Project no.:	**PIRSES-GA-2011-295261**
Project full title:	**Mobility between Europe and Argentina applying Logics to Systems**
Project Acronym:	**MEALS**
Deliverable no.:	**3.4 / 1**
Title of Deliverable:	**Dissecting Unlinkability**

Contractual Date of Delivery to the CEC:	**1-Apr-2013**
Actual Date of Delivery to the CEC:	**15-Mar-2013**
Organisation name of lead contractor for this deliverable:	**INR**
Author(s):	**Mayla Brusó, Konstantinos Chatzikokolakis Sandro Etalle, Jerry den Hartog**
Participants(s):	**INR,TUE,UNC,UBA**
Work package contributing to the deliverable:	**WP3**
Nature:	**R**
Dissemination Level:	**Public**
Total number of pages:	**39**
Start date of project:	1 Oct. 2011 **Duration:** 48 month

Abstract:

Unlinkability is a privacy property of crucial importance for several types of systems, such as RFID or voting systems. Informally, unlinkability states that, given two events/items in a system, an attacker is not able to infer whether they are related to each other.
However, in the literature we find several definitions for this notion, which are apparently unrelated and shows a potentially problematic lack of agreement.
This paper sheds new light on unlinkability by comparing different definitions from the literature and showing that in many practical situations they actually coincide. It does so by (a) expressing in a unifying framework four different existing definitions of unlinkability (a weak and three strong ones) (b) demonstrating how these definitions are different yet related to each other and to their dual notion of "inseparability" and (c) identifying conditions under which all these definitions become equivalent. We argue that these conditions are reasonable to expect in identification systems, and we prove that they hold for a generic class of protocols. Finally, we use our framework to define the stronger notions of forward and backward privacy.

This project has received funding from the European Union Seventh Framework Programme (FP7 2007-2013) under Grant Agreement Nr. 295261.

Contents

1 Introduction

Unlinkability is a privacy property which holds when an attacker cannot identify the link between two or more items in a system. For instance, one may expect that a protocol for online purchases prevents a seller from linking a customer to his/her browsing history (e.g. purchases at other shops); similarly, an e-voting protocol should prevent anybody from linking the vote to the identity of the voter. These examples show that unlinkability is fundamental for the development of new protocols in the context of e-commerce applications or identification systems.

In particular, Radio Frequency Identification (RFID) are systems where the unlinkability property is extremely relevant. RFID systems are a wireless technology for automatic identification consisting of a set of tags, readers and a backend. Tags are typically simple battery-less devices consisting of a tiny chip and an antenna offering very limited resources. Readers, on the other hand, have standard computational resources and communicate with tags wirelessly. Readers are connected to a backend system, where all the data required for the identification of tags are stored. An identification protocol allows tags to authenticate to a backend exchanging information through a reader.

The wireless nature of RFID makes access to tags extremely easy. Nowadays they are commonly used in supply chain management as a replacement for barcodes and are starting to make their way into the consumer realm: it is not uncommon to find RFID chips in clothes (often with a label "remove after purchase", although this suggestion is rarely followed). As time passes, people will increasingly often carry RFID tags around with them.

As one may expect, one of the main issues raised by the widespread use of RFID is that of privacy. Consumers already rose against famous brands like Benetton because they tried to introduce EPC (RFID replacement for barcodes) in some of their products. Also, consumer groups started a campaign against Gillette, for using RFID tags in razor blades to track their products. Clearly, a breach in these systems might lead to the disclosure of private data. The problem is that anyone in the neighbourhood of a tag may access it wirelessly, and the resource limitation of RFID tags makes it difficult to use full-fledged cryptographic algorithms (implementing e.g. two-way authentication). The ease of access paves the way to misuse: an attacker could exploit tags to follow the movements of people or goods. To do so, the attacker does not even need to break anonymity (i.e. to obtain the tag identity): a tag sending in plain text the expiry date of a product does not reveal its identification number but does allow a certain degree of tracking.

Privacy concerns such as those just mentioned lead to the definition of the concept of *unlinkability* (sometimes called *untraceability* or simply *privacy*). In the case of RFID systems, unlinkability [18, 22, 4, 5, 23, 8, 29] is satisfied if an attacker is not able to distinguish between tags; that is, she is not able to tell whether the tag she communicates with in a certain session is one of the tags she communicated with in a previous session. In the RFID literature, unlinkability is usually defined in terms of games in a computational setting [10, 18, 22, 4, 23]. More recently, several works focused on unlinkability properties for RFID systems in a symbolic setting [27, 28, 2, 3, 7]. Van Deursen et al. [27] propose a definition of untraceability in a trace-based model. Arapinis et al. [2, 3] formalize protocols in the applied pi-calculus and define weak and strong unlinkability in terms of trace equivalence and observational equivalence respectively. Finally, [7] proposes definitions of unlinkability and forward privacy in the applied pi-calculus,

inspired by the unlinkability games of the computational setting.

As most definitions are different in model and strength, there is clearly no agreement in the literature on the concept of unlinkability. The goal of this paper is to create a better understanding of this notion by comparing the strength of different definitions and determining whether the differences have a practical impact on real world systems. Technically, our contribution is:

1. First, we express four trace-based definitions of unlinkability from the literature in a unifying model, stated in terms of attacker's knowledge using epistemic logic. We start with the one of weak unlinkability from [27, 3]. By strengthening this definition in a natural way, we obtain the definition of strong unlinkability from [3]. Then, we express two game-based definitions, which have been used mostly in the computational [10, 18, 4, 21] but also in the formal setting [7]. Furthermore, we investigate *inseparability*, a notion dual to unlinkability, which requires that the attacker cannot infer that two messages are *not* linked. We give a weak and a strong variant, in correspondence to the respective unlinkability properties. We give examples to show that these notions are actually different.

2. Second, we identify a set of conditions and demonstrate that, when these conditions hold, all the above forms of unlinkability and inseparability coincide. We argue that these conditions are common to most of the identification systems, such as RFID systems.

3. Third, we prove that these conditions are indeed satisfied by a generic class of simple identification protocols from [7]. As corollary of this and the previous result, we conclude that weak unlinkability, for this family of protocols, provides much stronger privacy guarantees than it seems at first sight.

4. Last, starting from the definitions of unlinkability, we define the notions of forward and backward privacy, which assume a stronger attacker able to disclose all the secret information of an agent.

These results help us to understand the essence of these privacy properties. Working in an abstract setting, we can concentrate on their inherent nature – the inability to distinguish certain traces – without dealing with the complications of a concrete model. As a result, the definitions and the conditions under which they coincide become intuitive, while the results can be transferred to a concrete trace model as we do in Section 6.

