Probabilistic Anonymity via Coalgebraic Simulations

Ichiro Hasuo[1]* and Yoshinobu Kawabe[2]

[1] Radboud University Nijmegen, the Netherlands
http://www.cs.ru.nl/~ichiro
[2] NTT Communication Science Laboratories, NTT Corporation, Japan
http://www.brl.ntt.co.jp/people/kawabe

Abstract. There is a growing concern on anonymity and privacy on the Internet, resulting in lots of work on formalization and verification of anonymity. Especially, importance of probabilistic aspect of anonymity is claimed recently by many authors. Among them are Bhargava and Palamidessi who present the definition of *probabilistic anonymity* for which, however, proof methods are not yet elaborated. In this paper we introduce a simulation-based proof method for probabilistic anonymity. It is a probabilistic adaptation of the method by Kawabe et al. for non-deterministic anonymity: anonymity of a protocol is proved by finding out a forward/backward simulation between certain automata. For the jump from non-determinism to probability we fully exploit a generic, coalgebraic theory of traces and simulations developed by Hasuo and others. In particular, an appropriate notion of probabilistic simulations is obtained by instantiating a generic definition with suitable parameters.

1 Introduction

Nowadays most human activities rely on communication on the Internet, hence on communication protocols. This has made verification of communication protocols a trend in computer science. At the same time, the variety of purposes of communication protocols has identified new verification goals—or *security properties*—such as anonymity, in addition to rather traditional ones like secrecy or authentication.

Anonymity properties have attracted growing concern from the public. There are emerging threats as well: for example, the European Parliament in December 2005 approved rules forcing ISPs to retain access records. Consequently more and more research activities—especially from the formal methods community—are aiming at verification of anonymity properties (see [2]).

Formal verification of anonymity properties is at its relative youth compared to authentication or secrecy. The topic still allows for definitional work (such as [4,7,8,11,16]) pointing out many different aspects of anonymity notions. Notably many authors [4,8,20,21] claim the significant role of *probability* in anonymity notions. This is the focus of this paper.

Bhargava and Palamidessi [4] define the notion of *probabilistic anonymity* which is mathematically precise and which subsumes many competing notions of anonymity

* This work was done during the first author's stay at NTT Communication Science Laboratories in September–October 2006.

in probabilistic settings. However, it is not yet elaborated *how* we can verify if an anonymizing protocol satisfies this notion of probabilistic anonymity.

In this paper we introduce a simulation-based proof method for probabilistic anonymity as defined by Bhargava and Palamidessi. It is a probabilistic extension of the method by Kawabe et al. [13,12] for a non-deterministic (as opposed to probabilistic) setting. The basic scenario is common in both non-deterministic and probabilistic cases:

1. First we model an anonymizing protocol to be verified as a certain kind of automaton $\mathcal{X}$.
2. Second we construct the *anonymized* version $\mathsf{an}(\mathcal{X})$ of $\mathcal{X}$. The automaton $\mathsf{an}(\mathcal{X})$ satisfies the appropriate notion of anonymity because of the way it is constructed.
3. We prove that

$$(\text{trace semantics of } \mathcal{X}) = (\text{trace semantics of } \mathsf{an}(\mathcal{X})) \ .$$

   Then, since the notion of anonymity is defined in terms of traces, anonymity of $\mathsf{an}(\mathcal{X})$ yields anonymity of $\mathcal{X}$. The equality is proved by showing that the (appropriate notion of) inclusion order $\sqsubseteq$ holds in both directions.
   - $\sqsubseteq$ holds because of the construction of $\mathsf{an}(\mathcal{X})$.
   - $\sqsupseteq$ is proved by finding a (forward or backward) *simulation* from $\mathsf{an}(\mathcal{X})$ to $\mathcal{X}$. Here we appeal to soundness theorem of simulations—existence of a simulation yields trace inclusion.

Hence the anonymity proof of $\mathcal{X}$ is reduced to finding a suitable forward/backward simulation.

There is an obvious difficulty in conducting this scenario in a probabilistic setting. The theory of traces and simulations in a non-deterministic setting is well studied e.g. by [14]; however appropriate definitions of probabilistic traces and simulations are far from trivial.

For the jump from non-determinism to probability we exploit a generic, coalgebraic theory of traces and simulations developed by Hasuo, Jacobs and Sokolova [9,10]. In the generic theory, fundamental notions such as systems (or automata), trace semantics and forward/backward simulations are identified as certain kinds of coalgebraic constructs. On this level of abstraction the general soundness theorem—existence of a (coalgebraic) simulation yields (coalgebraic) trace inclusion—is proved by categorical arguments.

The theory is generic in that, by fixing two parameters appearing therein, it instantiates to a concrete theory for various kinds of systems. In particular, according to the choice of one parameter, systems can be non-deterministic or probabilistic.[3] In this work a complex definition of probabilistic simulations is obtained as an instance of the general, coalgebraic definition. Moreover, this definition is an *appropriate* one: soundness theorem comes for free from the general soundness theorem.

The paper is organized as follows. In Section 2 we illustrate the probabilistic aspect of anonymity properties using the well-known example of Dining Cryptographers.

[3] Unfortunately the combination of both non-determinism and probability—which is e.g. in probabilistic automata [19]—is not covered in this paper. In fact this combination is a notorious one [6,23]: many mathematical tools that are useful in a purely non-deterministic or probabilistic setting cease to work in the presence of both.

We model anonymizing protocols as a special kind of automata called (probabilistic) *anonymity automata*. This notion is introduced in Section 3; the definition of probabilistic anonymity following [4] is also there. Finally in Section 4 we describe our simulation-based proof method for anonymity and prove its correctness. In Section 5 we conclude.

Notations In the sequel the disjoint union of sets X and Y is denoted by $X + Y$.

The set of lists over an alphabet X with length ≥ 1 is denoted by X^*X in a regular-expression-like manner: obviously we have $X^* = X^*X + \{\langle\rangle\}$. This appears as a domain of trace semantics for anonymity automata.

2 Motivating example: dining cryptographers (DC)

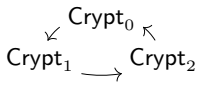
In this section—following [4]—we shall illustrate the probabilistic aspect of anonymity, using the well-known *dining cryptographers* (DC) protocol [5].

2.1 The DC protocol

There are three cryptographers (or users) dining together. The payment will be made either by one of the cryptographers, or NSA (U.S. National Security Agency) which organizes the dinner. Who is paying is determined by NSA; if one of the cryptographers is paying, she has been told so beforehand.

The goal of the DC protocol is as follows. The three cryptographers announce whether one of them is paying or not; but if it is the case, the information on *which* cryptographer is paying should be disguised from the viewpoint of an observer (called the *adversary* in the sequel) and also from that of the cryptographers who are not paying. This is where anonymity is involved.

The protocol proceeds in the following way. Three cryptographers Crypt_i for $i = 0, 1, 2$ sit in a circle, each with a coin Coin_i. The coins are held in such a way that they can be seen by the owner and one of the other two: in the following figure $\rightarrow$ denotes the "able-to-see-her-coin" relation.



