Probabilistic Anonymity via Coalgebraic Simulations[1]

Ichiro Hasuo[a,b,*,2] Yoshinobu Kawabe[c,3] Hideki Sakurada[d]

[a] *Research Institute for Mathematical Sciences, Kyoto University.*

[b] *PRESTO Research Promotion Program, Japan Science and Technology Agency.*

[c] *Department of Information Science, Aichi Institute of Technology.*

[d] *NTT Communication Science Laboratories, NTT Corporation.*

Abstract

There is a growing concern about anonymity and privacy on the Internet, resulting in lots of work on formalization and verification of anonymity. Especially, the importance of probabilistic aspect of anonymity is claimed recently by many authors. Several different notions of "probabilistic anonymity" have been studied so far, but proof methods for such probabilistic notions are not yet elaborated. In this paper we introduce a simulation-based proof method for one notion of probabilistic anonymity introduced by Bhargava and Palamidessi, called *strong probabilistic anonymity*. The method is a probabilistic adaptation of the one by Kawabe, Sakurada 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 exploit a generic, coalgebraic theory of traces and simulations developed by Hasuo, Jacobs and Sokolova. In particular, an appropriate notion of probabilistic simulation is obtained as an instantiation of the generic definition, for which soundness theorem comes for free. Additionally, we show how we can use a similar idea to verify a weaker notion of probabilistic anonymity called *probable innocence*.

Key words: Anonymity, Automaton, Coalgebra, Privacy, Probability

1 Introduction

Nowadays more and more human activities are relying critically 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, varying purposes of communication protocols have identified new verification goals—or *security properties*—such as anonymity, in addition to rather traditional ones like secrecy and 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 an increasing extent of research activities—especially from the formal methods community—are aiming at verification of anonymity properties (see [3]).

Formal verification of anonymity properties is at its relative youth compared to authentication or secrecy. The topic still allows for definitional work (such as [5,8,17,19,26,33]) pointing out many different aspects of anonymity notions. Notably many authors [5,8,19,40,41] claim the significant role of *probability* in anonymity notions. This is the focus of this paper.

There have been several different notions of anonymity proposed in probabilistic settings, e.g. [34,19,5]. There are also some case studies which have analyzed existing anonymizing protocols to see whether they satisfy these notions of probabilistic anonymity. However, generic verification methods for probabilistic anonymity have not yet much elaborated.

In this paper we introduce simulation-based proof methods for two differ-

* Corresponding author. Research Institute for Mathematical Sciences, Kyoto University, 606-8502 Japan. Tel: +81 75 753 7251, fax: +81 75 753 7266.

Email addresses: `ichiro@kurims.kyoto-u.ac.jp` (Ichiro Hasuo), `kawabe@aitech.ac.jp` (Yoshinobu Kawabe), `sakurada@theory.brl.ntt.co.jp` (Hideki Sakurada).

URLs: `http://www.kurims.kyoto-u.ac.jp/~ichiro` (Ichiro Hasuo), `http://aitech.ac.jp/~kawabe` (Yoshinobu Kawabe), `http://www.brl.ntt.co.jp/people/sakurada` (Hideki Sakurada).

[1] An earlier version [24] of this paper is presented at the 16th European Symposium on Programming (ESOP 2007), Braga, Portugal, March 2007.

[2] This work was done during the first author's stay at NTT Communication Science Laboratories in September–October 2006 and April 2007, in the period of his employment at Radboud University Nijmegen, the Netherlands.

[3] Partly supported by the Grant-in-Aid for Young Scientists (B), No.19700018, of the Ministry of Education, Culture, Sports, Science and Technology, Japan. This work was done during the second author's employment at NTT Communication Science Laboratories, NTT Corporation.

ent notions of probabilistic anonymity. The first one is *strong probabilistic anonymity* (which will be also called *strong anonymity*) introduced by Bhargava and Palamidessi [5]; the other one is *probable innocence* found e.g. in [34,19]. Strong anonymity requires that, by observing an execution of the protocol, the adversary should gain absolutely no information on who is the person to be blamed—the *culprit.* The notion is a natural probabilistic extension of the non-deterministic notion of *trace anonymity* [4].

What we shall do in this paper is to extend a proof method as well, from the one for (non-deterministic) trace anonymity to the one for probabilistic strong anonymity. The proof method we start with is the simulation-based one which is introduced in [28,27]. Its basic scenario is as follows.

(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 for simulations—existence of a simulation implies trace inclusion.

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

The basic scenario remains the same for strong (probabilistic) anonymity. However, there is an obvious difficulty in conducting it in a probabilistic setting. The theory of traces and simulations in a non-deterministic setting is well studied e.g. by [29]; 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 [20,23]. In the generic theory, fundamental notions such as system

[4] The notion of trace anonymity is originally introduced in [37] under the name of *strong anonymity.* In this paper we follow [28,27] and call it trace anonymity. This is for the sake of terminological convenience.

(or automaton), trace semantics and forward/backward simulation 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 generic theory has two parameters appearing in it; by making different choices of these parameters the theory can cover a wide variety of systems. In particular, according to the choice of one parameter, systems can be non-deterministic or probabilistic. [5] In this work we obtain a complex definition of probabilistic simulations 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.

After presenting a simulation-based proof method for strong (probabilistic) anonymity, we use a similar idea to verify a weaker anonymity notion called *probable innocence*. The notion appears in [34,19] and is extensively studied in [9]. The significance of this weaker notion is due to the fact that many anonymizing protocols—such as Crowds [34]—do not satisfy strong anonymity but do satisfy probable innocence. Intuitively, probable innocence does allow the adversary to learn some information about the culprit; but it requires that this information leak is only up to a certain bound.

We take the definition in [9]—which generalizes the ones in [34,19]—and present a simulation-based proof method for this notion. Although the basic scenario is not exactly the same as before, the idea about using simulations is quite similar.

1.1 Outline of the paper

In Section 2 we illustrate notions of probabilistic anonymity using a couple of examples. They include the Dining Cryptographers protocol and the Crowds protocol. The formal definitions of anonymity notions are presented in Section 3, where we also introduce our models of anonymizing protocols called *anonymity automata*. In Section 4 we describe our simulation-based proof method for strong anonymity and prove its correctness. In Section 5 the proof method for probable innocence is described and applied to Crowds. Relation to other work with a similar interest is discussed in Section 6.

One remark on coalgebras and category theory: the paper is written in such a way that the reader should be able to treat the coalgebraic theory of traces and

[5] Unfortunately the combination of both non-determinism and probability—which is e.g. in probabilistic automata [39]—is not covered in this paper. We will come back to this point later in Section 6.3.

simulations (and the category theory behind it) as a back-end. The correctness of our proof method, however, does rely on the coalgebraic theory; hence some account on it is desirable in an effort to be self-contained and to justify the results. Unfortunately we find the categorical formalism employed therein simply not presentable in the current limited space: it makes little sense unless it is accompanied by ample illustration. The reader is referred for the relevant coalgebraic theory to [20], which we believe serves the end.

1.2 Notations

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

The set of lists in an alphabet X with length ≥ 1 is denoted by X^+, that is, $X^+ = X^* \cdot X$ in a regular-expression-like notation. This appears later as a domain of trace semantics for anonymity automata.

2 Motivating examples

In this section we motivate

- the probabilistic aspect of anonymity, and
- possible candidates for a formal notion of "probabilistic anonymity,"

by presenting concrete examples of anonymizing protocols. The first example is the well-known Dining Cryptographers (DC) protocol [10]: this will illustrate the importance of probability in anonymizing protocols. The second example of simple and artificial protocols for anonymous donation will clarify the idea behind the notion of strong probabilistic anonymity. Finally we present an example of Crowds [34]—which does not satisfy strong probabilistic anonymity—to motivate the weaker notion of probable innocence.

2.1 The Dining Cryptographers (DC) protocol

Here we follow [5] to illustrate the role of probability in anonymity, using the Dining Cryptographers (DC) protocol.

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

The goal of the DC protocol is as follows. The three cryptographers

- should reveal whether there is a payer among them, but
- should not revealing who of them is the payer (if any),

to the observer (called the *adversary* in the sequel) and also to the cryptographers who are not paying. The second point 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.

$$\begin{tikzcd} & \mathsf{Crypt}_0 \arrow[dl] & \\ \mathsf{Crypt}_1 \arrow[rr, bend right] & & \mathsf{Crypt}_2 \arrow[ul] \end{tikzcd}$$

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—must lie on the announcement. For example, given that Crypt_0 is paying, the configuration of coins

$$(\mathsf{h}, \mathsf{t}, \mathsf{h}) \qquad \text{that is} \qquad \begin{tikzcd} & \mathsf{h} \arrow[dl] & \\ \mathsf{t} \arrow[rr, bend right] & & \mathsf{h} \arrow[ul] \end{tikzcd},$$

results in the announcement

$$(\mathsf{a}, \mathsf{d}, \mathsf{a}) \qquad \text{that is} \qquad \begin{tikzcd} & \mathsf{a} \arrow[dl] & \\ \mathsf{d} \arrow[rr, bend right] & & \mathsf{a} \arrow[ul] \end{tikzcd}.$$

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.

Crypt_0 pays, and coins are $(\mathsf{h}, \mathsf{t}, \mathsf{h})$ or $(\mathsf{t}, \mathsf{h}, \mathsf{t})$
Crypt_1 pays, and coins are $(\mathsf{h}, \mathsf{h}, \mathsf{h})$ or $(\mathsf{t}, \mathsf{t}, \mathsf{t})$
Crypt_2 pays, and coins are $(\mathsf{h}, \mathsf{h}, \mathsf{t})$ or $(\mathsf{t}, \mathsf{t}, \mathsf{h})$

Up to now our arguments have been non-deterministic; now we shall explain how probabilistic aspects in DC can 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 conditional 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 come up with the following two things.

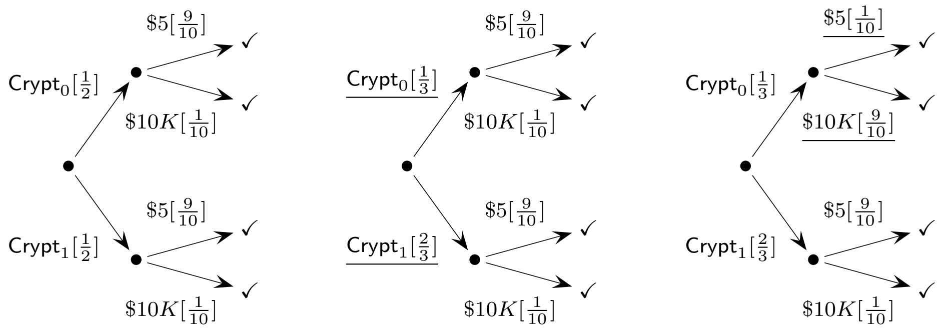
The first is an appropriate notion of "probabilistic anonymity" which holds with fair coins but is violated with biased coins. This is done in [5] and will be explained using the next simpler example.