Plan of the paper. Section 2 briefly introduces epistemic logic. Section 3 presents our abstract trace model. Section 4 states several definitions of privacy using epistemic logic and gives simple examples showing that these properties do not coincide in general. Section 5 presents several conditions and shows that, whenever they hold, all the privacy properties coincide. Section 6 presents the class of RFID single-step protocols and shows that for these protocols all the conditions stated in the previous section hold. Section 7 shows the definitions of forward and backward privacy as special cases of unlinkability Section 8 lists the related work. Section 9 provides conclusions.

2 Preliminaries

In this section we briefly introduce epistemic logic with public announcements, a logic modelling agent knowledge, that we later use to formalize privacy properties. Only the basic concepts are stated here, we refer the reader to [19] for more details.

Let P be a set of propositional constants (atoms). The set $\mathfrak{L}(P)$ of epistemic formulas φ, ϕ, ... over A is given by:

$$\varphi, \psi \quad ::= \quad p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid K\varphi \mid [\varphi]\psi$$

with $p \in P$. The first four formulas are standard and represent an atom, the notation for *not*, *and* and *or* respectively. The formula $K\varphi$ means "the attacker knows φ". Note that epistemic logic typically involves multiple agents, with K_i denoting the knowledge of agent i. For simplicity, we only consider a single agent, namely the global attacker. Finally, $[\varphi]\psi$ means "after φ is revealed, ψ holds".

The semantics is given in terms of Kripke structures. A Kripke structure M is a tuple $(S, f, \sim)$ where:

- S is a set of possible states;
- $f : S \to 2^P$ is a function assigning to each state a set of atoms that hold in that state;
- $\sim$ is an equivalence relation on S for the attacker.

Intuitively, $s_1 \sim s_2$ means that from the attacker's point of view, the two states are indistinguishable. The semantics of the logic is given by:

- $M, s \vDash p$ iff $p \in f(s)$;
- $M, s \vDash \varphi \wedge \psi$ iff $M, s \vDash \varphi$ and $M, s \vDash \psi$;
- $M, s \vDash \varphi \vee \psi$ iff $M, s \vDash \varphi$ or $M, s \vDash \psi$;
- $M, s \vDash \neg\varphi$ iff $s \nvDash \varphi$;
- $M, s \vDash K\varphi$ iff $s' \vDash \varphi$ for all the states s' such that $s' \sim s$.

$M, s \vDash \varphi$ means that M satisfies φ at state s; we simply write $s \vDash \varphi$ when M is clear from the context. The interesting case is the knowledge formula $K\varphi$: the attacker knows φ at state s iff φ is satisfied in all the states that are indistinguishable from s from the attacker's point of view.

Finally, let $M|\varphi$ be a Kripke structure obtained from M by restricting it to states that satisfy φ, i.e. having state space $S' = \{s \in S \mid M, s \vDash \varphi\}$. We define

- $M, s \vDash [\varphi]\psi$ iff ($M, s \vDash \varphi$ implies $M|\varphi, s \vDash \psi$)

Intuitively, revealing $[\varphi]$ in a state where φ holds, restricts the model to a smaller one where φ always holds. $[\varphi]\psi$ is true if ψ holds in the restricted model.

3 A trace-based model

In this section we present our model, which will be the basis for formalizing several privacy properties using epistemic logic in Section 4.

In a system, agents exchange messages according to a protocol with a specific purpose. Most communication takes place over an insecure channel, so an attacker is able to intercept and forge messages. In order to capture one or more protocol runs in our model, we introduce the concept of *transactions*. A transaction starts when the attacker gains access to an agent and lasts until the attacker loses this access. During the transaction the attacker can passively eavesdrop the communications or actively modify and forge messages. We allow the attacker to execute an arbitrary number of protocol sessions within a transaction, while knowing that the agent participating in the transaction does not change. However, when a new transaction starts, the agent involved can be either the same as before, or a different one, and the attacker's goal is to distinguish these two cases.

The attacker's intentions are captured by the concept of a *strategy*. For example, the attacker might passively eavesdrop the session of the first agent she sees, then build a message from that agent output and send it to a second agent, then interact with a third agent and so on. This strategy involves three transactions, and defines the messages sent to the agents in each transaction. On the other hand, the attacker does not control which agent will be involved in each transaction. Different agents will clearly lead to different executions producing different messages.

In our model, an attacker strategy, together with a mapping of transactions to agent identities, completely defines one of the possible executions of the system. Abstracting from the protocol details, we consider traces composed of strategies and agent mappings, and we define privacy properties based on the attacker's ability to distinguish these traces.

Definition 1. A system is a tuple $(A, \Sigma, \mathrm{T}, \sim)$ where:

- $A = \{a_1, a_2, \ldots\}$ is a (possibly infinite) set of agents; we assume an ordered set of transactions $\{p_1, p_2, \ldots\}$, and we denote by $\Pi_n = \{p_1, \ldots, p_n\} \to A$ the set of assignments of n transactions to agents;
- Σ is a set of strategies; each strategy $\sigma \in \Sigma$ has a length $|\sigma|$; we denote the set of transactions involved in the strategy by $Dom_\sigma = \{p_1, \ldots, p_{|\sigma|}\}$;
- $\mathrm{T} = \{(\pi, \sigma) \mid \sigma \in \Sigma, \pi \in \Pi_{|\sigma|}\}$ is the set of traces; each trace $\tau \in \mathrm{T}$ is a tuple (π, σ) where $\sigma \in \Sigma$ is a strategy and $\pi \in \Pi_{|\sigma|}$ is a mapping of transactions involved in σ to agents;
- $\sim$ is an equivalence relation on T such that $(\pi_1, \sigma_1) \sim (\pi_2, \sigma_2) \Rightarrow \sigma_1 = \sigma_2$.

Intuitively, Σ is the set of all strategies that the attacker can employ. A trace $\tau \in \mathrm{T}$ is a complete execution of the system, and is determined by a strategy σ (chosen by the attacker), and a mapping π (which the attacker does not control) that defines which agent participates in each transaction. In our model, a protocol is an abstract object that describes the behaviour of the agents in a system. Each protocol generates the set T of all the possible traces that can be obtained under any attacker strategy $\sigma \in \Sigma$. The relation $\sim$ is crucial for defining privacy

properties. Consider a trace $\tau_1 = (\pi_1, \sigma)$, produced when the attacker chooses the strategy σ and interacts with the agents in π_1, and a trace $\tau_2 = (\pi_2, \sigma)$ produced by the same strategy when different agents are involved. If $\tau_1 \sim \tau_2$, it means that when using the strategy σ, the attacker cannot tell which of the agents she was interacting with.