Then the coins are flipped; each cryptographer, comparing the two coins she can see, announces to the public whether they *agree* (showing the same side) or *disagree*. The trick is that the one who is paying—if there is—lies on the announcement. For example, given that Crypt_0 is paying, then the configuration of coins

$$(\mathsf{h}, \mathsf{t}, \mathsf{h}) \qquad \text{that is} \qquad \begin{array}{ccc} & \mathsf{h} & \\ \mathsf{t} & & \mathsf{h} \end{array} \ ,$$

results in the announcement

$$(\mathsf{a}, \mathsf{d}, \mathsf{a}) \qquad \text{that is} \qquad \begin{array}{ccc} & \mathsf{a} & \\ \mathsf{d} & & \mathsf{a} \end{array} \ .$$

This announcement is the only thing the adversary can observe; occurrence of an odd number of d's reveals the presence of a liar, hence the presence of a payer among the cryptographers.

Can the adversary say which cryptographer is paying? No. In fact, given an announcement with an odd number of d's and any payer Crypt_i, we can construct a coin configuration which yields the given announcement. For example, the announcement $(\mathsf{a},\mathsf{d},\mathsf{a})$ above can be yielded by any of the following configurations.

$$\begin{array}{c}\mathsf{Crypt}_0 \text{ pays, and coins are } (\mathsf{h},\mathsf{t},\mathsf{h}) \text{ or } (\mathsf{t},\mathsf{h},\mathsf{t})\\ \mathsf{Crypt}_1 \text{ pays, and coins are } (\mathsf{h},\mathsf{h},\mathsf{h}) \text{ or } (\mathsf{t},\mathsf{t},\mathsf{t})\\ \mathsf{Crypt}_2 \text{ pays, and coins are } (\mathsf{h},\mathsf{h},\mathsf{t}) \text{ or } (\mathsf{t},\mathsf{t},\mathsf{h})\end{array}$$

2.2 Probabilistic anonymity in DC

Up to now the arguments have been non-deterministic: now we shall explain how probabilistic aspects in DC emerge. Assume that the coins are biased: each of three Coin_i's gives head with the probability $9/10$. Provided that Crypt_0 is paying, the announcement $(\mathsf{a},\mathsf{d},\mathsf{a})$ occurs with the probability $(9\cdot 1\cdot 9 + 1\cdot 9\cdot 1)/10^3$, because it results from $(\mathsf{h},\mathsf{t},\mathsf{h})$ or $(\mathsf{t},\mathsf{h},\mathsf{t})$. Similar calculations lead to the following table of probabilities.

	$(\mathsf{d},\mathsf{a},\mathsf{a})$	$(\mathsf{a},\mathsf{d},\mathsf{a})$	$(\mathsf{a},\mathsf{a},\mathsf{d})$	$(\mathsf{d},\mathsf{d},\mathsf{d})$
Crypt_0 pays	0.73	0.09	0.09	0.09
Crypt_1 pays	0.09	0.73	0.09	0.09
Crypt_2 pays	0.09	0.09	0.73	0.09

Are the cryptographers still "anonymous"? We would not say so. For example, if the adversary observes an announcement $(\mathsf{d},\mathsf{a},\mathsf{a})$, it is reasonable for her to suspect Crypt_0 more than the other two.

Nevertheless, if the coins are not biased, we cannot find any symptom of broken anonymity. Therefore we want to obtain the following two things.

The first is an appropriate notion of "probabilistic anonymity" which holds with fair coins but is violated with biased coins—this is done in [4]. The intuition is quite similar to the one behind the notion of *conditional anonymity* [8]. The adversary has a priori knowledge on "who is likely to be blamed"; however, after observing a run of an anonymizing protocol, the adversary should not gain any additional information—each user looks as suspicious as it did before the actual execution.

The second is an effective proof method to verify this notion of anonymity: this is what we aim at in the current work.

3 Probabilistic anonymity

3.1 Anonymity automata: models of anonymizing protocols

In this work anonymizing protocols are formalized as a specific kind of probabilistic systems which we shall call (probabilistic) *anonymity automata*. The notion is similar to probabilistic automata [19]: however, in anonymity automata branching is purely

probabilistic without any non-determinism. This modification, together with other minor ones, is made so that the coalgebraic framework in [9] applies.

The features of an anonymity automaton are as follows.

- By making a transition it can either
  - execute an action and successfully terminate ($x \xrightarrow{a} \checkmark$), or
  - execute an action and move to another state ($x \xrightarrow{a} y$).

  Internal, silent actions are not explicitly present.
- An action a can be either
  - an *observable action* o which can be seen by the adversary, or
  - an *actor action* $\mathsf{blame}(i)$ which denotes that a user i has performed the action whose performer we want to disguise (such as payment in DC).
- Each state comes with a probability subdistribution over the set of possible transitions. By "sub"distribution it is meant that the sum of all the probabilities is ≤ 1 rather than $= 1$: the missing probability is understood as the probability for deadlock.

Here is a formal definition.

Definition 3.1 (Anonymity automata) An *anonymity automaton* is a 5-tuple $(X, \mathcal{U}, \mathcal{O}, c, s)$ where:

- X is a non-empty set called the *state space*.
- $\mathcal{U}$ is a non-empty set of *users*.[4]
- $\mathcal{O}$ is a non-empty set of *observable actions*.
- $c : X \to \mathcal{D}\big(\mathcal{A} \times \{\checkmark\} + \mathcal{A} \times X\big)$ is a function which assigns to each state $x \in X$ a probability subdistribution $c(x)$ over possible transitions. The set $\mathcal{A}$ is the set of *actions* and defined by

  $$\mathcal{A} = \mathcal{O} + \{\, \mathsf{blame}(i) \mid i \in \mathcal{U} \} \ .$$

  The operation $\mathcal{D}$ gives the set of subdistributions: for a set Y,

  $$\mathcal{D}Y = \big\{ d : Y \to [0,1] \mid \sum_{y \in Y} d(y) \leq 1 \big\} \ . \tag{1}$$

  This operation $\mathcal{D}$ canonically extends to a monad[5] which we shall call the *subdistribution monad*.
  For example, the value $c(x)(a, \checkmark)$[6] in $[0,1]$ is the probability with which a state x executes a and then successfully terminate (i.e. $x \xrightarrow{a} \checkmark$).
- s is a probability subdistribution over the state space X. This specifies which state would be a *starting* (or *initial*) one.

[4] A user is called an *anonymous user* in [4].

[5] *Monads* are a categorical notion. Interested readers are referred to [3] for the details.

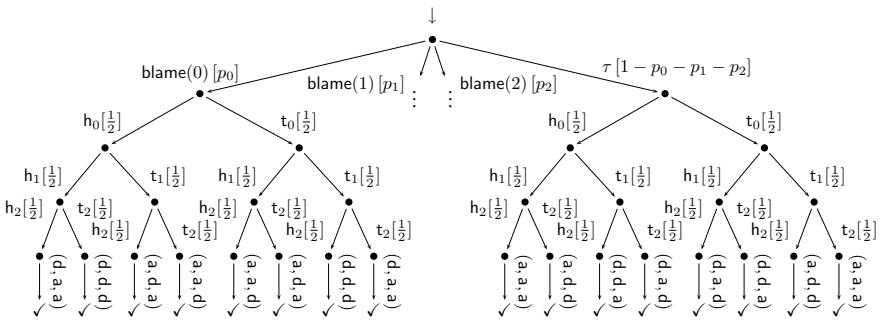
[6] To be precise this should be written as $c(x)\big(\kappa_1(a, \checkmark)\big)$, where $\kappa_1 : \mathcal{A} \times \{\checkmark\} \to \mathcal{A} \times \{\checkmark\} + \mathcal{A} \times X$ is the inclusion map.