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

2.3 Strong probabilistic anonymity

In this section we present toy examples of probabilistic anonymizing protocols to motivate the notion of strong (probabilistic) anonymity [5].

Let us think of a situation of *anonymous donation*: one of two cryptographers Crypt_0 and Crypt_1 tries to donate some money, without revealing the donor's identity. Consider the following three protocols, $\mathcal{X}_0, \mathcal{X}_1, \mathcal{X}_2$ from left to right.


$\mathsf{Crypt}_0[\frac{1}{2}]$
$\mathsf{Crypt}_1[\frac{1}{2}]$
$\$5[\frac{9}{10}]$
$\$10K[\frac{1}{10}]$
$\$5[\frac{9}{10}]$
$\$10K[\frac{1}{10}]$
$\underline{\mathsf{Crypt}_0[\frac{1}{3}]}$
$\underline{\mathsf{Crypt}_1[\frac{2}{3}]}$
$\$5[\frac{9}{10}]$
$\$10K[\frac{1}{10}]$
$\$5[\frac{9}{10}]$
$\$10K[\frac{1}{10}]$
$\mathsf{Crypt}_0[\frac{1}{3}]$
$\mathsf{Crypt}_1[\frac{2}{3}]$
$\underline{\$5[\frac{1}{10}]}$
$\underline{\$10K[\frac{9}{10}]}$
$\$5[\frac{9}{10}]$
$\$10K[\frac{1}{10}]$
✓

It is assumed that the labels Crypt_i—deciding which cryptographer is the donor—are invisible for the observer. Only the labels $\$n$—for money transactions—are visible.

The first protocol $\mathcal{X}_0$ works as follows. First the donor is chosen with the uniform probability distribution; no matter which is chosen, the donor donates \$5 with the probability 9/10 and $\$10K$ with 1/10. In the second one $\mathcal{X}_1$, it is in the specification of the protocol (hence is known to the adversary) that Crypt_1 is more likely to be the donor—the culprit. In $\mathcal{X}_2$, Crypt_0 is known to be rich so to say: when she is chosen more often than not she gives away $\$10K$.

Are these protocols “anonymous”? That is, by observing money transactions, can the adversary tell who is the culprit? Intuitively it is clear that $\mathcal{X}_0$ should be “anonymous” and $\mathcal{X}_2$ should not be. $\mathcal{X}_1$ can be debatable but we want to claim it to be anonymous. It is true that Crypt_1 is inherently more suspicious than Crypt_0 in $\mathcal{X}_1$ (the *a-priori* distribution of suspicion is not uniform). However, after observing any execution of the protocol, from the adversary’s viewpoint the cryptographers look exactly as suspicious as before. In other words, *an observation of transaction of* \$5 *or* $\$10K$ *does not carry any information on who is the culprit.*

This intuition leads to the notion of strong anonymity introduced in [5]. Here we give an informal description; a formal definition is found in Definition 3.8.

Definition 2.1 (Strong anonymity) *An anonymizing protocol $\mathcal{X}$ satisfies* strong (probabilistic) anonymity *if, for any observation o and users i and j, we have*

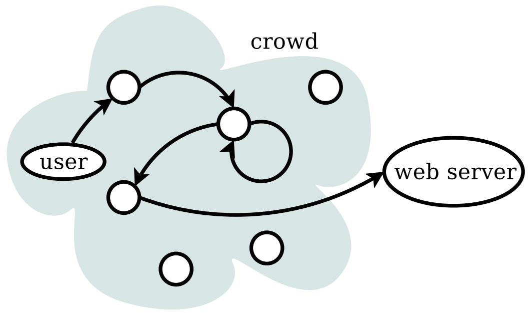
$$P_{\mathcal{X}}(o \mid i \text{ is a culprit}) = P_{\mathcal{X}}(o \mid j \text{ is a culprit}) \enspace .$$

Here $P_{\mathcal{X}}(o \mid i \text{ is a culprit})$ denotes the conditional probability for the event “given that the user i is the culprit, an execution of $\mathcal{X}$ yields the observation o.”

The intuition behind this definition is quite similar to the one behind the notion of *conditional anonymity* [19] whose formal definition is given later in Definition 3.9. In fact, it is shown in [5] that under reasonable assumptions these two notions of anonymity coincide.

Although the DC protocol with fair coins, as well as the anonymous donation protocols $\mathcal{X}_0$ and X_1 in Section 2.3, satisfies strong anonymity, there are a large body of real-world protocols which do not satisfy the property. Crowds [34] is one of them.

The Crowds protocol aims at anonymous web browsing, so that a user can send a request to a web server without revealing the user's identity. In essence the identity is concealed by relaying the request among a set ("crowd") of relays.



- A user, who has already joined a group ("crowd") of hosts, picks one relay out of the crowd with a uniform probability and forwards a request to the chosen relay. The relay can be the originating user herself.
- A relay in a crowd, on receiving a request, flips a (biased) coin to decide whether
  - · to deliver the request to its intended recipient, i.e. the web server (this happens with the probability $1 - p_f$), or
  - · to forward it to another relay (with the probability p_f). The next relay is chosen from the crowd with the uniform probability, including the current relay itself.

Here the probability p_f is a system-wide parameter. This way the web server only sees a request coming from a relay, hence it is not sure who originated the request.

It is obvious that Crowds gives no anonymity guarantee against a *global eavesdropper*—an adversary capable of observing all the network traffic. A more realistic attack scenario is that *some of the relays are corrupt and cooperates with the adversary.* A short calculation shows the following. The fact that a certain host i has sent a request to a corrupt relay—although the adversary cannot tell whether the host i is the originator of the request or is just forwarding it—makes the host i probabilistically more suspicious for being the originator than the other hosts. This makes the strong anonymity fail: the observation “i has sent a request to me” does carry some information on who is the originator.[6]

Still, when the number of relays (i.e. the cardinality of the crowd) is sufficiently bigger than that of corrupt relays, we want to say that Crowds is “anonymous,” that is, Crowds satisfies some weaker notion of anonymity that allows information leak up to a certain amount. The notion of *probable innocence* is one candidate of such an anonymity notion.

Intuitively, probable innocence is satisfied if *after any observation, the adversary can never have confidence exceeding* $1/2$ *that someone is the culprit.* In [34] it is shown that Crowds satisfies probable innocence if $p_f > 1/2$ and the number N of relays and the number c of corrupt relays satisfy[7]

$$N \geq \frac{p_f}{p_f - \frac{1}{2}}(c+1) \ .$$

However, the above statement in [34] is only true if every user in the crowd is equally suspicious before an execution of Crowds. Assume there is a user who is known to be much more frequently an originator of a request than other users; then the above “confidence is $\leq 1/2$” definition of probable innocence is more easily violated. The situation is similar to the comparison between $\mathcal{X}_0$ and $\mathcal{X}_1$ in Section 2.3; we want our definition robust to such a change of the a-priori distribution of suspicion. A user who is inherently more suspicious than others can still look more suspicious than others after an execution of the protocol; a bit more than it did before an execution because now we allow some marginal information leak; but not much more.

This idea led Chatzikokolakis and Palamidessi [9] to the following definition of probable innocence, which we formally present later in Definition 3.11. It is also shown to coincide with the definitions in [34,19] under suitable assumptions.

Definition 2.2 (Probable innocence) *An anonymizing protocol $\mathcal{X}$ satisfies* probable innocence *if, for any observation o and user i we have*

$$(n-1)\frac{P_{\mathcal{X}}(i \text{ is a culprit})}{\sum_{j\neq i} P_{\mathcal{X}}(j \text{ is a culprit})} \geq \frac{P_{\mathcal{X}}(i \text{ is a culprit} \mid o)}{\sum_{j\neq i} P_{\mathcal{X}}(j \text{ is a culprit} \mid o)} \ . \qquad (1)$$

Here n is the number of users who can possibly be a culprit; we are assuming that two different users cannot be culprits at the same time.

[6] Strong anonymity is satisfied if no relay in the crowd is corrupt. In this case, the adversary, when it receives a request from a host i, can tell for sure that i is forwarding the request. Hence the host i is no more suspicious than the other hosts.
[7] In fact, the definition of probable innocence used in [34] is slightly different from this one which is based on the definition in [19]. These two are shown in [9] to coincide under some mild conditions.

The left-hand side of (1) is about the a-priori distribution of suspicion. To get an idea let us assume that the a-priori distribution is uniform. Then (1) becomes

$$1 \geq \frac{P_{\mathcal{X}}(i \text{ is a culprit} \mid o)}{\sum_{j \neq i} P_{\mathcal{X}}(j \text{ is a culprit} \mid o)} \quad .$$

If we moreover assume that the observation o reveals existence of a culprit, that is,

$$\sum_i P_{\mathcal{X}}(i \text{ is a culprit} \mid o) = 1 \quad ,$$

the condition is equivalent to $1/2 \geq P_{\mathcal{X}}(i \text{ is a culprit} \mid o)$. In this way we can recover the original definition: "the adversary's confidence does not exceed 1/2."