We sometimes use π_τ to emphasize that it belongs to the trace τ. For a trace $\tau = (\pi, \sigma)$, we write Dom_τ for $Dom(\sigma)$, $|\tau|$ for $|\sigma|$ and A_τ, A_π for the image of π (i.e. the set of agents involved in the trace). For a mapping $\pi \in \Pi_n$ we define $|\pi| = n$ and we denote $\Pi = \cup_{n \geq 1} \Pi_n$. Since transactions are ordered, we write mappings as sequences of agents, e.g. $\pi = (a_4, a_1, a_2)$ assigns agent a_4 to the first transaction, a_1 to the second and a_2 to the third ones.

We extend $\sim$ to mappings as follows:

$$\pi \sim \pi' \quad \text{iff} \quad |\pi| = |\pi'| \text{ and } (\pi, \sigma) \sim (\pi', \sigma)\ \forall \sigma \in \Sigma \text{ s.t. } |\sigma| = |\pi|$$

Note that we keep our model abstract and do not explicitly define the messages in the protocol, the exact strategies Σ and the relation $\sim$. We assume that these are produced by a concrete protocol model (such as the one given in Section 6).

RFID systems. We use RFID systems as a case study of a real-world system where unlinkability is crucial. An RFID system consists of a number of tags and readers. A reader can communicate wirelessly with a tag in its proximity in order to identify it. An attacker can interact with a tag, although she does not know its identity (otherwise unlinkability would not possibly hold). A transaction happens when the attacker has access to a tag. For example, the attacker sees a tag at the entrance of a building at 12:00 and can interact with it. The attacker can query the tag many times at each transaction, and she can be sure that the same tag responds every time. When she loses proximity to the tag, this transaction ends. Then, at the same or some other location, the attacker sees once again a tag and can interact with it in a different transaction. Note that transactions play a similar role to the "interfaces" of [7], used to point out that the attacker knows she is continuously communicating with the same tag. For simplicity, we assume that all interactions happen between tags and the attacker (honest sessions are also allowed, since the attacker can forward messages between agents).

Our model can be instantiated into a concrete protocol model, such as those of [27, 3]. For example, when using the model of [3] based on the applied pi-calculus, a strategy corresponds to a context modelling the attacker, τ is the trace produced by the corresponding process, and $\sim$ is static equivalence between traces. This "instantiation" of our model is used in Section 6 to show that a class of protocols, expressed in the model of [3], satisfies a set of conditions expressed in our abstract model.

4 Unlinkability definitions

In this section we express several definitions of unlinkability from the literature in our trace-based model. Each definition is given in an intuitive form using epistemic logic, and also in a direct form in terms of trace equivalence. First, we state the definition of weak unlinkability from

[27, 3]. This definition requires that for every pair of transactions (p, p') the attacker cannot *know* whether they are linked. Then we move on to strong unlinkability for which there are several definitions in the literature. We first express the one of [3], requiring that every trace is equivalent to one without any linked messages (i.e. the attacker cannot even know about the existence of a link). Then, we focus on game-based definitions of unlinkability, which appear in the literature mostly in the computational [10, 18, 4, 21], but also in the formal setting [7]. Such definitions set up a game between an attacker and a challenger, in which the former tries to distinguish between situations created by the latter. We express two such definitions in our model, corresponding to different types of games, in terms of equivalence of the traces provided by the challenger or chosen by the attacker. Then, we give our definition of game-based unlinkability which captures both types of game. Note that our purpose is not a technical comparison between the computational and formal models. Instead, we are interested in the idea behind each definition, so we first express the computational definitions in our trace-based model, and then we compare them to the rest of the definitions within this formal model. Finally, we introduce the concept of inseparability, a dual notion to unlinkability, requiring that the attacker cannot infer whether transactions are *not* linked to each other. Although inseparability has not been previously studied in the literature, it arises naturally from the definitions of weak and strong unlinkability, thus we believe that investigating its relationship to the unlinkability definitions is of interest. Table 10 gathers all the resulting epistemic notions (Definition 4 is not reported because it does not have an epistemic counterpart, but it is equivalent to (c.) and (d.)).

4.1 Kripke structure

To express the various notions of unlinkability using epistemic logic, the first step is to define a Kripke structure M, starting from the system described in Section 3.

We recall that T is the set of all traces of our system and $\sim$ is an equivalence relation on T, representing the fact that an adversary cannot distinguish between those executions. An equivalence between executions is a fundamental concept underlying all definitions of unlinkability in the literature. In formal models, such an equivalence has been expressed either in terms of process equivalence (e.g. observational or trace equivalence in the applied pi calculus [3, 7]) or using the concept of "reinterpretation" [27]. In computational models, the equivalence is stated in terms of the inability of the adversary to distinguish the two cases in the corresponding game.

More concretely, starting from a system $(A, \Sigma, \mathrm{T}, \sim)$ we build a Kripke structure $M = (\mathrm{T}, f, \sim)$. We assume that the set A contains at least two agents. The set of states is T and the indistinguishability relation of the attacker $\sim$ is provided directly by the system. Moreover the set of atomic propositions P and the assignment function $f : \mathrm{T} \to P$ are built as follows:

$$P = \Pi \cup \{link(p, p') \mid p, p' \in Dom_\tau, p \neq p', \tau \in \mathrm{T}\}$$
$$f((\pi, \sigma)) = \{\pi\} \cup \{link(p, p') \mid \pi(p) = \pi(p')\}$$

We use two types of propositions: $\pi \in \Pi$ simply denotes that the mapping of the trace is π; $link(p, p')$ denotes that the transactions p, p' are linked, and it is true in a trace $\tau = (\pi, \sigma)$ iff both transactions are mapped to the same agent in that trace. Note that the proposition $link(p, p')$ holds only if $p \neq p'$; thus, we implicitly assume $p \neq p'$ in all the epistemic definitions.

4.2 Weak Unlinkability

The first definition is the one of weak unlinkability of van Deursen et al. [27] and Arapinis et al. [3]. Although presented in different models, the two definitions are similar in nature. They require that, given a trace where two messages are linked (sent by the same agent) an equivalent trace must exist where the corresponding messages are not linked. This can be expressed in an intuitive way using epistemic logic:

Definition 2 (Weak unlinkability)**.** A protocol generating the set of traces T guarantees weak unlinkability iff

$$\forall \tau \in \mathrm{T}, p, p' \in Dom_\tau : \tau \vDash \neg K(link(p, p'))$$

This definition states that a protocol is weakly unlinkable when the attacker does not know whether any two given transactions are linked to each other. This implies that for all traces τ and all pairs of distinct transactions, there must exist an equivalent trace $\tau' \sim \tau$ in which the corresponding transactions are mapped to two different agents. So the above definition can be written as:

$$\forall \tau \in \mathrm{T}, p, p' \in Dom_\tau : \exists \tau' \in \mathrm{T}, \tau' \sim \tau : \tau' \vDash \neg link(p, p')$$

(note that if $\tau \vDash \neg link(p, p')$ then we can simply take $\tau' = \tau$). This formulation of unlinkability corresponds exactly to the ones of [27, 3].