Example 3.2 (Anonymity automaton $\mathcal{X}_{\mathrm{DC}}$ for DC) To model the DC protocol, we take

$$\mathcal{U} = \{0, 1, 2\} \ , \quad \mathcal{O} = \{\mathsf{a}, \mathsf{d}\} \times \{\mathsf{a}, \mathsf{d}\} \times \{\mathsf{a}, \mathsf{d}\} = \big\{(\mathsf{x}, \mathsf{y}, \mathsf{z}) \mid \mathsf{x}, \mathsf{y}, \mathsf{z} \in \{\mathsf{a}, \mathsf{d}\}\big\} \ .$$

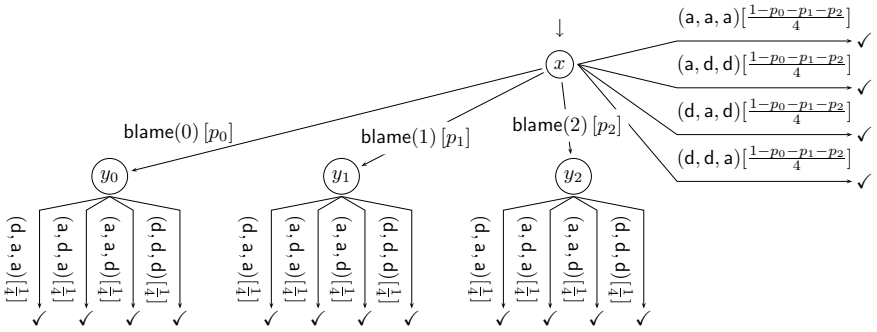
We need to fix the a priori probability distribution on who will make a payment, in view of the conditional notion of probabilistic anonymity. Let us denote by p_i the probability with which a user i pays.

The DC protocol (with its a priori probability distribution given by p_i's) is naturally described as follows. Probability for each transition is presented in square brackets; otherwise the transition occurs with probability 1.



Here τ denotes an internal action with the intention of "NSA pays".

However, the actions h_i and t_i—with their obvious meanings—must not be present because they are not observable by the adversary. These actions are replaced by τ's. Moreover, for technical simplicity we do not allow τ's to appear in an anonymity automaton. Hence we take the "closure" of the above automaton in an obvious way, and obtain the following.



The start state distribution s is: $x \mapsto 1$. This anonymity automaton we shall refer to as $\mathcal{X}_{\mathrm{DC}}$.

3.2 Anonymity automata reconciled as coalgebras

The generic, coalgebraic theory of traces and simulations in [9] applies to anonymity automata. The generic theory is developed with two parameters T and F:

– a monad T on $\mathbf{Sets}$ specifies the *branching-type*, such as non-determinism or probability;
– a functor F on $\mathbf{Sets}$ specifies the *transition-type*, i.e., what a system can do by making a transition.

Systems for which traces/simulations are defined are called (T, F)-*systems* in the generic theory, making the parameters explicit. The theory is coalgebraic because a (T, F)-system is essentially a coalgebra in a suitable category.

Anonymity automata fit in the generic theory. They are (T, F)-systems with the following choice of parameters T and F.

– T is the subdistribution monad $\mathcal{D}$, modeling purely probabilistic branching.
– $FX = \mathcal{A} \times \{\checkmark\} + \mathcal{A} \times X$, modeling the transition-type of "(action and terminate) or (action and next state)".

It is immediately seen that for this choice of F, the set $\mathcal{A}^*\mathcal{A}$ carries the following initial algebra in $\mathbf{Sets}$. We denote its structure map by α.

$$\begin{array}{ccccc} \mathcal{A} \times \{\checkmark\} + \mathcal{A} \times (\mathcal{A}^*\mathcal{A}) & & \kappa_1(a, \checkmark) & & \kappa_2(a, \boldsymbol{a}) \\ \alpha \downarrow \cong & & \downarrow & & \downarrow \\ \mathcal{A}^*\mathcal{A} & & \langle a \rangle & & a \cdot \boldsymbol{a} \end{array} \quad ,$$

where $\langle a \rangle$ denotes a list of length 1, and $a \cdot \boldsymbol{a}$ is what would be written as $(\texttt{cons}\ a\ \boldsymbol{a})$ in LISP. Therefore [9, Corollary 5.2] suggests that the set $\mathcal{A}^*\mathcal{A}$ is the appropriate domain of (finite) trace semantics for anonymity automata: this is actually the case later in Definition 3.3.

3.3 Trace semantics for anonymity automata

The trace semantics for anonymity automata is used in defining probabilistic anonymity. In a non-deterministic setting, trace semantics yields a *set* of lists of actions which can possibly occur. In contrast, trace semantics of a probabilistic system is a *probability subdistribution* over lists.

Definition 3.3 (Trace semantics for anonymity automata) Given an anonymity automaton $\mathcal{X} = (X, \mathcal{U}, \mathcal{O}, c, s)$, its *trace*

$$P_{\mathcal{X}} \in \mathcal{D}(\mathcal{A}^*\mathcal{A})$$

is defined as follows. For a list of actions $\langle a_0, a_1, \ldots, a_n \rangle$ with a finite length $n \geq 1$,

$$P_{\mathcal{X}}(\langle a_0, a_1, \ldots, a_n \rangle) = \sum_{x_0, x_1, \ldots, x_n \in X} P_{\mathcal{X}}(x_0 \xrightarrow{a_0} x_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} x_n \xrightarrow{a_n} \checkmark) \ ,$$

where the probability

$$\begin{aligned} &P_{\mathcal{X}}(x_0 \xrightarrow{a_0} x_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} x_n \xrightarrow{a_n} \checkmark) \\ &= s(x_0) \cdot c(x_0)(a_0, x_1) \cdot \cdots \cdot c(x_{n-1})(a_{n-1}, x_n) \cdot c(x_n)(a_n, \checkmark) \end{aligned}$$

is for the event that $\mathcal{X}$ starts at x_0, follows the path $\xrightarrow{a_0} x_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} x_n$ and finally terminates with $\xrightarrow{a_n} \checkmark$.

Intuitively the value $P_{\mathcal{X}}(\boldsymbol{a}) \in [0,1]$ for a list $\boldsymbol{a} \in \mathcal{A}^*\mathcal{A}$ is the probability with which the system $\mathcal{X}$ executes actions in $\boldsymbol{a}$ successively and then terminates. Our concern is on actions (observable actions or actor actions) the system makes but not on the states it exhibits.

The following alternative characterization allows us to apply the generic, coalgebraic theory of traces in [9,10].

