depend on the probabilities of the $a(i)$'s, while our notions abstract from these probabilities. On the other hand, Halpern and O'Neill have proposed also another notion, called *conditional anonymity* (cfr. Definition 4.4 in [5]), which corresponds to the strong probabilistic anonymity recalled in Definition 2.4.

References

[1] R. Alur and T.A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

[2] Mohit Bhargava and Catuscia Palamidessi. Probabilistic anonymity. Technical report, INRIA Futurs and LIX, 2005. To appear in the proceedings of CONCUR 2005. Report version available at `http://www.lix.polytechnique.fr/~catuscia/papers/Anonymity/report.ps`.

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

[4] Yuxin Deng, Catuscia Palamidessi, and Jun Pang. Weak probabilistic anonymity. Technical report, INRIA Futurs and LIX, 2005. Available at `http://www.lix.polytechnique.fr/~catuscia/papers/Anonymity/reportWA.pdf`.

[5] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. In *Proc. of the 16th IEEE Computer Security Foundations Workshop*, pages 75–88, 2003.

[6] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[7] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proceedings of Information Hiding Workshop (IH 1998)*. Springer-Verlag, LNCS 1525, 1998.

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

[9] Peter Y. Ryan and Steve Schneider. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001.

[10] 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-Verlag, 1996.

[11] 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.

[12] 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.