The weakness of this definition lies in the quantification. In particular, the attacker might still be able to infer that a transaction is linked to some other one in the trace, without being able to tell which one, as illustrated in the example below.

Example 1. Let T be the set of traces generated by a given protocol. Let $\tau_1 = (\pi_1, \sigma)$ and $\tau_2 = (\pi_2, \sigma)$ be two equivalent traces in T such that:

- $\pi_1 = (a_1, a_1, a_2)$;
- $\pi_2 = (a_1, a_2, a_1)$.

Assume that the equivalence classes of equivalence are $\{\{\tau_1, \tau_2\}, \mathrm{T} \setminus \{\tau_1, \tau_2\}\}$, i.e. the attacker can distinguish τ_1, τ_2 from other transactions in the system. We assume that the protocol generating T guarantees weak unlinkability. In fact, each pair of linked transactions in τ_1 is unlinked in τ_2 and vice versa. Still, the attacker knows that p_1 is linked to either p_2 or p_3 in the trace τ_1 (or τ_2); formally, $\tau_1 \vDash K(link(p_1, p_2) \vee link(p_1, p_3))$.

4.3 Strong Unlinkability

Arapinis et al. [3] define also a strong version of unlinkability by requiring that a system is equivalent to one where each agent executes a single protocol session. Their definition, in a simplified form and without entering into the details of the applied pi calculus, requires that:

$$!\,T \approx !\,T_s \quad \text{where} \tag{1}$$
$$T = \nu m.\ init.\ !main$$
$$T_s = \nu m.\ init.\ main$$

where T represents an agent carrying out an initialization phase ($init$) and then an unbounded number (denoted by !) of protocol sessions ($main$), while T_s is an agent executing a single session. $\approx$ denotes observational equivalence in [3], while we use trace equivalence here since it is directly expressable in our model.

To capture this definition in our framework, we not only require that the attacker is not able to infer the link between two given transactions, but also the *existence* of linked transactions. We define:

$$anyLink(\tau) = \bigvee_{p \in Dom_\tau} \bigvee_{p' \in Dom_\tau} link(p, p')$$

Intuitively, $anyLink(\tau)$ holds for a trace if there exists at least one linked transaction. We can now define strong unlinkability as:

Definition 3 (Strong unlinkability)**.** We say that a protocol generating the set of traces T guarantees strong unlinkability iff

$$\forall \tau \in \mathrm{T} : \tau \vDash \neg K(anyLink(\tau))$$

Strong unlinkability holds when the attacker does not know whether there is a link in a trace at all. For this to hold, each trace must be equivalent to one where no transaction is linked, namely:

$$\forall \tau \in \mathrm{T} : \exists \tau' \in \mathrm{T}, \tau' \sim \tau : \forall p, p' \in Dom_\tau : \tau' \vDash \neg link(p, p')$$

This formulation corresponds exactly to (1) since τ' is a trace that can be produced by the process $!T_s$, where no agent executes more than one transaction.

4.4 Game-based definitions of privacy

We now turn our attention to the game-based definitions of privacy. In these definitions, privacy is defined as the result of a game between an attacker (whose goal is to distinguish between the actions of different agents) and a challenger.

In this section we refer to two different types of game-based definition of privacy: the first is related to the definition given by Ohkubo et al. [22], variations of which can be found also in [10, 18, 4, 23], while the second corresponds to the definition given by [10, 21]. We believe that most of the game-based definitions of privacy given in the literature fall under one of these two types.

Here we demonstrate that in our model, these two classes of definitions are equivalent to each other and to a third *simpler* definition of game-base unlinkability based on trace equivalence. Since the challenge involves two agents in the first type of game and three agents in the second one, we name the corresponding properties two-agents game unlinkability and three-agents game unlinkability, respectively.

Both types of games consist of three phases. In the game of the first type, during the first phase, the attacker is allowed to interact with all the agents of the system. In the second phase, the attacker is asked to select two agents a, a'. Then, the challenger selects an agent $x \in \{a, a'\}$, and gives x back to the attacker for interactions, hiding its identity. The attacker is still allowed to interact with all agents in the system, including a and a'. During the final phase, the attacker

responds to the challenge and wins the game if she can infer whether x is a or a' with non-negligible probability.
In order to express this definition in our framework, we need to introduce some notation:

- π_x is a partial mapping from transactions to agents, where some transactions are mapped to a variable x. This represents the situation in which the attacker knows the identities of the agents involved in all the transactions of a trace, but the ones mapped into x.
- Π_x denotes the set of all partial mappings.
- π_a is a complete mapping obtained from π_x by mapping to an agent a all transactions previously mapped to the variable x.

In our model, we formalize the unlinkability game by requiring that the attacker cannot infer whether she is given a mapping π_a or $\pi_{a'}$, where a and a' correspond to the agents selected by the challenger during the second phase of the game. Formally, we say that a protocol generating the set of traces T guarantees *two-agents game unlinkability* iff

$$\forall \tau \in \mathrm{T}, a, a' \in A, \pi_x \in \Pi_x : \tau \vDash [\pi_a \vee \pi_{a'}] \neg K(\pi_a) \tag{2}$$

Note that, although the only forbidden knowledge concerns π_a, (2) is in fact equivalent to $\tau \vDash [\pi_a \vee \pi_{a'}] \neg K(\pi_{a'})$, thus the attacker is not allowed to know $\pi_{a'}$ either. For two-agents game unlinkability to hold, the two mappings π_a and $\pi_{a'}$ should be equivalent under all strategies, thus we can equivalently express (2) as:

$$\forall a, a' \in A, \pi_x \in \Pi_x : \pi_a \sim \pi_{a'}$$

We now turn our attention to the second type of game. This variation can be mostly found in the computational setting [10, 21]; however, in [7], a similar definition is given in a formal model. As for the previous game, during the first phase the attacker is allowed to interact with all the agents. Then, she selects three agents a, a_1, a_2 to be challenged on. In some of the referenced works the choice of the agents participating in the challenge is left unspecified. We follow the strongest approach of giving this choice to the attacker. The challenger gives to the attacker access to two agents x, y. The challenger can either set $x = y = a$ or $x = a_1, y = a_2$. In the last phase, the attacker wins the game if she can infer whether x and y are linked.
Again, we need to introduce some notation:

- $\pi_{x,y}$ is a partial mapping from transactions to agents, where some transactions are mapped to a variable x and some others to a variable y. This represents the situation in which the attacker knows the identities of the agents involved in all the transactions but the ones mapped into x and y.
- $\Pi_{x,y}$ denotes the set of all the partial mappings.
- $\pi_{a,b}$ is a complete mapping obtained from $\pi_{x,y}$ by mapping to an agent a all transactions previously mapped to the variable x and an agent b to the variable y.