3 Formalizing notions for 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 [39]. 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 [20] 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* $\mathsf{act}(i)$ which means that a user i is chosen as the culprit.
- 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*.*[8]
- $\mathcal{O}$ *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*

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

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

$$\mathcal{D}Y = \Big\{ d : Y \to [0,1] \mid \sum_{y \in Y} d(y) \leq 1 \Big\} \ . \tag{2}$$

  This operation $\mathcal{D}$ canonically extends to a monad[9] *which we shall call the* subdistribution monad*.*

  For example, the value $c(x)(a,\checkmark)$[10] *in $[0,1]$ is the probability with which a state x executes a and then successfully terminate (i.e. $x \xrightarrow{a} \checkmark$).*
- $s \in \mathcal{D}X$ *is a probability subdistribution over the state space X. This specifies which state would be a* starting *(or* initial*) one.*

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

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

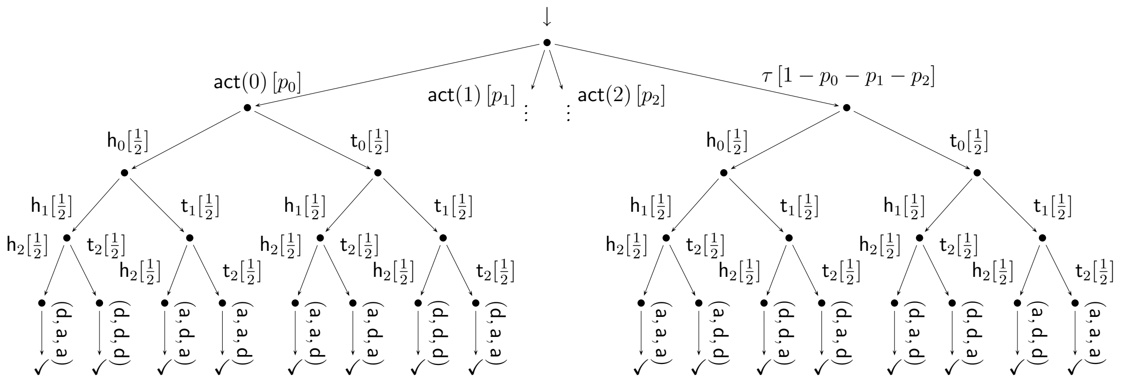
We need to fix the a-priori probability distribution on who will make a payment. Let us denote by p_i the probability with which the user i pays.

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*.*

[8] A user is called an *anonymous user* in [5].

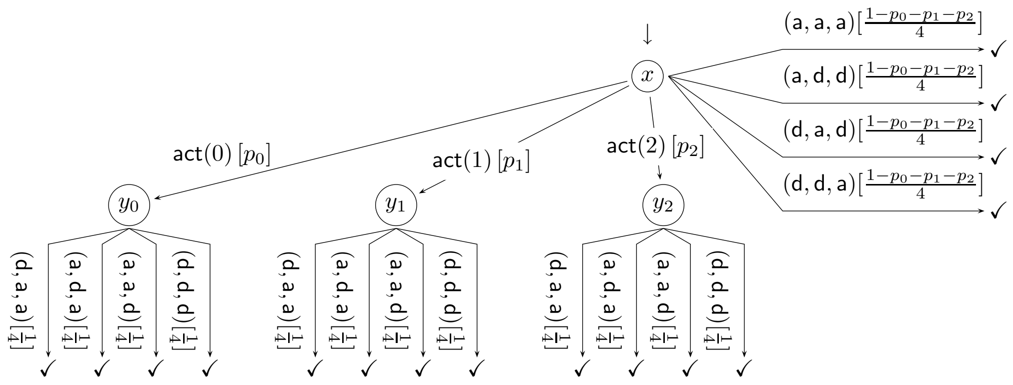
[9] *Monads* are a categorical notion. Interested readers are referred to [4] for the details.

[10] 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.



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 [20,23] applies to anonymity automata.[11] The generic theory is developed with two parameters T and F:

[11] As explained in Section 1.1, the reader can safely take the relevant (categorical) theory of coalgebras as a back-end and ignore its details for the rest of the paper.

- a monad T on **Sets** specifies the *branching-type*, such as non-determinism or probability;
- a functor F on **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 this generic framework. 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 $\mathcal{A}^+$ carries the following initial algebra in **Sets**. We denote its structure map by α.

$$\begin{tikzcd}
\mathcal{A} \times \{\checkmark\} + \mathcal{A} \times \mathcal{A}^+ \arrow[d, "\alpha"', "\cong"] & \kappa_1(a, \checkmark) \arrow[d, mapsto] & \kappa_2(a, \vec{a}) \arrow[d, mapsto] \\
\mathcal{A}^+ & \langle a \rangle & a \cdot \vec{a}
\end{tikzcd} ,$$

where $\langle a \rangle$ denotes a list of length 1, and $a \cdot \vec{a}$ is what would be written as (`cons` a $\vec{a}$) in the LISP-style. Therefore [20, Corollary 5.2] suggests that the set $\mathcal{A}^+$ is the appropriate "domain" of (finite) trace semantics for anonymity automata: this is actually the case later in Definition 3.3. [12]

3.3 Trace semantics for anonymity automata

Trace semantics for anonymity automata is used in defining probabilistic notions of anonymity. In a non-deterministic setting, trace semantics yields a *set* of lists ("traces") of actions which can possibly occur during an execution. In contrast, trace semantics of a probabilistic system is given by 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 semantics

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

[12] To be precise, the set $\mathcal{A}^+$ is that of all possible *traces*. This is the same for both non-deterministic and probabilistic settings. What is different in these two is whether *trace semantics* is given by a *set* of traces or a *probability distribution* over traces.

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

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

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 an execution of $\mathcal{X}$ *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 $P_{\mathcal{X}}(\vec{a}) \in [0,1]$ for a list $\vec{a} \in \mathcal{A}^+$ is the probability with which the system $\mathcal{X}$ executes actions in $\vec{a}$ 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 [20,23].

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 semantics $P_{\mathcal{X}}$ *of* $\mathcal{X}$ *coincides with the coalgebraic trace semantics* $\mathsf{tr}_{(s,c)}$ *defined in the generic theory [20, Definition 5.7] for* (s, c)*.* $\square$

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

$$\begin{array}{lcl@{\qquad}lcl} \langle \mathsf{act}(i), (\mathsf{d},\mathsf{a},\mathsf{a}) \rangle & \mapsto & p_i/4 & \langle (\mathsf{a},\mathsf{a},\mathsf{a}) \rangle & \mapsto & (1-p_0-p_1-p_2)/4 \\ \langle \mathsf{act}(i), (\mathsf{a},\mathsf{d},\mathsf{a}) \rangle & \mapsto & p_i/4 & \langle (\mathsf{a},\mathsf{d},\mathsf{d}) \rangle & \mapsto & (1-p_0-p_1-p_2)/4 \\ \langle \mathsf{act}(i), (\mathsf{a},\mathsf{a},\mathsf{d}) \rangle & \mapsto & p_i/4 & \langle (\mathsf{d},\mathsf{a},\mathsf{d}) \rangle & \mapsto & (1-p_0-p_1-p_2)/4 \\ \langle \mathsf{act}(i), (\mathsf{d},\mathsf{d},\mathsf{d}) \rangle & \mapsto & p_i/4 & \langle (\mathsf{d},\mathsf{d},\mathsf{a}) \rangle & \mapsto & (1-p_0-p_1-p_2)/4 \\ & & \textit{(for } i = 0,1,2\textit{)} & & & \end{array}$$

The other lists in $\mathcal{A}^+$ *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 [5, Assumption 1] and is true in all the examples in this paper. [13]

[13] Relaxing the assumption is desired in order to analyze a situation where an anonymizing protocol has multiple runs simultaneously. At now it is unclear even how to adapt the anonymity notions.

Assumption 3.6 (At most one actor action) *Let $\mathcal{X} = (X, \mathcal{U}, \mathcal{O}, c, s)$ be an anonymity automaton and $\vec{a} \in \mathcal{A}^+$. If $\vec{a}$ contains more than one actor actions, then we have $P_{\mathcal{X}}(\vec{a}) = 0$.*

3.4 Notions of probabilistic anonymity

In this section we formalize the two different notions of probabilistic anonymity, namely strong (probabilistic) anonymity [5] and probable innocence [34,19,9].

First, for the sake of simplicity of presentation, we shall introduce the following notations for predicates (i.e. subsets) on $\mathcal{A}^+$.

Definition 3.7 (Predicates $[\mathsf{act}(i)]$ and $[\vec{o}]$)

- *For each $i \in \mathcal{U}$, a predicate $[\mathsf{act}(i)]$ on $\mathcal{A}^+$ is defined as follows. In a regular-expression-like notation,*

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

that is, it is the set of lists which contains only one actor action which is $\mathsf{act}(i)$. Obviously we have $[\mathsf{act}(i)] \cap [\mathsf{act}(j)] = \emptyset$ if $i \neq j$.
- *For each $\vec{o} \in \mathcal{O}^*$, a predicate $[\vec{o}]$ on $\mathcal{A}^+$ is defined as follows.*

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

where the function $\mathsf{removeActor} : \mathcal{A}^+ \to \mathcal{O}^*$—which is defined by a suitable induction—removes actor actions appearing in a list. Hence the set $[\vec{o}] \subseteq \mathcal{A}^+$ consists of those lists which yield $\vec{o}$ as the adversary's observation. It is emphasized that $[\vec{o}]$ is not *the set of lists which contain $\vec{o}$ 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 $P_{\mathcal{X}}(\,[\mathsf{act}(i)]\,)$ are defined in a straightforward manner:

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

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

3.4.1 Strong (probabilistic) anonymity

Based on anonymity automata as models of anonymity protocols, we shall formalize the notion of strong anonymity which is informally introduced in

Definition 2.1.

Definition 3.8 (Strong anonymity [5]) *We say an anonymity automaton* $\mathcal{X}$ *satisfies* strong anonymity *if, for each* $i, j \in \mathcal{U}$ *and* $\vec{o} \in \mathcal{O}^*$,

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

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

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

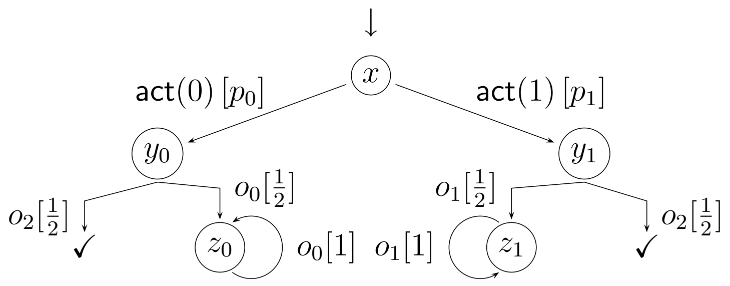
It is shown in [5] that under reasonable assumptions this notion coincides with *conditional anonymity* [19], whose formal definition we present now for completeness.

Definition 3.9 (Conditional anonimity [19]) *An anonymity automaton* $\mathcal{X}$ *satisfies* conditional anonymity *if for each* $i \in \mathcal{U}$ *and* $\vec{o} \in \mathcal{O}^*$,

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

The notion in Definition 3.8 is a natural probabilistic extension of *trace anonymity* in [37]. 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 [18]. 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 [28].