Lemma 3.4 (Trace semantics via the generic theory) *Given an anonymity automaton $\mathcal{X}$, let (s,c) be a (T,F)-system identified with $\mathcal{X}$ as in Section 3.2.*

The trace $P_{\mathcal{X}}$ of $\mathcal{X}$ coincides with the coalgebraic trace $\mathsf{tr}_{(s,c)}$ defined in the generic theory [9, Definition 5.7] for (s,c). □

Example 3.5 (Dining cryptographers) For the anonymity automaton $\mathcal{X}_{\mathrm{DC}}$ in Example 3.2, its trace $P_{\mathcal{X}_{\mathrm{DC}}}$ is the following probability subdistribution.

$$\begin{array}{ll} \langle\, \mathsf{blame}(i), (\mathsf{d},\mathsf{a},\mathsf{a}) \,\rangle \;\mapsto\; p_i/4 & \langle\, (\mathsf{a},\mathsf{a},\mathsf{a}) \,\rangle \;\mapsto\; (1-p_0-p_1-p_2)/4 \\ \langle\, \mathsf{blame}(i), (\mathsf{a},\mathsf{d},\mathsf{a}) \,\rangle \;\mapsto\; p_i/4 & \langle\, (\mathsf{a},\mathsf{d},\mathsf{d}) \,\rangle \;\mapsto\; (1-p_0-p_1-p_2)/4 \\ \langle\, \mathsf{blame}(i), (\mathsf{a},\mathsf{a},\mathsf{d}) \,\rangle \;\mapsto\; p_i/4 & \langle\, (\mathsf{d},\mathsf{a},\mathsf{d}) \,\rangle \;\mapsto\; (1-p_0-p_1-p_2)/4 \\ \langle\, \mathsf{blame}(i), (\mathsf{d},\mathsf{d},\mathsf{d}) \,\rangle \;\mapsto\; p_i/4 & \langle\, (\mathsf{d},\mathsf{d},\mathsf{a}) \,\rangle \;\mapsto\; (1-p_0-p_1-p_2)/4 \end{array}$$
(for $i = 0,1,2$)

The other lists in $\mathcal{A}^*\mathcal{A}$ have probability 0.

In this work we assume that in each execution of an anonymizing protocol there appears at most one actor action. This is the same assumption as [4, Assumption 1] and is true in all the examples in this paper.

Assumption 3.6 (At most one actor action) Let $\mathcal{X} = (X, \mathcal{U}, \mathcal{O}, c, s)$ be an anonymity automaton and $\boldsymbol{a} \in \mathcal{A}^*\mathcal{A}$. If $\boldsymbol{a}$ contains more than one actor actions, then

$$P_{\mathcal{X}}(\boldsymbol{a}) = 0 \ .$$

3.4 Definition of probabilistic anonymity

In this section we formalize the notion of probabilistic anonymity following [4]. First, for the sake of simplicity of presentation, we shall introduce the following notations for predicates (i.e. subsets) on $\mathcal{A}^*\mathcal{A}$.

Definition 3.7 (Predicates $[\mathsf{blame}(i)]$ and $[\boldsymbol{o}]$) – For each $i \in \mathcal{U}$, a predicate $[\mathsf{blame}(i)]$ on $\mathcal{A}^*\mathcal{A}$ is defined as follows.

$$[\mathsf{blame}(i)] = \{\boldsymbol{a} \in \mathcal{A}^*\mathcal{A} \mid \mathsf{blame}(i) \text{ appears in } \boldsymbol{a}\}$$

By Assumption 3.6, it is the set of lists obtained by augmenting $\mathsf{blame}(i)$ with observable actions: in a regular-expression-like notation,

$$[\mathsf{blame}(i)] = \mathcal{O}^*\, \mathsf{blame}(i)\, \mathcal{O}^* \ .$$

Moreover, $[\mathsf{blame}(i)] \cap [\mathsf{blame}(j)] = \emptyset$ if $i \neq j$.

– For each $\boldsymbol{o} \in \mathcal{O}^*$, a predicate $[\boldsymbol{o}]$ on $\mathcal{A}^*\mathcal{A}$ is defined as follows.

$$[\boldsymbol{o}] = \{\boldsymbol{a} \in \mathcal{A}^*\mathcal{A} \mid \mathsf{removeActor}(\boldsymbol{a}) = \boldsymbol{o}\} \ ,$$

where the function $\mathsf{removeActor} : \mathcal{A}^*\mathcal{A} \to \mathcal{O}^*$—which is defined by a suitable induction—removes actor actions appearing in a list. The set $[\boldsymbol{o}] \subseteq \mathcal{A}^*\mathcal{A}$ consists of those lists which yield $\boldsymbol{o}$ as the adversary's observation. It is emphasized that $[\boldsymbol{o}]$ is *not* the set of lists which contain $\boldsymbol{o}$ as sublists: we remove only actor actions, but not observable actions.

Note that we are overriding the notation $[\_]$: no confusion would arise since the arguments are of different types. Values such as $P_{\mathcal{X}}(\,[\mathsf{blame}(i)]\,)$ are defined in a straightforward manner:

$$P_{\mathcal{X}}(\,[\mathsf{blame}(i)]\,) = \sum_{\boldsymbol{a} \in [\mathsf{blame}(i)]} P_{\mathcal{X}}(\boldsymbol{a}) \ .$$

This is the probability with which $\mathcal{X}$ yields an execution in which a user i is to be blamed.

We follow [4] and adopt the following definition of anonymity.

Definition 3.8 (Probabilistic anonymity [4]) We say an anonymity automaton $\mathcal{X}$ is *anonymous* if for each $i, j \in \mathcal{U}$ and $\boldsymbol{o} \in \mathcal{O}^*$,

$$\begin{aligned} & P_{\mathcal{X}}(\,[\mathsf{blame}(i)]\,) > 0 \quad \wedge \quad P_{\mathcal{X}}(\,[\mathsf{blame}(j)]\,) > 0 \\ \Longrightarrow \quad & P_{\mathcal{X}}(\,[\boldsymbol{o}] \mid [\mathsf{blame}(i)]\,) = P_{\mathcal{X}}(\,[\boldsymbol{o}] \mid [\mathsf{blame}(j)]\,) \ . \end{aligned}$$

Here $P_{\mathcal{X}}(\,[\boldsymbol{o}] \mid [\mathsf{blame}(i)]\,)$ is a conditional probability: it is given by

$$P_{\mathcal{X}}(\,[\boldsymbol{o}] \mid [\mathsf{blame}(i)]\,) = \frac{P_{\mathcal{X}}(\,[\boldsymbol{o}] \cap [\mathsf{blame}(i)]\,)}{P_{\mathcal{X}}(\,[\mathsf{blame}(i)]\,)} \ .$$

The intuition behind this notion—sketched in Section 2.2—is similar to the one behind conditional anonymity [8]. In fact, it is shown in [4] that under reasonable assumptions the two notions of anonymity coincide. For completeness the definition of conditional anonymity (adapted to the current setting) is also presented.