We require that the attacker cannot infer whether she is given a mapping $\pi_{a,a}$ or π_{a_1,a_2}. Formally, we say that a protocol generating the set of traces T guarantees three-agents game unlinkability iff

$$\forall \tau \in \mathrm{T}, a, a_1, a_2 \in A, \pi_{x,y} \in \Pi_{x,y} : \tau \vDash [\pi_{a,a} \vee \pi_{a_1,a_2}] \neg K(\pi_{a,a}) \tag{3}$$

In terms of equivalence of traces, (3) can be restated as follows:

$$\forall \tau \in \mathrm{T}, a, a_1, a_2 \in A, \pi_{x,y} \in \Pi_{x,y} : \pi_{a,a} \sim \pi_{a_1,a_2}$$

It is easy to see that both two-agents game unlinkability and three-agents game unlinkability requires all the mappings to be equivalent. Therefore we give a definition of game-based unlinkability which unifies the two notions given above:

Definition 4 (Game-based unlinkability)**.** We say that a protocol generating the set of traces T guarantees game-based unlinkability iff

$$\forall \pi, \pi' \in \Pi, |\pi| = |\pi'| : \pi \sim \pi' \tag{4}$$

Each of the referenced works uses a variant of either (2) or (3), while [10] mentions both, referring to two-agents game unlinkability as untraceability and to three-agents game unlinkability as unlinkability, but does not explore the relation between the two. Instead, we can demonstrate that both definitions reduce to Definition 4:

Theorem 1. *A protocol satisfies game-based unlinkability if and only if it satisfies two-agents game unlinkability, which it does if and only if it satisfies three-agents game unlinkability.*

From now on we will only use the definition of game-based unlinkability. However, by Theorem 1, all the results that involve game-based unlinkability in the following hold also for two-agents game unlinkability and three-agents game unlinkability.

Proof. First, we prove that both two-agents game unlinkability and three-agents game unlinkability imply game-based unlinkability, which corresponds to proving that all the traces are equivalent to each other. To show this, we demonstrate that, for any given length n, all the traces are equivalent to the trace executed entirely by a single agent.

- *Two-agents game unlinkability*. For each trace τ, we need to consider its mapping $\pi_\tau = (\pi_\tau(1), \ldots, \pi_\tau(n))$. We define i as the first mapping index such that $\pi_\tau(1) \neq \pi_\tau(i)$ and we choose π_x such that all the occurrences of the agent $\pi_\tau(i)$ in the mapping π_τ are replaced by x. Then we set $a = \pi_\tau(1)$ and $a' = \pi_\tau(i)$. By (2), the original trace τ is equivalent to a trace τ' that has the same mapping except for the occurrences of agent $\pi_\tau(i)$, which are replaced by a mapping to agent $\pi_\tau(1)$. The trace τ' has now one less agent in its mapping with respect to τ. If we repeat the same reasoning on τ' until the number of agents of the resulting trace is 1, we can conclude that τ is equivalent to the trace executed by a single agent by transitivity.
  For example, consider a trace with mapping $\pi_\tau = (a_1, a_2, a_3, a_2)$. The first agent different from a_1 is a_2 on the transaction $i = 2$, thus we replace all the occurrences of

a_2 by x, obtaining $\pi_x = (a_1, x, a_3, x)$. By (2), we have that $\pi_{a_1} \sim \pi_{a_2} = \pi_\tau$, i.e. $(a_1, a_1, a_3, a_1) \sim (a_1, a_2, a_3, a_2)$. Then, we choose $\pi'_x = (a_1, a_1, x, a_1)$, and by (2) we have that $\pi'_{a_1} \sim \pi'_{a_3}$, i.e. $(a_1, a_1, a_1, a_1) \sim (a_1, a_1, a_3, a_1)$. By transitivity, we can conclude that the original mapping π_τ is equivalent to π'_{a_1}, which is executed by a single agent, namely $\pi_\tau = (a_1, a_2, a_3, a_2) \sim (a_1, a_1, a_1, a_1) = \pi'_{a_1}$.

- *Three-agents game unlinkability*. For each trace τ, we consider its mapping $\pi_\tau = (\pi_\tau(1), \ldots, \pi_\tau(n))$ and we define i as the first mapping index such that $\pi_\tau(1) \neq \pi_\tau(i)$. Then, we choose $\pi_{x,y}$ such that $\pi_\tau(1)$ is replaced by x and all the occurrences of the agent $\pi_\tau(i)$ by y. Then we set $a = a' = \pi_\tau(1)$ and $a'' = \pi_\tau(i)$. By 3, the original trace τ is equivalent to a trace τ' that has the same mapping except for the occurrences of agent $\pi_\tau(i)$, which are replaced by a mapping to agent $\pi_\tau(1)$. Again, this method must be applied until the resulting trace is run by a single agent, which, by transitivity, is equivalent to τ.

We can conclude that when (2) or (3) hold, then all the traces are equivalent, therefore also Definition 4 holds.
Now we need to prove that Definition 4 in turn implies two-agents game unlinkability and three-agents game unlinkability. This implication trivially holds, since they both require *specific* pairs of mappings to be equivalent while Definition 4 already implies the equivalence of *any* pair of mappings. □

Finally, as one may expect, we can show that strong unlinkability and game-based unlinkability are both stronger than weak unlinkability:

Theorem 2. *Strong unlinkability and game-based unlinkability imply both weak unlinkability.*

Proof. First, we prove that strong unlinkability implies weak unlinkability, namely that:

$$\forall \tau \in \mathrm{T} : \tau \vDash \neg K(anyLink(\tau)) \Rightarrow \forall \tau \in \mathrm{T}, p, p' \in Dom_\tau : \tau \vDash \neg K(link(p, p'))$$

This is a tautology in the epistemic logic, as shown below. Note that in the proof we use the following property of K (Prop. K) in its contrapositive form: $\bigvee_i K(P_i) \Rightarrow K \bigvee_i (P_i)$ (this is a tautology for any predicate P_i).
For all $\tau \in \mathrm{T}$ we have:

$$\begin{array}{lll}
\tau \vDash \neg K(anyLink(\tau)) & & \\
\quad \equiv & \text{[Def. } anyLink(\tau)\text{]} & \tau \vDash \neg K \bigvee_p \bigvee_{p'} link(p, p') \\
\quad \Rightarrow & \text{[Prop. } K\text{]} & \tau \vDash \neg \bigvee_p K \bigvee_{p'} link(p, p') \\
\quad \equiv & & \forall p \in Dom_\tau : \tau \vDash \neg K \bigvee_{p'} link(p, p') \\
\quad \Rightarrow & \text{[Prop. } K\text{]} & \forall p \in Dom_\tau : \tau \vDash \neg \bigvee_{p'} K(link(p, p')) \\
\quad \equiv & & \forall p, p' \in Dom_\tau : \tau \vDash \neg K(link(p, p'))
\end{array}$$

Therefore strong unlinkability implies weak unlinkability.
Second, we want to prove that game-based unlinkability implies weak unlinkability. This follows directly from the definition of game-based unlinkability which trivially implies the equivalence of any trace, thus also weak unlinkability. □

In section 4.6 it is shown that weak unlinkability actually differs from all the strong definitions of unlinkability, and that strong unlinkability is incomparable to game-based unlinkability.