It is noted that in the current work, the adversary's observation ($\vec{o}$ in Definition 3.8 and elsewhere) is a *finite* sequence of observable actions which *leads to termination* ✓. This choice is made for a technical reason: the coalgebraic trace semantics in [20,23] assigns probabilities only to finite sequences; hence so does our current definition of trace semantics (Definition 3.3). This rules out infinite traces (infinite streams of observable actions) from our consideration and is problematic e.g. in the following protocol.



In this protocol an occurrence of o_0 (or o_1) immediately reveals that the culprit is the user 0 (or the user 1, respectively). However due to Definition 3.8 the protocol is still anonymous because it does not take infinite traces (like o_0^ω) or incomplete traces (like o_0^n) into account.

This leads to the following assumption.

Assumption 3.10 (No infinite traces) *There are no infinite traces in an anonymity automaton $\mathcal{X}$. More precisely, an execution of the automaton $\mathcal{X}$ leads to termination with probability* 1*:* $\sum_{\vec{a}\in\mathcal{A}^+} P_{\mathcal{X}}(\vec{a}) = 1$.

The assumption is trivially satisfied when there is no infinite path in $\mathcal{X}$. Unfortunately this stronger condition fails for our Crowds example (Section 5.3).

3.4.2 Probable innocence

The weaker notion of *probable innocence*—informally introduced in Definition 2.2—is also formalized based on anonymity automata and their trace semantics. It is based on the definition in [9].

Definition 3.11 (Probable innocence [9]) *We say an anonymity automaton $\mathcal{X}$ satisfies* probable innocence *if, for each $i \in \mathcal{U}$ and $\vec{o} \in \mathcal{O}^*$,*

- *Neither of the probabilities $P_{\mathcal{X}}(\bigcup_{j\neq i}[\mathsf{act}(j)])$ and $P_{\mathcal{X}}(\,\bigcup_{j\neq i}[\mathsf{act}(j)] \mid [\vec{o}]\,)$ is 0. These appear as denominators in the following inequality.*
- *We have*

$$(n-1)\frac{P_{\mathcal{X}}(\,[\mathsf{act}(i)]\,)}{P_{\mathcal{X}}(\,\bigcup_{j\neq i}[\mathsf{act}(j)]\,)} \geq \frac{P_{\mathcal{X}}(\,[\mathsf{act}(i)] \mid [\vec{o}]\,)}{P_{\mathcal{X}}(\,\bigcup_{j\neq i}[\mathsf{act}(j)] \mid [\vec{o}]\,)} \quad . \tag{3}$$

Here $n = |\mathcal{U}|$ is the number of users.

4 Verifying strong anonymity via probabilistic simulations

In this section we extend the proof method [28,27] for (non-deterministic) trace anonymity to the probabilistic setting for strong (probabilistic) anonymity. 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.

We start with the definition of $\mathsf{an}(\mathcal{X})$, the *anonymized version* of an anonymity automaton $\mathcal{X}$. Recall that the notion of strong 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 is a culprit is given by $P_{\mathcal{X}}(\,[\mathsf{act}(i)]\,)$. Its normalized, conditional version

$$\begin{aligned} r_i \overset{\text{def.}}{=} P_{\mathcal{X}}(\,[\mathsf{act}(i)] \mid \bigcup_{j\in\mathcal{U}} [\mathsf{act}(j)]\,) &= \frac{P_{\mathcal{X}}(\,[\mathsf{act}(i)] \cap \bigcup_{j\in\mathcal{U}} [\mathsf{act}(j)]\,)}{P_{\mathcal{X}}(\,\bigcup_{j\in\mathcal{U}} [\mathsf{act}(j)]\,)} \\ &= \frac{P_{\mathcal{X}}(\,[\mathsf{act}(i)]\,)}{\sum_{j\in\mathcal{U}} P_{\mathcal{X}}(\,[\mathsf{act}(j)]\,)} \end{aligned} \tag{4}$$

(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 a culprit, given that there is any culprit; 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 can be a culprit) *For an anonymity automaton $\mathcal{X}$,*

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

Intuitively, $\mathsf{an}(\mathcal{X})$ is obtained from $\mathcal{X}$ by distributing the probability for an actor action $\mathsf{act}(i)$ to each user j 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 $x \in X$,*

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

Here r_i is the a-priori suspicion defined in (4). On the first equation, the summand $r_i \cdot c(x)(\mathsf{act}(j), u)$ results from distributing the probability $c(x)(\mathsf{act}(j), u)$ for a transition $x \xrightarrow{\mathsf{act}(j)} u$, to a user i. This is illustrated in the following figure: here $\mathcal{U} = \{0, 1, \ldots, n-1\}$ and $q = c(x)(\mathsf{act}(j), u)$.

$$\begin{tikzcd} \boxed{\textit{In } \mathcal{X}} & \bullet \arrow[d, "{\mathsf{act}(j)\,[q]}"'] & & \boxed{\textit{In } \mathsf{an}(\mathcal{X})} & \bullet \arrow[d, bend right=80, "{\mathsf{act}(0)\,[r_0 \cdot q]}"'] \arrow[d, phantom, "\cdots"] \arrow[d, bend right=30] \arrow[d, bend left=30] \arrow[d, bend left=80, "{\mathsf{act}(n-1)\,[r_{n-1} \cdot q]}"] \\ & \bullet & & & \bullet \end{tikzcd} \tag{5}$$

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*

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

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

PROOF. Obvious from the definition of c^{an}. $\square$

Lemma 4.4 ($\mathsf{an}(\mathcal{X})$ satisfies strong anonymity) *Given an anonymity automaton $\mathcal{X}$, its anonymized version $\mathsf{an}(\mathcal{X})$ satisfies strong anonymity in the sense of Definition 3.8.*

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

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

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

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

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

$$r_j \cdot P_{\mathsf{an}(\mathcal{X})}(\,[\vec{o}] \cap [\mathsf{act}(i)]\,) = r_i \cdot P_{\mathsf{an}(\mathcal{X})}(\,[\vec{o}] \cap [\mathsf{act}(j)]\,) \enspace . \tag{6}$$

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

$$\begin{aligned}
P_{\mathsf{an}(\mathcal{X})}(\,[\vec{o}] \mid [\mathsf{act}(i)]\,) &= \frac{P_{\mathsf{an}(\mathcal{X})}(\,[\vec{o}] \cap [\mathsf{act}(i)]\,)}{P_{\mathsf{an}(\mathcal{X})}(\,[\mathsf{act}(i)]\,)} \\
&= \frac{r_i}{r_j} \cdot \frac{P_{\mathsf{an}(\mathcal{X})}(\,[\vec{o}] \cap [\mathsf{act}(j)]\,)}{P_{\mathsf{an}(\mathcal{X})}(\,[\mathsf{act}(i)]\,)} && \text{by (6)} \\
&= \frac{P_{\mathsf{an}(\mathcal{X})}(\,[\vec{o}] \cap [\mathsf{act}(j)]\,)}{P_{\mathsf{an}(\mathcal{X})}(\,[\mathsf{act}(j)]\,)} && \text{by definition of } r_i, r_j \\
&= P_{\mathsf{an}(\mathcal{X})}(\,[\vec{o}] \mid [\mathsf{act}(j)]\,)\ . && \square
\end{aligned}$$

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 [20]. 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 $[0,1]$.

$$\begin{aligned}
s(x) &\le \textstyle\sum_{y\in Y} t(y) \cdot f(y)(x) && \textit{for any } x \in X, \\
\textstyle\sum_{x\in X} f(y)(x) \cdot c(x)(e, \checkmark) &\le d(y)(e, \checkmark) && \textit{for any } y \in Y \textit{ and } e \in \mathcal{A}, \\
\textstyle\sum_{x\in X} f(y)(x) \cdot c(x)(e, x') &\le \textstyle\sum_{y'\in Y} d(y)(e, y') \cdot f(y')(x') \\
& && \textit{for any } y \in Y,\ e \in \mathcal{A} \textit{ and } x' \in X.
\end{aligned}$$

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 $[0, 1]$.

$$\textstyle\sum_{x \in X} s(x) \cdot b(x)(y) \leq t(y) \qquad \text{for any } y \in Y,$$

$$c(x)(e, \checkmark) \leq \textstyle\sum_{y \in Y} b(x)(y) \cdot d(y)(e, \checkmark) \text{ for any } x \in X \text{ and } e \in \mathcal{A},$$

$$\textstyle\sum_{x' \in X} c(x)(e, x') \cdot b(x')(y') \leq \sum_{y \in Y} b(x)(y) \cdot d(y)(e, y')$$

$$\text{for any } x \in X,\ e \in \mathcal{A} \text{ and } y' \in Y.$$

The definition looks puzzling. Why is a forward simulation a function of 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 [20, 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* $\vec{a} \in \mathcal{A}^+$,

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

PROOF. We know (Lemma 3.4) that the notions of trace semantics and simulation for anonymity automata are instantiations of the general, coalgebraic notions in [20,23]. Therefore we can appeal to the general soundness theorem [20, Theorem 6.1]. □

4.3 Probabilistic anonymity via simulations

We shall use the materials in Section 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 the automaton $\mathsf{an}(\mathcal{X})$ is constructed. The proof is a bit more complicated here than in the non-deterministic setting [28,27].

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:*

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

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

$$P_{\mathcal{X}} \sqsupseteq P_{\mathsf{an}(\mathcal{X})} \enspace , \tag{7}$$

where $\sqsupseteq$ refers to the pointwise order between functions $\mathcal{A}^+ \rightrightarrows [0,1]$. 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 $\mathsf{act}(i)$ with single $\mathsf{act}(\mathsf{sb})$—$\mathsf{sb}$ is for "somebody." This conceals actor actions in $\mathcal{Y}$; hence $\mathsf{obs}(\mathcal{Y})$ only carries information on the observable actions of $\mathcal{Y}$.

$$\begin{tikzcd}[column sep=large]
\boxed{\text{In } \mathcal{X}} & \bullet \arrow[d, bend right=70, "{\mathsf{act}(0)\,[q_0]}"'] \arrow[d, bend right=25] \arrow[d, phantom, "\cdots"] \arrow[d, bend left=25] \arrow[d, bend left=70, "{\mathsf{act}(n-1)\,[q_{n-1}]}"] & \boxed{\text{In } \mathsf{obs}(\mathcal{X})} & \bullet \arrow[d, "{\mathsf{act}(\mathsf{sb})\,[q_0 + \cdots + q_{n-1}]}"'] \\
& \bullet & & \bullet
\end{tikzcd} \tag{8}$$

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

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

The following fact is obvious.