Definition 3.9 (Conditional anonimity [8]) An anonymity automaton $\mathcal{X}$ satisfies *conditional anonymity* if for each $i \in \mathcal{U}$ and $\boldsymbol{o}, \boldsymbol{o}' \in \mathcal{O}^*$,

$$\begin{aligned} & P_{\mathcal{X}}(\,[\mathsf{blame}(i)] \cap [\boldsymbol{o}]\,) > 0 \\ \Longrightarrow \quad & P_{\mathcal{X}}(\,[\mathsf{blame}(i)] \mid [\boldsymbol{o}]\,) = P_{\mathcal{X}}(\,[\mathsf{blame}(i)] \mid \bigcup_{j \in \mathcal{U}} [\mathsf{blame}(j)]\,) \ . \end{aligned}$$

The notion in Definition 3.8 is (one possibility of) probabilistic extension of *trace anonymity* in [18]. It is emphasized that these anonymity notions are based on trace semantics which is at the coarsest end in the linear time-branching time spectrum [22]. Hence our adversary has less observation power than one in [1] for example where security notions are bisimulation-based. A justification for having such a weaker adversary is found in [13].

4 Anonymity proof via probabilistic simulations

In this section we extend the proof method [13,12] for anonymity to the probabilistic setting. In the introduction we have presented the basic scenario. Now we shall describe its details, with all the notions therein (traces, simulations, *etc.*) interpreted probabilistically.

4.1 Anonymized automaton $\mathsf{an}(\mathcal{X})$

We start with the definition of $\mathsf{an}(\mathcal{X})$, the *anonymized version* of an anonymity automaton $\mathcal{X}$. Recall that our notion of anonymity is conditional: the adversary has a priori knowledge on who is more suspicious. In an anonymity automaton $\mathcal{X}$, the a priori probability with which a user i does wrong is given by $P_{\mathcal{X}}(\,[\mathsf{blame}(i)]\,)$. Its normalized, conditional version

$$r_i \stackrel{\text{def.}}{=} P_{\mathcal{X}}(\,[\mathsf{blame}(i)] \mid \bigcup_{j\in\mathcal{U}} [\mathsf{blame}(j)]\,) = \frac{P_{\mathcal{X}}(\,[\mathsf{blame}(i)] \cap \bigcup_{j\in\mathcal{U}} [\mathsf{blame}(j)]\,)}{P_{\mathcal{X}}(\,\bigcup_{j\in\mathcal{U}} [\mathsf{blame}(j)]\,)} = \frac{P_{\mathcal{X}}(\,[\mathsf{blame}(i)]\,)}{\sum_{j\in\mathcal{U}} P_{\mathcal{X}}(\,[\mathsf{blame}(j)]\,)}$$

(the equalities are due to Assumption 3.6) plays an important role in the following definition of $\mathsf{an}(\mathcal{X})$. The value r_i is the conditional probability with which a user i is to be blamed, given that there is any user to be blamed; we have $\sum_{i\in\mathcal{U}} r_i = 1$. Of course, for the values r_i to be well-defined, the anonymity automaton $\mathcal{X}$ needs to satisfy the following reasonable assumption.

Assumption 4.1 (There is someone to blame) For an anonymity automaton $\mathcal{X}$,

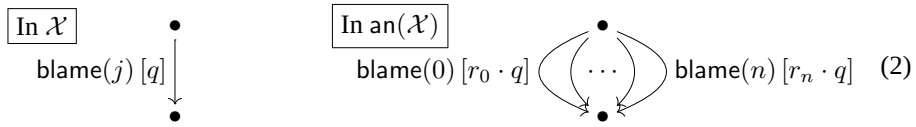
$$\sum_{j\in\mathcal{U}} P_{\mathcal{X}}(\,[\mathsf{blame}(j)]\,) \neq 0 \ .$$

Intuitively, $\mathsf{an}(\mathcal{X})$ is obtained from $\mathcal{X}$ by distributing an actor action $\mathsf{blame}(i)$ to each user j, with the probability distributed in proportion to r_j.

Definition 4.2 (Anonymized anonymity automaton $\mathsf{an}(\mathcal{X})$) Given an anonymity automaton $\mathcal{X} = (X, \mathcal{U}, \mathcal{O}, c, s)$, its *anonymized automaton* $\mathsf{an}(\mathcal{X})$ is a 5-tuple $(X, \mathcal{U}, \mathcal{O}, c^{\mathsf{an}}, s)$, where c^{an} is defined as follows. For each $x \in X$,

$$\begin{array}{ll} c^{\mathsf{an}}(x)(\mathsf{blame}(i), u) = \sum_{j\in\mathcal{U}} r_i \cdot c(x)(\mathsf{blame}(j), u) & \text{for } i \in \mathcal{U} \text{ and } u \in \{\checkmark\} + X, \\ c^{\mathsf{an}}(x)(o, u) = c(x)(o, u) & \text{for } o \in \mathcal{O} \text{ and } u \in \{\checkmark\} + X. \end{array}$$

On the first equation, the summand $r_i \cdot c(x)(\mathsf{blame}(j), u)$ results from distributing the probability $c(x)(\mathsf{blame}(j), u)$ for a transition $x \xrightarrow{\mathsf{blame}(j)} u$, to a user i. This is illustrated in the following figure: here $\mathcal{U} = \{0, 1, \ldots, n\}$ and $q = c(x)(\mathsf{blame}(j), u)$.



(2)

The automaton $\mathsf{an}(\mathcal{X})$ is "anonymized" in the sense of the following lemmas.

Lemma 4.3 *Let $\mathcal{X}$ be an anonymity automaton. In its anonymized version $\mathsf{an}(\mathcal{X}) = (X, \mathcal{U}, \mathcal{O}, c^{\mathsf{an}}, s)$ we have*

$$r_j \cdot c^{\mathsf{an}}(x)(\mathsf{blame}(i), u) = r_i \cdot c^{\mathsf{an}}(x)(\mathsf{blame}(j), u)$$

for any $i, j \in \mathcal{U}$, $x \in X$ and $u \in \{\checkmark\} + X$.

Proof. Obvious from the definition of c^{an}. □

Lemma 4.4 ($\mathsf{an}(\mathcal{X})$ is anonymous) *For an anonymity automaton $\mathcal{X}$, $\mathsf{an}(\mathcal{X})$ is anonymous in the sense of Definition 3.8.*