4.5 Inseparability

In some situations we want to hide the existence of *unlinked* transactions instead of linked ones. For instance, an attacker might be interested in changes in the system rather than in tracking agents. Examples where this might be useful to the attacker are easily found in access control systems. For example, consider a high security location protected by a guard who authenticates himself using an RFID tag. Since the guard is the same every day, all the authentication messages are obviously linked, therefore this information is useless to the attacker. However, the attacker might want to be alerted when a *new* guard appears. The definitions of weak and strong unlinkability impose no condition when two messages are unlinked, so they offer no protection in this case. To model similar situations, we need to introduce the concept of *inseparability* which requires that the attacker does not know that two specific transactions are *not* linked.

Definition 5 (Weak inseparability)**.** We say that a protocol generating the set of traces T guarantees weak inseparability iff

$$\forall \tau \in \mathrm{T}, p, p' \in Dom_\tau : \tau \vDash \neg K(unlink(p, p')) \tag{5}$$

where $unlink(p, p') = \neg link(p, p')$.

The definition states that the attacker should not be able to infer whether any two given transactions are not linked. Thus, if there is a pair of unlinked transactions in the trace τ then there must exists an equivalent trace where the same transactions are linked to each other. Then, (5) corresponds to:

$$\forall \tau \in \mathrm{T}, p, p' \in Dom_\tau : \tau \vDash unlink(p, p') \Rightarrow \exists \tau' \in \mathrm{T}, \tau' \sim \tau : \tau' \vDash link(p, p')$$

Similarly to the case of weak unlinkability, this definition does not capture the situation in which the attacker is able to infer the *existence* of two unlinked transactions. This brings us to the definition of a stronger notion of inseparability.

Definition 6 (Strong inseparability)**.** A protocol generating the set of traces T guarantees strong inseparability iff

$$\forall \tau \in \mathrm{T} : \tau \vDash \neg K(anyUnlink(\tau))$$

where $anyUnlink(\tau) = \bigvee_p \bigvee_{p' \neq p} unlink(p, p')$

Note that, similarly to unlinkability, also the definitions of inseparability can be stated in a direct way in terms of trace equivalence.
As expected, strong inseparability is stronger than weak inseparability. On the other hand, somewhat surprisingly, game-based unlinkability turns out to be stronger than strong inseparability, although game-based unlinkability is incomparable to strong unlinkability, which is incomparable to strong inseparability. The reason is that game-based unlinkability implies that all the

traces are equivalent to each other, thus also to the one run by a single agent, as required by strong inseparability. Game-based unlinkability does not explicitly talk about linked or unlinked messages, therefore it offers both "unlinkability" and "inseparability" guarantees.

Theorem 3. *Game-based unlinkability implies strong inseparability, which implies weak inseparability.*

Proof. The first part of the theorem states that the game-based definition of strong unlinkability implies strong inseparability. This follows directly by Theorem 1. In fact, strong inseparability holds when all the traces are equivalent to a trace executed by one agent. Game-based unlinkability holds when all the possible traces are equivalent; in particular, they must be equivalent to the traces executed by one agent, and this corresponds exactly to the definition of strong inseparability.
For the second part of the theorem, the steps showing that strong inseparability implies weak inseparability are the same as those for unlinkability. □

The above properties are in general different. In the following section we provide examples that illustrate the differences between those properties and later we investigate conditions under which some or all of the properties become equivalent.

4.6 RFID systems: protocols where the properties do not coincide

In this section we list some examples of RFID protocols that guarantee only some of the properties described in Section 4. The examples are constructed and do not correspond to real protocols. Indeed, the differences between the properties arise from features of the examples that are unlikely to be present in realistic protocols. In the next section, we will identify some conditions under which weak and strong properties become equivalent. The examples are variations of the OSK protocol for RFID systems [22], which guarantees strong unlinkability [7]. In the original protocol, each tag is initialized with a unique secret. During each session, the tag outputs $h(x)$ and updates its current state with $g(x)$, where h and g are two distinct hash functions, and x its current state. Note that a full understanding of the protocol is not required in order to follow the examples.

Example 2 (System with a bounded number of tags). Consider a system with a bounded number of tags. If the attacker observes a number of sessions greater than the number of tags, she knows that there exist some linked sessions, although she is not able to point to any specific message, i.e. weak unlinkability holds, but strong unlinkability does not. Note that the definition of strong unlinkability from [3] in terms of the process equivalence (1) implicitly assumes the existence of an unbounded number of tags. Still, all the traces of equal length produced by the OSK protocol are equivalent, therefore game-based unlinkability holds. Strong inseparability also holds, because the restriction on the number of tags does not affect the equivalence of all the possible traces to the one completely linked, which exists for any length in this system.

Example 3 (System with several "types" of tags). Consider a system in which there are two types of tags, $Type_1$ and $Type_2$, that the attacker can distinguish, for example because tags

with different technical characteristics are used. Weak inseparability and strong inseparability are violated: when two transactions of different types are observed, the attacker knows that the transactions cannot come from the same tag. Game-based unlinkability is also violated, because the adversary can trivially distinguish some traces from others (e.g. one completely linked from one executed by tags of different type). On the other hand, strong unlinkability still holds: the adversary cannot know the existence of any linked transactions since all transactions of the same type could come from different tags. Together with the previous example, this shows that strong unlinkability and the game-based definitions are incomparable.

However, if the number of tags of $Type_2$ is bounded, we have a situation similar to the previous example (although the total number of tags might still be unbounded). Strong unlinkability is violated if the number of protocol sessions belonging to tags of $Type_2$ in the trace is greater than the number of tags of $Type_2$, but weak unlinkability still holds.