Sublemma 4.9 *Given an anonymity automaton $\mathcal{X}$, the two automata $\mathsf{obs}(\mathcal{X})$ and $\mathsf{obs}(\mathsf{an}(\mathcal{X}))$ are identical.* $\square$

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 $\mathcal{Y}$,*

$$\sum_{\vec{a}\in\mathcal{A}^+} P_{\mathcal{Y}}(\vec{a}) = \sum_{\vec{a'}\in(\mathcal{A}^{\mathsf{obs}})^+} P_{\mathsf{obs}(\mathcal{Y})}(\vec{a'}) \enspace .$$

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

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 (7) is strict. That is, there exists $\vec{a}_0 \in \mathcal{A}^+$ such that $P_{\mathcal{X}}(\vec{a}_0) \gneqq P_{\mathsf{an}(\mathcal{X})}(\vec{a}_0)$. Then, by (7) we have $\sum_{\vec{a}\in\mathcal{A}^+} P_{\mathcal{X}}(\vec{a}) \gneqq \sum_{\vec{a}\in\mathcal{A}^+} P_{\mathsf{an}(\mathcal{X})}(\vec{a})$. However,

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

This contradiction concludes the proof of Lemma 4.7. □

Now we are ready to state our main result for verifying strong anonymity.

Theorem 4.11 (Strong anonymity via simulations) *If there exists a forward or backward simulation from* $\mathsf{an}(\mathcal{X})$ *to* $\mathcal{X}$*, then* $\mathcal{X}$ *satisfies strong anonymity.*

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 strongly anonymous. This proves strong anonymity of $\mathcal{X}$: recall that strong 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.

Remark 4.13 *For our proof method to work, choosing a right anonymity automaton to represent a given protocol is important. Consider the following protocol (suggested by one of the referees): users in the set $\mathcal{U} = \{0, 1, \ldots, n-1\}$ pass a token through the group, starting from 0 all the way to $n-1$, each step being from the user i to the user $i+1$. Exactly one user is allowed to write into an empty token. The adversary observes the movement of the token but not its content nor the action of writing in.*

For simplicity let us assume $n = 2$. Although the protocol obviously satisfies strong anonymity, the representation below on the left—which represents the natural order of actions—does not allow the application of our simulation-based method.

$$\begin{tikzcd}
& {} \arrow[d] & \\
& x \arrow[dl, "\mathsf{act}(0)\,[p_0]"'] \arrow[dr, "\mathsf{pass}_{0\to1}\,[p_1]"] & \\
y_0 \arrow[d, "\mathsf{pass}_{0\to1}[1]"'] & & y_1 \arrow[d, "\mathsf{act}(1)[1]"] \\
\checkmark & & \checkmark
\end{tikzcd}
\qquad
\begin{tikzcd}
& {} \arrow[d] & \\
& x \arrow[dl, "\mathsf{act}(0)\,[p_0]"'] \arrow[dr, "\mathsf{act}(1)[p_1]"] & \\
y_0 \arrow[d, "\mathsf{pass}_{0\to1}[1]"'] & & y_1 \arrow[d, "\mathsf{pass}_{0\to1}\,[1]"] \\
\checkmark & & \checkmark
\end{tikzcd}$$

Such a problem can be always mended by "moving up actor actions" i.e. transformation into an automaton where an actor action is always the first action to be taken. Above on the right is such an automaton, to which our proof method is successfully applied.

5 Verifying probable innocence via probabilistic simulations

In this section we present a simulation-based proof method for the weaker notion of probable innocence. Although the basic idea is similar to that in the last section for strong anonymity, there are certain differences as well. Most notably, now we have to find a simulation from $\mathcal{X}$ to its "innocent version" $\mathsf{inno}(\mathcal{X})$; this is in the opposite direction of what we do for strong anonymity.

The basic scenario is as follows.

(1) We model an anonymizing protocol as an anonymity automaton $\mathcal{X}$.
(2) We construct the *innocent version* $\mathsf{inno}(\mathcal{X})$ of $\mathcal{X}$. Here it is not much of our concern whether $\mathsf{inno}(\mathcal{X})$ satisfies probable innocence or not. Rather $\mathsf{inno}(\mathcal{X})$ is thought of as the automaton describing the upper bound of admissible information leakage on who is the culprit. In fact, by examining

the construction of $\mathsf{inno}(\mathcal{X})$, we can prove that

$$P_{\mathcal{X}} \sqsubseteq P_{\mathsf{inno}(\mathcal{X})} \quad \Longrightarrow \quad \mathcal{X} \text{ satisfies probable innocence.} \tag{9}$$

(3) We find a simulation from $\mathcal{X}$ to $\mathsf{inno}(\mathcal{X})$. By soundness theorem this yields trace inclusion $P_{\mathcal{X}} \sqsubseteq P_{\mathsf{inno}(\mathcal{X})}$, which by (9) proves that $\mathcal{X}$ satisfies probable innocence.

Although this scenario is somewhat different from the one for strong anonymity (or for non-deterministic trace anonymity [27,28]), they are the same in that:

- we model a (trace-based) anonymity property as the "idealized version" of $\mathcal{X}$;
- by finding a simulation and then appealing to soundness theorem, we show that the property is preserved from the idealized version to the original $\mathcal{X}$.

Therein the construction of the "idealized version of $\mathcal{X}$" must be much tailored to the specific property to be verified.

5.1 *"Innocent" automaton* $\mathsf{inno}(\mathcal{X})$

We describe the construction of the innocent version $\mathsf{inno}(\mathcal{X})$ of a given anonymity automaton $\mathcal{X}$. To illustrate the intuition, let us assume that the a-priori suspicion is uniformly distributed, that is, $r_i = 1/n$ where $\mathcal{U} = \{0, 1, \dots, n-1\}$. The automaton $\mathsf{inno}(\mathcal{X})$ is obtained by replacing actor actions in $\mathcal{X}$ as follows: compare it with (5).

$$\begin{tikzcd} \boxed{\text{In } \mathcal{X}} & \bullet \arrow[d, "{\mathsf{act}(j)\,[q]}"'] & & \boxed{\text{In } \mathsf{inno}(\mathcal{X})} & \bullet \arrow[d, bend right=60, "{\mathsf{act}(0)\,[\frac{q}{2}]}"'] \arrow[d, phantom, "\cdots"] \arrow[d, bend left=60, "{\mathsf{act}(n-1)\,[\frac{q}{2}]}"] \\ & \bullet & & & \bullet \end{tikzcd} \tag{10}$$

There is an obvious problem in this definition: the resulting $\mathsf{inno}(\mathcal{X})$ may not be an anonymity automaton, since the probabilities $q/2$ appearing n times can add up to more than 1. We shall call such an automaton an *extended anonymity automaton.*

Definition 5.1 (Extended anonymity automata) *An* extended anonymity automaton $(X, \mathcal{U}, \mathcal{O}, c, s)$ *is the same thing as an anonymity automaton (Definition 3.1), except for the fact that* c *is a function of the type*

$$X \longrightarrow \mathcal{V}\big(\mathcal{A} \times \{\checkmark\} + \mathcal{A} \times X\big)$$

rather than $X \to \mathcal{D}(\mathcal{A} \times \{\checkmark\} + \mathcal{A} \times X)$. Here the operation $\mathcal{V}$ is such that[14]

$$\mathcal{V}Y = [0,\infty]^Y = \Big\{ d : Y \to [0,\infty] \Big\} \ . \tag{11}$$

The operation $\mathcal{V}$ canonically extends to a monad, which we shall call the valuation monad.

Given an extended anonymity automaton $\mathcal{X}$, its trace semantics

$$P_{\mathcal{X}} \in \mathcal{V}(A^+) \tag{12}$$

is defined exactly in the same way as for anonymity automata (Definition 3.3).

Given an anonymity automaton $\mathcal{X}$, it can be thought of also as an extended anonymity automaton because there is an inclusion map $\mathcal{D}Y \hookrightarrow \mathcal{V}Y$:

$$X \quad \overset{c}{\to} \quad \mathcal{D}(\mathcal{A} \times \{\checkmark\} + \mathcal{A} \times X) \quad \hookrightarrow \quad \mathcal{V}(\mathcal{A} \times \{\checkmark\} + \mathcal{A} \times X) \ . \tag{13}$$

Therefore we are overriding the notation $P_{\mathcal{X}}$ in (12). This is not a problem, since for an anonymity automaton $\mathcal{X}$ the following two trace semantics obviously coincide.

- Its trace semantics (Definition 3.3) as an anonymity automaton, and
- the trace semantics of the extended anonymity automaton induced by (13).

Before giving a formal definition of $\mathsf{inno}(\mathcal{X})$, note that the core inequality (3) in the definition of probable innocence is equivalent to the following one, which uses the a-priori distribution of suspicion $(r_i)_{i \in \mathcal{U}}$ from (4):

$$P_{\mathcal{X}}(\, [\mathsf{act}(i)] \cap [\vec{o}]\,) \quad \leq \quad \frac{(n-1)r_i}{1+(n-2)r_i} \cdot P_{\mathcal{X}}\Big(\big(\bigcup_{j \in \mathcal{U}} [\mathsf{act}(j)] \big) \cap [\vec{o}]\, \Big) \ . \tag{14}$$

[14] The range of $d \in \mathcal{V}Y$ is $[0,\infty]$ including ∞. This choice is made because, in order to apply the coalgebraic theory of traces and simulations, $\mathcal{V}Y$ needs to carry a **Cppo**-structure. In particular, any increasing ω-chain in $\mathcal{V}Y$ must have its upper bound in $\mathcal{V}Y$.