Proof. Let $\boldsymbol{o} = \langle o_1, o_2, \ldots, o_n \rangle \in \mathcal{O}^*$ and $i, j \in \mathcal{U}$. Moreover, assume

$$P_{\mathsf{an}(\mathcal{X})}(\,[\mathsf{blame}(i)]\,) \neq 0 \quad \text{and} \quad P_{\mathsf{an}(\mathcal{X})}(\,[\mathsf{blame}(j)]\,) \neq 0 \ ,$$

hence $r_i \neq 0$ and $r_j \neq 0$. Then

$$\begin{aligned}
&P_{\mathsf{an}(\mathcal{X})}(\,[\boldsymbol{o}] \cap [\mathsf{blame}(i)]\,) \\
&= P_{\mathsf{an}(\mathcal{X})}(\,\langle \mathsf{blame}(i), o_1, o_2, \ldots, o_n \rangle\,) \\
&\quad + P_{\mathsf{an}(\mathcal{X})}(\,\langle o_1, \mathsf{blame}(i), o_2, \ldots, o_n \rangle\,) \\
&\quad + \cdots + P_{\mathsf{an}(\mathcal{X})}(\,\langle o_1, o_2, \ldots, o_n, \mathsf{blame}(i) \rangle\,) \\
&= \sum_{x_0, x_1, \ldots, x_n \in X} s(x_0) \cdot c^{\mathsf{an}}(x_0)(\mathsf{blame}(i), x_1) \cdot c^{\mathsf{an}}(x_1)(o_1, x_2) \cdot \cdots \cdot c^{\mathsf{an}}(x_n)(o_n, \checkmark) \\
&\quad + \sum_{x_0, x_1, \ldots, x_n \in X} s(x_0) \cdot c^{\mathsf{an}}(x_0)(o_1, x_1) \cdot c^{\mathsf{an}}(x_1)(\mathsf{blame}(i), x_2) \cdot \cdots \cdot c^{\mathsf{an}}(x_n)(o_n, \checkmark) \\
&\quad + \cdots \\
&\quad + \sum_{x_0, x_1, \ldots, x_n \in X} s(x_0) \cdot c^{\mathsf{an}}(x_0)(o_1, x_1) \cdot c^{\mathsf{an}}(x_1)(o_2, x_2) \cdot \cdots \cdot c^{\mathsf{an}}(x_n)(\mathsf{blame}(i), \checkmark) \ .
\end{aligned}$$

We have the same equation for j instead of i. Hence by Lemma 4.3 we have

$$r_j \cdot P_{\mathsf{an}(\mathcal{X})}(\,[\boldsymbol{o}] \cap [\mathsf{blame}(i)]\,) = r_i \cdot P_{\mathsf{an}(\mathcal{X})}(\,[\boldsymbol{o}] \cap [\mathsf{blame}(j)]\,) \ . \tag{3}$$

This is used to show the equality of two conditional probabilities.

$$\begin{aligned}
P_{\mathsf{an}(\mathcal{X})}(\,[\boldsymbol{o}] \mid [\mathsf{blame}(i)]\,) &= \frac{P_{\mathsf{an}(\mathcal{X})}(\,[\boldsymbol{o}] \cap [\mathsf{blame}(i)]\,)}{P_{\mathsf{an}(\mathcal{X})}(\,[\mathsf{blame}(i)]\,)} \\
&= \frac{r_i}{r_j} \cdot \frac{P_{\mathsf{an}(\mathcal{X})}(\,[\boldsymbol{o}] \cap [\mathsf{blame}(j)]\,)}{P_{\mathsf{an}(\mathcal{X})}(\,[\mathsf{blame}(i)]\,)} \qquad \text{By (3)} \\
&= \frac{P_{\mathsf{an}(\mathcal{X})}(\,[\boldsymbol{o}] \cap [\mathsf{blame}(j)]\,)}{P_{\mathsf{an}(\mathcal{X})}(\,[\mathsf{blame}(j)]\,)} \qquad \text{By definition of } r_i, r_j \\
&= P_{\mathsf{an}(\mathcal{X})}(\,[\boldsymbol{o}] \mid [\mathsf{blame}(j)]\,) \ .
\end{aligned}$$

□

4.2 Forward/backward simulations for anonymity automata

We proceed to introduce appropriate notions of forward and backward simulations. The (tedious) definition and soundness theorem—existence of a forward/backward simulation implies trace inclusion—come for free from the generic theory in [9]. This forms a crucial part of our simulation-based proof method.

Definition 4.5 (Forward/backward simulations for anonymity automata) Let $\mathcal{X} = (X, \mathcal{U}, \mathcal{O}, c, s)$ and $\mathcal{Y} = (Y, \mathcal{U}, \mathcal{O}, d, t)$ be anonymity automata which have the same sets of users and observable actions.

A *forward simulation* from $\mathcal{X}$ to $\mathcal{Y}$—through which $\mathcal{Y}$ *forward-simulates* $\mathcal{X}$—is a function

$$f : Y \longrightarrow \mathcal{D}X$$

which satisfies the following inequalities in $[0, 1]$.

$$\begin{array}{lll} s(x) & \leq \sum_{y \in Y} t(y) \cdot f(y)(x) & \text{for any } x \in X, \\ \sum_{x \in X} f(y)(x) \cdot c(x)(e, \checkmark) & \leq d(y)(e, \checkmark) & \text{for any } y \in Y \text{ and } e \in \mathcal{A}, \\ \sum_{x \in X} f(y)(x) \cdot c(x)(e, x') & \leq \sum_{y' \in Y} d(y)(e, y') \cdot f(y')(x') & \\ & & \text{for any } y \in Y,\ e \in \mathcal{A} \text{ and } x' \in X. \end{array}$$

A *backward simulation* from $\mathcal{X}$ to $\mathcal{Y}$—through which $\mathcal{Y}$ *backward-simulates* $\mathcal{X}$—is a function

$$b : X \longrightarrow \mathcal{D}Y$$

which satisfies the following inequalities in $[0, 1]$.

$$\begin{array}{lll} \sum_{x \in X} s(x) \cdot b(x)(y) & \leq t(y) & \text{for any } y \in Y, \\ c(x)(e, \checkmark) & \leq \sum_{y \in Y} b(x)(y) \cdot d(y)(e, \checkmark) & \text{for any } x \in X \text{ and } e \in \mathcal{A}, \\ \sum_{x' \in X} c(x)(e, x') \cdot b(x')(y') & \leq \sum_{y \in Y} b(x)(y) \cdot d(y)(e, y') & \\ & & \text{for any } x \in X,\ e \in \mathcal{A} \text{ and } y' \in Y. \end{array}$$

The definition definitely looks puzzling. Why does a forward simulation have the type $Y \to \mathcal{D}X$? Why is a backward simulation not of the same type? How come the complex inequalities? How do we know that the inequalities are in the correct direction?

In fact, this definition is an instantiation of the general, coalgebraic notions of forward/backward simulations [9, Definitions 4.1, 4.2]. More specifically, the two parameters T and F in the generic definition are instantiated as in Section 3.2.

Theorem 4.6 (Soundness of forward/backward simulations) *Assume there is a forward (or backward) simulation from one anonymity automaton $\mathcal{X}$ to another $\mathcal{Y}$. Then we have trace inclusion*

$$P_{\mathcal{X}} \sqsubseteq P_{\mathcal{Y}} \ ,$$

where the order $\sqsubseteq$ is defined to be the pointwise order: for each $\boldsymbol{a} \in \mathcal{A}^*\mathcal{A}$,

$$P_{\mathcal{X}}(\boldsymbol{a}) \leq P_{\mathcal{Y}}(\boldsymbol{a}) \ .$$

Proof. We know (Lemma 3.4) that the notions of traces and simulations for anonymity automata are instantiations of the general, coalgebraic notions in [9,10]. Therefore we can appeal to the general soundness theorem [9, Theorem 6.1]. □

4.3 Probabilistic anonymity via simulations

We shall use the materials in Sections 4.1 and 4.2 to prove the validity of our simulation-based proof method (Theorem 4.11).