Example 4 (Protocol outputs depending on past sessions I). Consider a variation of the OSK protocol where the reader does some observable action (a "beep") when the session it is executing is linked to a previous session, but only if at least two tags appeared in the past sessions of the trace. This protocol satisfies weak unlinkability: the beep does not allow the attacker to point to any two specific sessions and tell that they are linked. Despite this, the attacker knows that the session that made the reader beep must be linked to a past session of the trace. Consider the observation that provides more knowledge to the attacker, namely the one where the reader beeps at the third session. The beep tells him that the third session is either linked to the first or the second one, violating strong unlinkability. Since not all mappings are equivalent to each other due to the observable action, game-based unlinkability is also violated. Finally, weak inseparability and strong inseparability are violated too, because the trace which causes the reader to beep at the third session tells the attacker that the first two sessions are *not* linked.

Example 5 (Protocol outputs depending on past sessions II). Consider a variation of the OSK protocol where the reader beeps when the third tag of a trace first appears. This example is similar to the previous one, but it leads to the definition of a different condition in the next section. The protocol satisfies weak unlinkability, but violates strong unlinkability: if the reader beeps after four or more sessions, there must exist a link in the previous sessions. Game-based unlinkability is also violated, since different traces lead to different observations. Finally, it breaks weak inseparability and strong inseparability, since a beep during the third session means that the first three sessions are not linked.

Example 6 (Protocol outputs depending on past sessions III). Consider a variation of the OSK protocol where the reader beeps when it sees at least two tags and one link. With respect to unlinkability, the protocol behaves similarly to the previous ones: it satisfies weak unlinkability but violates strong unlinkability and game-based unlinkability. With respect to inseparability, strong inseparability is violated, because a beep means that at least two tags were involved in that trace. However, the attacker cannot point to two sessions and claim that they are unlinked; therefore, weak inseparability still holds.

Table 2 summarizes the relationship between the properties and lists the references to the corresponding examples and theorems.

5 Conditions under which the properties coincide

In Section 4 we gave three definitions of unlinkability as well as two of inseparability, and we showed examples where these definitions do not coincide. However, these examples have features that are unlikely to be found in practice. In this section we identify a (large) class of protocols C and we demonstrate for these protocols that all the notions of unlinkability and inseparability are equivalent to each other: if a protocol in C satisfies any of them, then it satisfies all of them. The class C is given by all the protocols satisfying the five conditions that we identify in the next section, where we also argue that most RFID protocols actually satisfy them, at least in their abstract form. We also show that the class of single-step protocols is contained in C.

5.1 Conditions

5.1.1 Condition Unbounded number of agents.

As we showed in Example 2, a system with a bounded number of agents cannot satisfy strong unlinkability, since observing a greater number of transactions reveals that at least two transactions are linked. Thus, protocols in C contain an unbounded number of agents.

Definition 7 (Unbounded number of agents)**.** We say that a protocol has an unbounded number of agents iff

$$\forall n > 0 \;\; \exists \tau \in \mathrm{T} : \quad |A_\tau| = n$$

Clearly, an unbounded number of agents cannot exist in a real identification system. However, such systems are usually needed to identify a wide number of agents, therefore an attacker cannot usually communicate with all the agents in a limited amount of time. Moreover, the attacker does not usually know the number of agents in the system at all. This is why, at an abstract level, this condition is often assumed in the literature.

5.1.2 Condition Renaming.

As shown in Example 3, having multiple distinguishable types of agents can be problematic. For instance, observing two transactions of different type clearly violates inseparability, as we can conclude that the transactions are not linked. However, in real systems agents are usually identical in functionality, differing only in the secret key used for their identification. Indeed, agents usually identify themselves to a system by means of similar or identical devices, e.g. RFID tags or desktop computers. As a result, we can expect that replacing *all* transactions of an agent with a new one (not participating in other transactions) will not have a visible effect, since the two agents are identical except for their secret information. We capture this by the following condition:

Definition 8 (Renaming)**.** Let π be a mapping, $a \in A_\pi$ and $a' \notin A_\pi$. The renaming of a to a' in π, denoted by $\pi[a'/a]$, is a mapping such that

$$\pi[a'/a](p) = \begin{cases} a' & \text{if } \pi(p) = a \\ \pi(p) & \text{otherwise} \end{cases}$$

We say that a protocol satisfies the condition Renaming iff $\pi \sim \pi[a'/a]$ for all mappings π and agents $a' \notin A_\pi$.

In other words, for protocols satisfying the condition Renaming, the only thing that matters is the positions in which an agent appears in the trace, and not the exact identity of the agent. For example, the mappings $(a_3, a_1, a_2, a_1, a_1)$ and $(a_3, a_4, a_2, a_4, a_4)$ should be equivalent (i.e. produce equivalent traces under all strategies σ) since they are identical, except for the renaming of a_1 to a_4.

In the rest of the paper, we assume that this condition holds and we write all mappings in a normalized form, sorting the agents by their order of appearance: the first agent is always a_1, the next agent different from a_1 will be a_2 and so on. For example, we write $(a_1, a_1, a_2, a_3, a_1, a_2)$ instead of $(a_5, a_5, a_3, a_1, a_5, a_3)$ since these mappings are assumed to be equivalent. The main advantage of this normalization is that the number of possible traces is always finite for any given length (even when the number of agents is infinite).

5.1.3 Condition Swapping.

Consider $\pi_1 = (\ldots, a_1, a_2, \ldots)$ and $\pi_2 = (\ldots, a_3, a_4, \ldots)$, two mappings where the k-th and $k+1$-st transactions involve different agents (which could appear elsewhere in the mapping). Now assume that $\pi_1 \sim \pi_2$, that is no attacker strategy can distinguish these mappings, and consider the mappings $\pi_1' = (\ldots, a_2, a_1, \ldots)$ and $\pi_2' = (\ldots, a_4, a_3, \ldots)$ that differ from π_1 and π_2 only for the k-th and $k+1$-st agents, which have been swapped. We require that different agents act in an independent way and the execution of the one should not affect the execution of the other. The transactions of a_1, a_3 should not depend on whether a_2, a_4 were previously executed or not and vice versa. Thus, π_1' and π_2' should also be indistinguishable. Formally, we require the following condition:

Definition 9 (Swapping)**.** Let π be a mapping. The swapping of π at position $k < |\pi|$, denoted by $sw_k(\pi)$ is a new mapping such that:

$$sw_k(\pi)(p_i) = \begin{cases} \pi(p_{k+1}) & \text{if } p_i = p_k \\ \pi(p_k) & \text{if } p_i = p_{k+1} \\ \pi(p_i) & \text{otherwise} \end{cases}$$

We say that a protocol satisfies the condition Swapping iff

$$\pi \sim \pi' \Rightarrow sw_k(\pi) \sim sw_k(\pi')$$

for all π, π', k such that $\pi(p_k) \neq \pi(p_{k+1})$ and $\pi'(p_k) \neq \pi'(p_{k+1})$.

In practice, the condition Swapping simply implies that the agents are independent of each other. As a consequence, the order of appearance of agents does not affect the knowledge of the attacker.