The equivalence is shown by the following easy calculation:

$$\begin{array}{rrcl}
& (n-1)\dfrac{P_{\mathcal{X}}(\,[\mathsf{act}(i)]\,)}{P_{\mathcal{X}}(\bigcup_{j\neq i}[\mathsf{act}(j)]\,)} & \geq & \dfrac{P_{\mathcal{X}}(\,[\mathsf{act}(i)] \mid [\vec{o}]\,)}{P_{\mathcal{X}}(\bigcup_{j\neq i}[\mathsf{act}(j)] \mid [\vec{o}]\,)} \\
\iff & \dfrac{(n-1)r_i}{\sum_{j\neq i} r_j} & \geq & \dfrac{P_{\mathcal{X}}(\,[\mathsf{act}(i)] \mid [\vec{o}]\,)}{P_{\mathcal{X}}(\bigcup_{j\neq i}[\mathsf{act}(j)] \mid [\vec{o}]\,)} \\
\iff & \dfrac{(n-1)r_i}{1-r_i} & \geq & \dfrac{P_{\mathcal{X}}(\,[\mathsf{act}(i)] \mid [\vec{o}]\,)}{P_{\mathcal{X}}(\bigcup_{j\neq i}[\mathsf{act}(j)] \mid [\vec{o}]\,)} \\
\iff & P_{\mathcal{X}}(\,[\mathsf{act}(i)] \mid [\vec{o}]\,) & \leq & \dfrac{(n-1)r_i}{1-r_i}\cdot P_{\mathcal{X}}(\bigcup_{j\neq i}[\mathsf{act}(j)] \mid [\vec{o}]\,) \\
\iff & \left(1+\dfrac{(n-1)r_i}{1-r_i}\right)P_{\mathcal{X}}(\,[\mathsf{act}(i)] \mid [\vec{o}]\,) & \leq & \dfrac{(n-1)r_i}{1-r_i}\cdot P_{\mathcal{X}}(\bigcup_{j\in\mathcal{U}}[\mathsf{act}(j)] \mid [\vec{o}]\,) \\
\iff & \dfrac{1+(n-2)r_i}{1-r_i}\cdot\dfrac{P_{\mathcal{X}}(\,[\mathsf{act}(i)]\cap[\vec{o}]\,)}{P_{\mathcal{X}}(\,[\vec{o}]\,)} & \leq & \dfrac{(n-1)r_i}{1-r_i}\cdot\dfrac{P_{\mathcal{X}}(\bigcup_{j\in\mathcal{U}}[\mathsf{act}(j)]\cap[\vec{o}]\,)}{P_{\mathcal{X}}(\,[\vec{o}]\,)}\quad ,
\end{array}$$

from which (14) follows immediately.

Definition 5.2 (Anonymity automaton $\mathsf{inno}(\mathcal{X})$) *Given an anonymity automaton $\mathcal{X} = (X, \mathcal{U}, \mathcal{O}, c, s)$, its* innocent version *$\mathsf{inno}(\mathcal{X})$ is an extended anonymity automaton given by the 5-tuple $(X, \mathcal{U}, \mathcal{O}, c^{\mathsf{inno}}, s)$, where c^{inno} is defined as follows. For each $x \in X$,*

$$\begin{aligned}
c^{\mathsf{inno}}(x)(\mathsf{act}(i), u) &= \frac{(n-1)r_i}{1+(n-2)r_i}\cdot\sum_{j\in\mathcal{U}} c(x)(\mathsf{act}(j), u) \\
&\qquad\qquad\qquad \textit{for } i \in \mathcal{U} \textit{ and } u \in \{\checkmark\} + X, \\
c^{\mathsf{inno}}(x)(o, u) &= c(x)(o, u) \qquad \textit{for } o \in \mathcal{O} \textit{ and } u \in \{\checkmark\} + X.
\end{aligned}$$

The coefficient $(n-1)r_i/(1+(n-2)r_i)$ comes from the inequality (14). The definition agrees with the informal description (10) when the a-priori suspicion $(r_i)_{i\in\mathcal{U}}$ is uniform.

5.2 Probable innocence via simulations

The intuition about the automaton $\mathsf{inno}(\mathcal{X})$ is that it represents the upper bound of admissible information leak. This intuition is made precise in the following lemma, which is a crucial step in our simulation-based proof method for probable innocence.

Lemma 5.3 *If we have*

$$P_{\mathcal{X}} \sqsubseteq P_{\mathsf{inno}(\mathcal{X})} \qquad \textit{that is,} \qquad \forall \vec{a} \in \mathcal{A}^+.\quad P_{\mathcal{X}}(\vec{a}) \leq P_{\mathsf{inno}(\mathcal{X})}(\vec{a})\ ,$$

then the anonymity automaton $\mathcal{X}$ satisfies probable innocence (Definition 3.11).

PROOF. By calculation much like the one which shows the equality (6) in the proof of Lemma 4.4, we obtain the following.

$$P_{\mathsf{inno}(\mathcal{X})}(\,[\vec{o}] \cap [\mathsf{act}(i)]\,) = \frac{(n-1)r_i}{1+(n-2)r_i} \cdot \sum_{j \in \mathcal{U}} P_{\mathcal{X}}(\,[\vec{o}] \cap [\mathsf{act}(j)]\,) \qquad (15)$$

By assumption, we have

$$P_{\mathcal{X}}(\,[\vec{o}] \cap [\mathsf{act}(i)]\,) \le P_{\mathsf{inno}(\mathcal{X})}(\,[\vec{o}] \cap [\mathsf{act}(i)]\,) \ ,$$

which, together with (15), derives (14). □

The assumption of Lemma 5.3—trace inclusion $P_{\mathcal{X}} \sqsubseteq P_{\mathsf{inno}(\mathcal{X})}$—shall be shown by finding a probabilistic simulation. Now that $\mathsf{inno}(\mathcal{X})$ is an *extended* anonymity automaton, we have to adapt the notion of probabilistic simulation and its soundness result to the extended setting. In terms of the generic theory in [20,23], this corresponds to changing the parameter T from $\mathcal{D}$ to $\mathcal{V}$.

Definition 5.4 (Forward/backward simulations, extended) *Let $\mathcal{X}$ and $\mathcal{Y}$ be extended anonymity automata. A* forward (or backward) simulation *from $\mathcal{X}$ to $\mathcal{Y}$ is the same thing as a forward (or backward) simulation for anonymity automata (Definition 4.5), except that*

- *we have now $\mathcal{V}$ in place of $\mathcal{D}$, and*
- *the inequalities are interpreted in $[0,\infty]$ rather than in $[0,1]$.*

For example, a forward simulation from $\mathcal{X}$ to $\mathcal{Y}$ is a function

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

which satisfies suitable inequalities as in Definition 4.5.

Theorem 5.5 (Soundness of simulations, extended) *If there is a forward or backward simulation from an extended anonymity automaton $\mathcal{X}$ to another $\mathcal{Y}$, then we have trace inclusion*

$$P_{\mathcal{X}} \sqsubseteq P_{\mathcal{Y}} \ , \qquad \textit{that is,} \qquad \forall \vec{a} \in \mathcal{A}^{+}. \quad P_{\mathcal{X}}(\vec{a}) \le P_{\mathcal{Y}}(\vec{a}) \ .$$

PROOF. The general, coalgebraic theory of traces [20,23] applies to the choice of parameters $T = \mathcal{V}$ and $F = \mathcal{A} \times \{\checkmark\} + \mathcal{A} \times \_$, in which case the coalgebraic notion of (T,F)-system instantiates to extended anonymity automata. Moreover, the coalgebraic notions of trace semantics and simulation coincide with the corresponding notions for extended anonymity automata

(Definitions 5.1 and 5.4). Therefore the statement is an instance of the general soundness theorem [20, Theorem 6.1]. □

Theorem 5.6 (Probable innocence via simulations) *If there exists a forward or backward simulation from $\mathcal{X}$ to $\mathsf{inno}(\mathcal{X})$, then $\mathcal{X}$ satisfies probable innocence.*

PROOF. By the soundness theorem (Theorem 5.5) we have $P_{\mathcal{X}} \sqsubseteq P_{\mathsf{inno}(\mathcal{X})}$; by Lemma 5.3 this implies $\mathcal{X}$'s probable innocence. □

5.3 Example: Crowds

We shall verify probable innocence of the Crowds protocol using our simulation-based method. We focus on a restricted setting where:

- there is only one corrupt relay (this is for simplicity); and
- the a-priori suspicion r_i is uniform. This is to observe a certain coincidence (Proposition 5.7) between our analysis and the original one made in [34]. The latter says: the inequality
$$N \geq \frac{p_f}{p_f - \frac{1}{2}}(c+1) \tag{16}$$
and $p_f > 1/2$ together is sufficient for Crowds to satisfy probable innocence. Here N is the number of all relays and c is that of the corrupt ones, hence $c = 1$ now.

5.3.1 First modeling

Let us denote the set of users by $\mathcal{U} = [0, n-1] = \{0, 1, \ldots, n-1\}$. It is now the set of incorrupt relays, hence the number of all relays is $N = n+1$. Our first, naive modeling of the Crowds protocol is as follows. On the left in Figure 1 is this automaton when $n = 2$.

- The state space is $\{x\} + \{y_0, \ldots, y_{n-1}\} + \{z_0, \ldots, z_{n-1}\} + \{z_c\}$. The intuition is:
  · x is the initial state;
  · y_i is the state where the user i is holding a message as its originator;
  · z_i is where the user i is holding the message as a relay; and
  · z_c is where the (only) corrupt user is in possession of the message.
- The transitions from each state are:
  · $x \xrightarrow{\mathsf{act}(i)[\frac{1}{n}]} y_i$ for each $i \in [0, n-1]$, since we assumed that the a-priori suspicion is uniform;

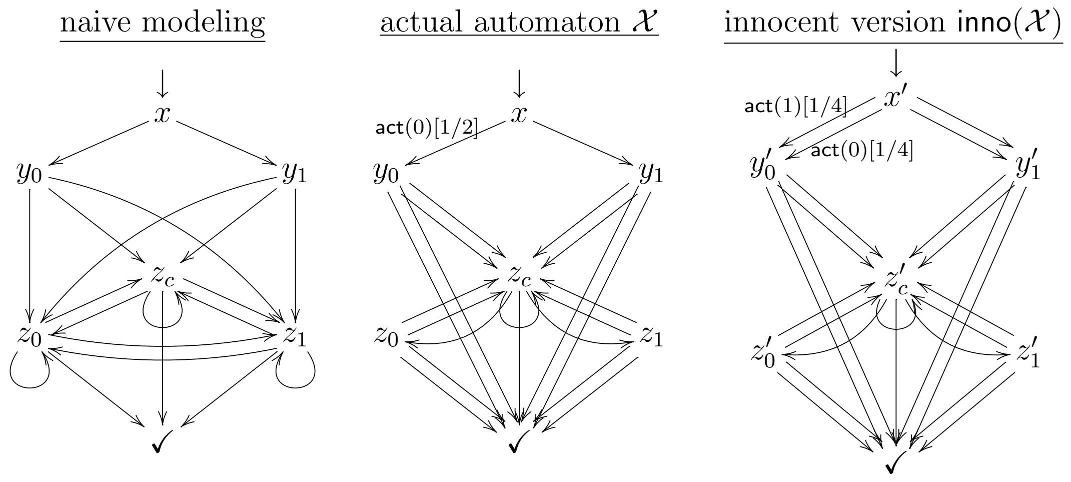


Figure 1. Automata for the Crowds protocol

· $y_i \overset{\mathsf{s}(i,j)[\frac{1}{n+1}]}{\longrightarrow} z_j$ for each $i, j \in [0, n]$. Here $\mathsf{s}(i, j)$ is an action that represents relaying a message from the user i to j. From the state y_i there is another transition $y_i \overset{\mathsf{s}(i,c)[\frac{1}{n+1}]}{\longrightarrow} z_c$ where the message is sent to the corrupt relay;

· from the state z_i a message is either

  delivered to the intended recipient: $z_i \overset{\mathsf{s}(i,\checkmark)[1-p_f]}{\longrightarrow} \checkmark$; or

  forwarded to another relay: $z_i \overset{\mathsf{s}(i,j)[\frac{p_f}{n+1}]}{\longrightarrow} z_j$ for each $j \in [0, n-1]$ and $z_i \overset{\mathsf{s}(i,c)[\frac{p_f}{n+1}]}{\longrightarrow} z_c$.