The following lemma—which essentially says $P_{\mathcal{X}} \sqsubseteq P_{\mathsf{an}(\mathcal{X})}$—relies on the way $\mathsf{an}(\mathcal{X})$ is constructed. The proof is a bit more complicated than in the non-deterministic setting [13,12].

Lemma 4.7 *Let $\mathcal{X}$ be an anonymity automaton. Assume there exists a forward or backward simulation from $\mathsf{an}(\mathcal{X})$ to $\mathcal{X}$—through which $\mathcal{X}$ simulates $\mathsf{an}(\mathcal{X})$. Then their trace semantics are equal:*

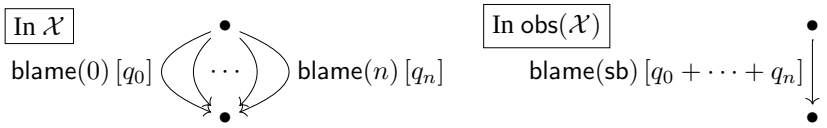
$$P_{\mathcal{X}} = P_{\mathsf{an}(\mathcal{X})} \ .$$

Proof. By the soundness theorem (Theorem 4.6) we have

$$P_{\mathcal{X}} \sqsupseteq P_{\mathsf{an}(\mathcal{X})} \ , \tag{4}$$

where $\sqsupseteq$ refers to the pointwise order between functions $\mathcal{A}^*\mathcal{A} \rightrightarrows [0,1]$. We shall show that this inequality is in fact an equality.

First we introduce an operation obs which acts on anonymity automata. Intuitively, $\mathsf{obs}(\mathcal{Y})$ is obtained from $\mathcal{Y}$ by replacing all the different actor actions $\mathsf{blame}(i)$ with single $\mathsf{blame}(\mathsf{sb})$—$\mathsf{sb}$ is for "somebody". This conceals actor actions in $\mathcal{Y}$; hence $\mathsf{obs}(\mathcal{Y})$ only carries information on the observable actions of $\mathcal{Y}$.



(5)

Formally,

Definition 4.8 (Anonymity automaton $\mathsf{obs}(\mathcal{Y})$) Given an anonymity automaton $\mathcal{Y} = (Y, \mathcal{U}, \mathcal{O}, d, t)$, we define an anonymity automaton $\mathsf{obs}(\mathcal{Y})$ as the 5-tuple $(Y, \{\mathsf{sb}\}, \mathcal{O}, d^{\mathsf{obs}}, t)$ where:

- sb is a fresh entity,
- d^{obs} is a function

$$d^{\mathsf{obs}} : Y \longrightarrow \mathcal{D}\big(\mathcal{A}^{\mathsf{obs}} \times \{\checkmark\} + \mathcal{A}^{\mathsf{obs}} \times Y\big)$$

where $\mathcal{A}^{\mathsf{obs}} = \mathcal{O} + \{\mathsf{blame}(\mathsf{sb})\}$, defined by:

$$\begin{array}{lll} d^{\mathsf{obs}}(y)(\mathsf{blame}(\mathsf{sb}), u) = \sum_{i \in \mathcal{U}} d(y)(\mathsf{blame}(i), u) & & \text{for } y \in Y \text{ and } u \in \{\checkmark\} + Y, \\ d^{\mathsf{obs}}(y)(o, u) = d(y)(o, u) & & \text{for } y \in Y,\ o \in \mathcal{O} \text{ and } u \in \{\checkmark\} + Y. \end{array}$$

The following fact is obvious.

Sublemma 4.9 *For an anonymity automaton $\mathcal{X}$, $\mathsf{obs}(\mathcal{X})$ and $\mathsf{obs}(\mathsf{an}(\mathcal{X}))$ are identical.* □

The following sublemma is crucial in the proof of Lemma 4.7. Two automata $\mathcal{Y}$ and $\mathsf{obs}(\mathcal{Y})$, although their trace semantics distributes over different sets, have the same sum of probabilities taken over all executions.

Sublemma 4.10 *For an anonymity automaton $\mathcal{Y}$,*

$$\sum_{\boldsymbol{a}\in\mathcal{A}^*\mathcal{A}} P_{\mathcal{Y}}(\boldsymbol{a}) = \sum_{\boldsymbol{a}'\in(\mathcal{A}^{\mathsf{obs}})^*\mathcal{A}^{\mathsf{obs}}} P_{\mathsf{obs}(\mathcal{Y})}(\boldsymbol{a}') \ .$$

Recall that $\mathcal{A} = \mathcal{O} + \{\mathsf{blame}(i) \mid i \in \mathcal{U}\}$ and $\mathcal{A}^{\mathsf{obs}} = \mathcal{O} + \{\mathsf{blame}(\mathsf{sb})\}$.

Proof. From the definition of trace semantics (Definition 3.3), the sublemma is proved by easy calculation. □

We turn back to the proof of Lemma 4.7. We argue by contradiction—assume that the inequality in (4) is strict. That is, there exists $\boldsymbol{a}_0 \in \mathcal{A}^*\mathcal{A}$ such that $P_{\mathcal{X}}(\boldsymbol{a}_0) \gneq P_{\mathsf{an}(\mathcal{X})}(\boldsymbol{a}_0)$. Then, by (4) we have $\sum_{\boldsymbol{a}\in\mathcal{A}^*\mathcal{A}} P_{\mathcal{X}}(\boldsymbol{a}) \gneq \sum_{\boldsymbol{a}\in\mathcal{A}^*\mathcal{A}} P_{\mathsf{an}(\mathcal{X})}(\boldsymbol{a})$. However,

$$\begin{aligned}
\textstyle\sum_{\boldsymbol{a}\in\mathcal{A}^*\mathcal{A}} P_{\mathcal{X}}(\boldsymbol{a}) &= \textstyle\sum_{\boldsymbol{a}'\in(\mathcal{A}^{\mathsf{obs}})^*\mathcal{A}^{\mathsf{obs}}} P_{\mathsf{obs}(\mathcal{X})}(\boldsymbol{a}') && \text{By Sublemma 4.10}\\
&= \textstyle\sum_{\boldsymbol{a}'\in(\mathcal{A}^{\mathsf{obs}})^*\mathcal{A}^{\mathsf{obs}}} P_{\mathsf{obs}(\mathsf{an}(\mathcal{X}))}(\boldsymbol{a}') && \text{By Sublemma 4.9}\\
&= \textstyle\sum_{\boldsymbol{a}\in\mathcal{A}^*\mathcal{A}} P_{\mathsf{an}(\mathcal{X})}(\boldsymbol{a}) \ . && \text{By Sublemma 4.10}
\end{aligned}$$

This contradiction concludes the proof of Lemma 4.7. □

Now we are ready to state the main result.

Theorem 4.11 (Main theorem: probabilistic anonymity via simulations) *If there exists a forward or backward simulation from $\mathsf{an}(\mathcal{X})$ to $\mathcal{X}$, then $\mathcal{X}$ is anonymous.*

Proof. By Lemma 4.7 we have $P_{\mathcal{X}} = P_{\mathsf{an}(\mathcal{X})}$. Moreover, by Lemma 4.4, $\mathsf{an}(\mathcal{X})$ is anonymous. This proves anonymity of $\mathcal{X}$: recall that probabilistic anonymity is a property defined in terms of traces (Definition 3.8). □

Example 4.12 (Dining cryptographers) We demonstrate our proof method via simulations by applying it to the DC protocol.