  The state z_c has the same kind of transitions;

- x is the initial state with probability 1.

5.3.2 Taking the closure to obtain $\mathcal{X}$

Although the last one is a straightforward modeling of the protocol, many actions in it must be kept invisible to the adversary. We replace these actions—namely the actions $\mathsf{s}(i, j)$ for each $i, j \in [0, n-1]$, those which do not involve c or ✓—by the silent action τ. After that we have to get rid of τ's by taking the closure: for example a transition $y_i \overset{\mathsf{s}(j,c)}{\longrightarrow} z_c$ in the closure is understood as a sequence of transitions

$$y_i \overset{\tau}{\longrightarrow} \cdots \overset{\tau}{\longrightarrow} \cdot \overset{\mathsf{s}(j,c)}{\longrightarrow} z_c$$

combined as one. A (rather laborious, but) straightforward calculation derives the following automaton as the outcome. Only its transitions are different from the original automaton:

- the transitions from the initial state x (where only actor actions take place) remain the same;

- from the state y_i, a message is either
  - · delivered to the intended recipient without going through the corrupt relay: $y_i \xrightarrow{\mathsf{s}(j,\checkmark)} \checkmark$ for each $j \in [0, n-1]$. Note that i and j need not be identical: a message originated by i can be passed around and finally delivered to the recipient by $j \neq i$. Each of these transitions occurs with the probability $\frac{1-p_f}{n+1-np_f}$; or
  - · passed to the corrupt relay: $y_i \xrightarrow{\mathsf{s}(j,c)} z_c$. The assigned probability is: $\frac{n+1-(n-1)p_f}{(n+1)(n+1-np_f)}$ if $i = j$; $\frac{p_f}{(n+1)(n+1-np_f)}$ if $i \neq j$;
- from the state z_i,
  - · $z_i \xrightarrow{\mathsf{s}(j,\checkmark)} \checkmark$ occurs with the probability $\frac{n+1-2np_f+(n-1)p_f^2}{n+1-np_f}$ if $i = j$; with $\frac{p_f-p_f^2}{n+1-np_f}$ if $i \neq j$;
  - · $z_i \xrightarrow{\mathsf{s}(j,c)} z_c$ occurs with the probability $\frac{(n+1)p_f-(n-1)p_f^2}{(n+1)(n+1-np_f)}$ if $i = j$; with $\frac{p_f^2}{(n+1)(n+1-np_f)}$ if $i \neq j$.

  The transitions from the state z_c remain the same as in the original, naive modeling.

The resulting "closure" automaton looks like the one in the center of Figure 1. This is our anonymity automaton $\mathcal{X}$ for the Crowd protocol.

5.3.3 Probable innocence by simulation

We shall denote the states of $\mathsf{inno}(\mathcal{X})$ by x', y'_i, z'_i and z'_c, making clear their correspondence to the states of $\mathcal{X}$—this is the same convention as in Example 4.12. The innocent version $\mathsf{inno}(\mathcal{X})$ is obtained from $\mathcal{X}$ by distributing the probability for each actor action to others (Definition 5.2). We assumed that $r_i = 1/n$ for each $i \in [0, n-1]$; hence the coefficient in Definition 5.2 is now

$$\frac{(n-1)r_i}{1+(n-2)r_i} = \frac{1}{2} \quad . \tag{17}$$

Therefore in the automaton $\mathsf{inno}(\mathcal{X})$, the transitions from the initial state x' are: $x' \xrightarrow{\mathsf{act}(j)[\frac{1}{2n}]} y'_i$ for each $i, j \in [0, n-1]$. The automaton on the right in Figure 1 is $\mathsf{inno}(\mathcal{X})$ when $n = 2$. [15]

Let us find a simulation from $\mathcal{X}$ to $\mathsf{inno}(\mathcal{X})$, which by Theorem 5.6 guarantees probable innocence. Although it seemed impossible for us to find a forward simulation, there is a natural candidate for a backward simulation. Namely,

[15] Although the automata when $n = 2$ are depicted for illustration, note that they do not satisfy probable innocence. In particular, no choice of p_f satisfies the key inequality $N \geq \frac{p_f}{p_f-1/2}(c+1)$ in [34] if $c = 1$ and $N = n+1 = 3$.

it is a function $b : \{x, y_0, \dots\} \to \mathcal{D}\{x', y'_0, \dots\}$ (see Definition 4.5) defined as follows.

$$\begin{aligned}
b(x) &= [x' \mapsto 1] \ , \\
b(y_i) &= [y'_j \mapsto 1/2]_{j \in [0,n-1]} && \text{for each } i \in [0, n-1], \\
b(z_i) &= [z'_j \mapsto 1/2]_{j \in [0,n-1]} && \text{for each } i \in [0, n-1], \\
b(z_c) &= [z'_c \mapsto 1] \ .
\end{aligned}$$

Here the value 1/2 seems to come from (17). Some calculation shows the following.

Proposition 5.7 *The function b defined above is a backward simulation from $\mathcal{X}$ to* $\mathsf{inno}(\mathcal{X})$, *if and only if the parameters n and p_f satisfy $p_f > 1/2$ and the inequality*

$$n \geq \frac{p_f + \frac{1}{2}}{p_f - \frac{1}{2}} \ , \quad \textit{that is} \quad N = n + 1 \geq \frac{p_f}{p_f - \frac{1}{2}}(1 + 1) \ .$$

By Theorem 5.6, $\mathcal{X}$ satisfies probable innocence if these inequalities are satisfied. □

The latter inequality is exactly the inequality (16) taken from [34]. Thus we have reproduced the analysis in [34] by means of simulations.

6 Related work

6.1 Probabilistic non-interference

In the theory of programming languages, the property of *non-interference* [30] has attracted much attention. Intuitively, a program C satisfies non-interference if there is no insecure information flow from variables with high confidentiality (*high variables*) to those with low confidentiality (*low variables*).

Let us put it slightly more formally. Assume that two memory states μ and ν differ only in low variables ($\mu =_L \nu$). Here a memory state is a list of pairs of a variable name and a value (such as `[H=0,L=0]`). Non-interference of a program C requires that the execution of C with the initial memory state μ is observationally indistinguishable from that with ν. In particular, if μ' and ν' are the memory states after these executions, respectively, then they have to agree on low variables:

$$\mu =_L \nu, \quad \mu \overset{C}{\rightsquigarrow} \mu', \quad \nu \overset{C}{\rightsquigarrow} \nu' \quad \Longrightarrow \quad \mu' =_L \nu'.$$

A possible variant of the formal definition concerns C's termination as well.

Volpano and Smith [45] noticed the importance of probabilistic aspects in non-interference. The leading example is in a simple programming language which has the *multi-threading* construct $C \mid C'$ which denotes running two threads C and C' in parallel, in an interleaving manner. Even though there is no explicit probabilistic construct in the language, a probabilistic *scheduler*—a mechanism that determines which thread to execute first—introduces probability in an operational model. In [45] a simple example is presented in which insecure information flow emerges only in a probabilistic setting (but not in a non-deterministic setting).

A definition of probabilistic non-interference (disregarding termination) can be presented as follows. Let μ_H, ν_H be memory states of high variables; μ_L, μ'_L be states of low variables. A program C satisfies probabilistic non-interference if, for every μ_H, ν_H, μ_L and μ'_L,

$$\begin{aligned} &\Pr(\mu_L \overset{C}{\rightsquigarrow} \mu'_L \mid \text{initial high-memory is } \mu_H) \\ &= \Pr(\mu_L \overset{C}{\rightsquigarrow} \mu'_L \mid \text{initial high-memory is } \nu_H)\ . \end{aligned}$$

Now notice the similarity to the notion of strong anonymity (Definition 2.1). We can think of the change of low-memory $\mu_L \overset{C}{\rightsquigarrow} \mu'_L$ as an "observation"; and the initial high-memory as a "culprit," i.e. the information that we want to disguise. This similarity between the two notions—(strong) probabilistic anonymity and probabilistic non-interference—suggests that a technique to ensure one of these properties can be used to ensure the other property.

Use of *type systems* is a standard technique to ensure that a certain piece of program satisfies a certain desirable property, such as non-interference. A type system typically has *typing rules*, and *soundness theorem* stating: if a program C is *typable*—meaning that we can derive a typing judgment for C using the given rules—then C satisfies the desired property. One big advantage of the type-based approach is that type systems are often accompanied with *typing algorithms* that effectively determine whether a program is typable or not.

For probabilistic non-interference, a couple of type systems have been proposed [45,36]. They are in principle extensions of the type system in [42] that ensures non-deterministic notion of non-interference. For example, the following program with an obvious (but indirect) insecure information flow

```
if H=0 then L=0 else L=1;
```
(18)

is typable in none of the type systems in [42,45,36]. Moreover, any program containing this piece is not typable either, because of the compositional way of type derivation in these type systems.

Because of this fact the type systems from [45,36] do not seem widely applicable to verification of anonymizing protocols. Consider the example of Dining Cryptographers; a natural representation of the protocol as a program should include a piece like (18) expressing "if a cryptographer is the payer, then she lies on her announcement." Therefore such a program for DC is not typable.

In the DC protocol each cryptographer's behavior is certainly influenced by the information to be hidden; this is why the type systems from [45,36] do not seem to apply. Nevertheless, when three cryptographers are "composed" in the way prescribed by the protocol, they manage to hide who is the payer. In this sense, *broken anonymity is not compositional.* A type system which is applicable to verification of anonymizing protocols needs to suitably address such a non-compositional phenomenon. However potential complexity of such a type system would make it harder to come up with a useful typing algorithm.

6.2 Process algebraic techniques

In the current work we have presented anonymizing protocols in the form of automata, or transition systems. Using *process algebraic terms* or *process terms*—such as $P + Q$ for "non-deterministic choice between P and Q"—is another way of expressing processes such as anonymizing protocols. In fact, a process term induces a transition system via the structural operational semantics of the process algebra [2], hence a process algebra can be seen as a "programming language" which denotes transition systems.

One merit of using process algebras is that an algebraic structure in process terms often aids reasoning about properties of the term (more precisely, properties of the automaton denoted by the term). For example, we can employ inductive reasoning on the (inductive) construction of process terms. Many tools are available for analyzing properties of process terms.

Process algebraic techniques have been successfully applied in many anonymity applications. In [12], the MUTE anonymous file-sharing system [35] is analyzed by representing the protocol as a π-calculus term. The analysis led to discovery of a flaw, which can be detected by the ABC bisimulation checker [6] for the π-calculus. A detailed account of the flaw is in [14]. In [15] a general framework for automatic checking of anonymity is proposed; the framework is based on the process algebra μCRL. However, all the work mentioned here are done in non-deterministic settings. A probabilistic framework of such a kind which would aid automatic error-detection/verification of anonymity is one interesting direction of future work.

The *probabilistic π-calculus* [25,32] is a process algebra that can model probabilistic choices. A process term in that calculus yields a probabilistic automaton [39] as its operational model. It is used to model anonymity protocols in [5]. In a similar direction, the recent work [7] uses a variant of CCS with an additional operator for probabilistic choices.

6.3 Systems with both non-determinism and probability

The importance of having both of non-deterministic and probabilistic branching in system verification has been claimed by many authors, such as [38]; also in anonymity applications [31]. However our current method cannot handle this combination due to the lack of suitable coalgebraic framework.

In fact the combination of non-deterministic and probabilistic branching is a notoriously difficult one from a theoretical point of view [11,44,43]: many mathematical tools that are useful in a purely non-deterministic or probabilistic setting cease to work in the presence of both. Hence we take it as a fundamental and important challenge to find a suitable coalgebraic framework that accommodates the combined branching.

The notion of probabilistic automaton [39] is one of the standard models for probabilistic processes; it has both non-deterministic and probabilistic branchings. Trace semantics and simulations for probabilistic automata are studied in [38]. Soundness theorem—existence of a simulation yields trace inclusion—is also shown there.

However, one should be aware that the notion of "trace inclusion" in [38] is different from that in the current work. In [38], trace semantics $P_{\mathcal{X}}$ for a probabilistic automaton $\mathcal{X}$ is given by a set of probability distributions over lists on actions. An intuitive understanding is: $P_{\mathcal{X}}$ is the set of all the possible probabilistic behavior of $\mathcal{X}$, taken over all the choices of *schedulers*—mechanisms which resolve non-deterministic choices. Then the trace inclusion relation $P_{\mathcal{X}} \sqsubseteq P_{\mathcal{Y}}$ is defined to be simply the set inclusion $P_{\mathcal{X}} \subseteq P_{\mathcal{Y}}$. Therefore, in the setting of [38], trace inclusion does not refer to the magnitude of probabilities (such as "$\mathcal{Y}$ yields a list $\vec{a}$ with *more probability than* $\mathcal{X}$ does"). This is different from our notion of trace inclusion used in this paper. It is not clear whether simulations and trace semantics in [38] are useful in our current application of probabilistic anonymity.

Presence of schedulers causes another problem: it is just easy to think of a scheduler which collaborates with the adversary and makes anonymity fail. For example, suppose that we have an implementation of the DC protocol where we have a scheduler that resolves the non-deterministic choice on which cryptographer makes her announcement (agree or disagree) first. Then we can think of a scheduler which always put the payer's announcement in the last place. In fact, Tom Chothia [13] found out that this is how the Fedora JVM scheduler works for a certain implementation of DC in Java. There have been some work dealing with this problem. The work [16] proposes a class of *admissible* schedulers whose choices do not depend on the information to be disguised. In the work [7], a scheduler is explicitly written down as a process term; the designer of a protocol has control—in syntactic means—which actions/probabilistic choices should be visible to a scheduler.

7 Conclusion and future work

We have extended the simulation-based proof method [28,27] for non-deterministic anonymity to apply to the notion of strong probabilistic anonymity [5]. For the move we have exploited a generic theory of traces and simulations [20,23] in which the difference between non-determinism and probability is just a different choice of a parameter. Additionally a simulation-based proof method for the weaker notion of probable innocence is introduced.

The DC example in this paper fails to demonstrate the true potentiality 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 [27]. A probabilistic case study of such kind is currently missing.

Acknowledgments Thanks are due to Kostas Chatzikokolakis, Tom Chothia, Ken Mano, Catuscia Palamidessi, Peter van Rossum, Yasuaki Tsukada and the anonymous referees for the current and the earlier versions of the paper, 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] L. Aceto, W. Fokkink, and C. Verhoef. Structural operational semantics. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 197–292. Elsevier, 2001.

[3] Anonymity bibliography.
`http://freehaven.net/anonbib/`.

[4] M. Barr and C. Wells. *Toposes, Triples and Theories.* Springer, Berlin, 1985.

[5] 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.

[6] S. Briais. ABC bisimulation checker.
`http://lamp.epfl.ch/~sbriais/abc/abc.html`, 2003.

[7] K. Chatzikokolakis and C. Palamidessi. Making random choices invisible to the scheduler. In *International Conference on Concurrency Theory (CONCUR 2007)*, 2007. To appear.

[8] K. Chatzikokolakis, C. Palamidessi, and P. Panangaden. Probability of error in information-hiding protocols. In *IEEE Computer Security Foundations Symposium (CSF20)*, 2007. To appear.

[9] K. Chatzikokolakis and C. Palamidessi. Probable innocence revisited. *Theor. Comp. Sci.*, 367(1–2):123–138, 2006.

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

[11] L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices.* PhD thesis, Radboud Univ. Nijmegen, 2006.

[12] T. Chothia. Analysing the MUTE anonymous file-sharing system using the pi-calculus. In E. Najm, J.F. Pradat-Peyre, and V. Donzeau-Gouge, editors, *FORTE*, volume 4229 of *Lect. Notes Comp. Sci.*, pages 115–130. Springer, 2006.

[13] T. Chothia. How anonymity can fail because of the scheduler.
`http://homepages.cwi.nl/~chothia/DCschedular/`, 2007.

[14] T. Chothia. Securing pseudo identities in an anonymous peer-to-peer file-sharing network. In *International Conference on Security and Privacy in Communication Networks (SecureComm 2007)*, 2007. To appear.

[15] T. Chothia, S. Orzan, J. Pang, and M.T. Dashti. A framework for automatically checking anonymity with mCRL. In *The 2nd Symposium on Trustworthy Global Computing (TGC 2006)*, Lect. Notes Comp. Sci., 2006.

[16] F.D. Garcia, P. van Rossum, and A. Sokolova. Probabilistic anonymity and admissible schedulers. arXiv:0706.1019v1, 2007.

[17] 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.

[18] 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.

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

[20] 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.

[21] I. Hasuo and B. Jacobs. Context-free languages via coalgebraic trace semantics. In J.L. Fiadeiro, N. Harman, M. Roggenbach, and J. Rutten, editors, *International Conference on Algebra and Coalgebra in Computer Science (CALCO'05)*, volume 3629 of *Lect. Notes Comp. Sci.*, pages 213–231. Springer, Berlin, 2005.

[22] I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace theory. In N. Ghani and A.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.

[23] I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace semantics via coinduction. Extended version of [21,22], available from `http://www.cs.ru.nl/~ichiro`, 2007.

[24] I. Hasuo and Y. Kawabe. Probabilistic anonymity via coalgebraic simulations. In R. De Nicola, editor, *European Symposium on Programming (ESOP 2007)*, volume 4421 of *Lect. Notes Comp. Sci.*, pages 379–394. Springer, 2007.

[25] O.M. Herescu and C. Palamidessi. Probabilistic asynchronous pi-calculus. In J. Tiuryn, editor, *FoSSaCS*, volume 1784 of *Lecture Notes in Computer Science*, pages 146–160. Springer, 2000.

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

[27] 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.

[28] Y. Kawabe, K. Mano, H. Sakurada, and Y. Tsukada. Theorem-proving anonymity of infinite state systems. *Information Processing Letters*, 101(1):46–51, 2007.

[29] N. Lynch and F. Vaandrager. Forward and backward simulations. I. Untimed systems. *Inf. & Comp.*, 121(2):214–233, 1995.

[30] J. McLean. Security models. In J. Marciniak, editor, *Encyclopedia of Software Engineering*, pages 1136–1145. John Wiley & Sons, 1994.

[31] 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.

[32] C. Palamidessi and O.M. Herescu. A randomized encoding of the pi-calculus with mixed choice. *Theor. Comp. Sci.*, 335(2–3):373–404, 2005.

[33] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity: A proposal for terminology. Draft, version 0.17, July 2000.

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

[35] J. Rohrer. MUTE technical details. `http://mutenet.sourceforge.net/technicalDetails.shtml`, 2006.

[36] 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.

[37] 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.

[38] R. Segala. *Modeling and verification of randomized distributed real-time systems*. PhD thesis, MIT, 1995.

[39] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journ. Comput.*, 2(2):250–273, 1995.

[40] A. Serjantov. *On the Anonymity of Anonymity Systems*. PhD thesis, University of Cambridge, March 2004.

[41] V. Shmatikov. Probabilistic model checking of an anonymity system. *Journ. of Computer Security*, 12(3):355–377, 2004.

[42] G. Smith and D.M. Volpano. Secure information flow in a multi-threaded imperative language. In *POPL*, pages 355–364, 1998.

[43] R. Tix, K. Keimel, and G.D. Plotkin. Semantic domains for combining probability and non-determinism. *Elect. Notes in Theor. Comp. Sci.*, 129:1–104, 2005.

[44] D. Varacca and G. Winskel. Distributing probabililty over nondeterminism. *Math. Struct. in Comp. Sci.*, 16(1):87–113, 2006.

[45] D.M. Volpano and G. Smith. Probabilistic noninterference in a concurrent language. *Journ. of Computer Security*, 7(1), 1999.