Let $X = \{x, y_0, y_1, y_2\}$ be the state space of $\mathcal{X}_{\mathrm{DC}}$. Its anonymized version $\mathsf{an}(\mathcal{X}_{\mathrm{DC}})$ has the same state space: for notational convenience the state space of $\mathsf{an}(\mathcal{X}_{\mathrm{DC}})$ is denoted by $X' = \{x', {y_0}', {y_1}', {y_2}'\}$. It is verified by easy calculation that the following function $f : X \to \mathcal{D}(X')$ is a forward simulation from $\mathsf{an}(\mathcal{X}_{\mathrm{DC}})$ to $\mathcal{X}_{\mathrm{DC}}$.

$$f(x) = [x' \mapsto 1] \qquad f(y_0) = f(y_1) = f(y_2) = \begin{bmatrix} {y_0}' \mapsto \frac{p_0}{p_0+p_1+p_2} \\ {y_1}' \mapsto \frac{p_1}{p_0+p_1+p_2} \\ {y_2}' \mapsto \frac{p_2}{p_0+p_1+p_2} \end{bmatrix}$$

By Theorem 4.11 this proves (probabilistic) anonymity of $\mathcal{X}_{\mathrm{DC}}$, hence of the DC protocol.

5 Conclusion and future work

We have extended the simulation-based proof method [13,12] for non-deterministic anonymity to apply to the notion of probabilistic anonymity defined in [4]. For the move we have exploited a generic theory of traces and simulations [9,10] in which the difference between non-determinism and probability is just a different choice of a parameter.

The DC example in this paper fails to demonstrate the usefulness of our proof method: for this small example direct calculation of trace distribution is not hard. A real benefit would arise in theorem-proving anonymity of an unboundedly large system (which we cannot model-check). In fact, the non-deterministic version of our proof method is used to theorem-prove anonymity of a voting protocol with arbitrary many voters [12]. A probabilistic case study of such kind is currently missing.

In [4] the probabilistic π-calculus is utilized as a specification language for automata. We have not yet elaborated which subset of the calculus is suitable for describing our notion of anonymity automata.

There is a well-established body of work on verification of probabilistic information-hiding properties such as non-interference [24,17]. Our proof method could be reconciled in this context by, for example, finding a translation of anonymity into a non-interference property.

The significance of having both non-deterministic and probabilistic branching in considering anonymity is claimed in [15]. However the current method cannot handle this combination due to the lack of suitable coalgebraic framework. Elaboration in this direction would also help better understanding of the nature of the (notorious) combination of non-determinism and probability.

Acknowledgments Thanks are due to Ken Mano, Peter van Rossum, Hideki Sakurada, Ana Sokolova, Yasuaki Tsukada and the anonymous referees for helpful discussions and comments. The first author is grateful to his supervisor Bart Jacobs for encouragement.

References

1. M. Abadi and A. Gordon. A calculus for cryptographic protocols: The Spi calculus. In *Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.
2. Anonymity bibliography. `http://freehaven.net/anonbib/`.
3. M. Barr and C. Wells. *Toposes, Triples and Theories*. Springer, Berlin, 1985.
4. M. Bhargava and C. Palamidessi. Probabilistic anonymity. In M. Abadi and L. de Alfaro, editors, *CONCUR 2005*, volume 3653 of *Lect. Notes Comp. Sci.*, pages 171–185. Springer, 2005.
5. D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journ. of Cryptology*, 1(1):65–75, 1988.
6. L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices*. PhD thesis, Radboud Univ. Nijmegen, 2006.

7. F.D. Garcia, I. Hasuo, W. Pieters, and P. van Rossum. Provable anonymity. In R. Küsters and J. Mitchell, editors, *3rd ACM Workshop on Formal Methods in Security Engineering (FMSE05)*, pages 63–72, Alexandria , VA, U.S.A., November 2005. ACM Press.
8. J.Y. Halpern and K.R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, to appear.
9. I. Hasuo. Generic forward and backward simulations. In C. Baier and H. Hermanns, editors, *International Conference on Concurrency Theory (CONCUR 2006)*, volume 4137 of *Lect. Notes Comp. Sci.*, pages 406–420. Springer, Berlin, 2006.
10. I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace theory. In N. Ghani and J. Power, editors, *International Workshop on Coalgebraic Methods in Computer Science (CMCS 2006)*, volume 164 of *Elect. Notes in Theor. Comp. Sci.*, pages 47–65. Elsevier, Amsterdam, 2006.
11. D. Hughes and V. Shmatikov. Information hiding, anonymity and privacy: A modular approach. *Journal of Computer Security*, 12(1):3–36, 2004.
12. Y. Kawabe, K. Mano, H. Sakurada, and Y. Tsukada. Backward simulations for anonymity. In *International Workshop on Issues in the Theory of Security (WITS '06)*, 2006.
13. Y. Kawabe, K. Mano, H. Sakurada, and Y. Tsukada. Theorem-proving anonymity of infinite state systems. *Information Processing Letters*, 101(1), 2007.
14. N. Lynch and F. Vaandrager. Forward and backward simulations. I. Untimed systems. *Inf. & Comp.*, 121(2):214–233, 1995.
15. C. Palamidessi. Probabilistic and nondeterministic aspects of anonymity. In *MFPS '05*, volume 155 of *Elect. Notes in Theor. Comp. Sci.*, pages 33–42. Elsevier, 2006.
16. A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity: A proposal for terminology. Draft, version 0.17, July 2000.
17. A. Sabelfeld and D. Sands. Probabilistic noninterference for multi-threaded programs. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop (CSFW'00)*, pages 200–214, 2000.
18. S. Schneider and A. Sidiropoulos. CSP and anonymity. In *ESORICS '96: Proceedings of the 4th European Symposium on Research in Computer Security*, pages 198–218, London, UK, 1996. Springer-Verlag.
19. R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journ. Comput.*, 2(2):250–273, 1995.
20. A. Serjantov. *On the Anonymity of Anonymity Systems*. PhD thesis, University of Cambridge, March 2004.
21. V. Shmatikov. Probabilistic model checking of an anonymity system. *Journ. of Computer Security*, 12(3):355–377, 2004.
22. R. van Glabbeek. The linear time-branching time spectrum (extended abstract). In J. Baeten and J. Klop, editors, Proceedings *CONCUR '90, Theories of Concurrency: Unification and Extension*, Amsterdam, August 1990, volume 458 of *Lect. Notes Comp. Sci.*, pages 278–297. Springer-Verlag, 1990.
23. D. Varacca and G. Winskel. Distributing probabililty over nondeterminism. *Math. Struct. in Comp. Sci.*, 16(1):87–113, 2006.
24. D.M. Volpano and G. Smith. Probabilistic noninterference in a concurrent language. *Journ. of Computer Security*, 7(1), 1999.