Note that this condition is violated in the Example 4. Consider a passive attacker that eavesdrops three legitimate sessions; we use "$\_$" and "$beep$" for "reader does nothing" and "reader beeps" respectively. The mappings (a_1, a_1, a_2) and (a_1, a_2, a_3) produce the same observations $(\_, \_, \_)$. However, by swapping the second and third transactions, we obtain the mappings (a_1, a_2, a_1) and (a_1, a_2, a_3), which produce different observations: $(\_, \_, beep)$ and $(\_, \_, \_)$. Clearly, the problem here is that the execution of one agent depends on the previous executions of other agents. Note that (a_1, a_2, a_3) does not change because it is written in canonical form. (a_1, a_2, a_3) and (a_1, a_3, a_2) are equivalent due to the condition Renaming.

With regard to RFID systems, note that the condition Swapping defined above has no relation to the swapping attack of [17]. In fact, while in the attack of [17] some messages of two sessions are swapped in order to swap their labels, the condition swapping assumes that two *entire* sessions are swapped in the trace.

5.1.4 Conditions: Extension I and II.

We now introduce two last conditions. The first states that two equivalent mappings should preserve their equivalence when extended with a new transaction mapped to a fresh agent, i.e. not appearing in the original mapping. Similarly to the swapping rule, the underlying idea is that the execution of an agent should not depend on the previous executions of other agents. Therefore, adding a new agent should not make two traces distinguishable, if they could not be distinguished before. This is formally stated in the following condition.

Definition 10 (Extension I)**.** Let π be a mapping. The extension of π with a new agent $a \notin A_\pi$, denoted by $extn(\pi)$, is a mapping of length $|\pi| + 1$ such that

$$extn(\pi)(p_i) = \begin{cases} \pi(p_i) & i \leq |\pi| \\ a & i = |\pi| + 1 \end{cases}$$

We say that a protocol satisfies the condition Extension I if

$$\pi \sim \pi' \Rightarrow extn(\pi) \sim extn(\pi')$$

for all mappings π, π'.

Note that this condition is violated by the protocol in the Example 5. Consider a passive attacker that eavesdrops three legitimate sessions: if the attacker observes no beep, she knows that they could come from the mappings (a_1, a_1, a_1), (a_1, a_1, a_2), (a_1, a_2, a_1) or (a_1, a_2, a_2); if we add a new tag to all the mappings, the equivalence is not preserved since the mapping (a_1, a_1, a_1, a_2) does not make the reader beep, while (a_1, a_1, a_2, a_3), (a_1, a_2, a_1, a_3) and (a_1, a_2, a_2, a_3) do, thus the condition Extension I does not hold.

The second extension condition works as follows: consider two equivalent mappings π_1, π_2 of length n. We extend these mappings with a new transaction p_{n+1}, which is mapped to the last agent appearing in each mapping. Intuitively, since the attacker had access to these agents during the p_n-th transaction, and she could not distinguish the two mappings, she does not gain any new knowledge by querying the same agents in the new transactions. The idea is that what the attacker can do in the transaction p_{n+1} to disclose information could already be done in p_n in the original mapping. Thus, we require the following condition.

Definition 11 (Extension II)**.** Let π be a mapping. The extension of π with the last appearing agent, denoted by $extl(\pi)$ is a mapping of length $|\pi| + 1$ such that

$$extl(\pi)(p_i) = \begin{cases} \pi(p_i) & i \leq |\pi| \\ \pi(p_{|\pi|}) & i = |\pi| + 1 \end{cases}$$

We say that a protocol satisfies the condition Extension II if

$$\pi \sim \pi' \Rightarrow extl(\pi) \sim extl(\pi')$$

for all mappings π, π'.

This condition is violated by the protocol in the Example 6. In fact, the traces with mappings (a_1, a_1) and (a_1, a_2) are not distinguishable; instead, their extensions producing the mappings (a_1, a_1, a_1) and (a_1, a_2, a_2) are distinguishable because the first trace produces no beep while the second beeps.

5.2 Equivalence results

In this section we state the main results of this paper. Namely, we demonstrate that under the conditions stated in the previous section, all the definitions of unlinkability coincide. Moreover, we prove that, under a smaller set of conditions, the definitions of inseparability coincide as well as all the strong definitions (of both unlinkability and inseparability).

Theorem 4 (Unification of unlinkability)**.** *If a protocol guarantees all the following conditions:*

- *Unbounded number of agents*
- *Renaming*
- *Swapping*
- *Extension I and II*

then all the unlinkability properties (weak unlinkability, strong unlinkability, game-based unlinkability) coincide.

The intuition is that, under these conditions, all the definitions require all the mappings of the same length to be equivalent under all the possible attacker strategies. In particular, this implies that all the traces are equivalent to one where all the transactions are not linked, which implies both weak unlinkability and strong unlinkability.

Proof. We first prove that weak unlinkability implies the game-based notion of strong unlinkability. In particular, we need to show that weak unlinkability together with the conditions imply that all the traces in T are equivalent under any attacker strategy $\sigma \in \Sigma$, namely:

$$\forall(\tau, \tau') \in \mathrm{T} : \pi_\tau \sim \pi_{\tau'}$$

The proof is by induction on the number of transactions n and is split in two parts: one for systems with two agents only and the other for systems with more than two agents.

Case with two agents

Base case. For $n = 1$ we only have a single trace (under the condition Renaming) so there is nothing to prove. For the case $n = 2$ we have two possible mappings $\pi = (a_1, a_1)$ and $\pi' = (a_1, a_2)$. By weak unlinkability, under any attacker strategy the traces with mapping π must be equivalent to traces where the link is broken, i.e. with mapping π'. Thus all the mappings are equivalent.

Inductive case. We have to show that all the mappings of length n (> 2) are equivalent. The induction hypothesis gives that all mappings with length $n - 1$ are equivalent. We divide the mappings of length n over three sets that contain mappings equivalent to all the other mappings in that set. We use π^{-i} to denote the sequence of the first $n - i$ agents in a trace.

- EE Mappings ending in two transactions executed by the same agent (π^{-2}, x, x). They are equivalent to each other according to the induction assumption plus condition Extension II.
- DE_1 Mappings of the form (π^{-3}, x, y, x) with $x \neq y$. Any two mappings in this set can be obtained from mappings in EE by swapping at position $n - 2$ (when possible). So they are equivalent by the Swapping condition and (EE) (note that we write EE and DE_1 for the sets and (EE) and (DE_1) for the corresponding equivalence properties).
- DE_2 Mappings of the form (π^{-3}, x, x, y) with $x \neq y$. Any two mappings in this set can be obtained from mappings in DE_1 by swapping at position $n - 1$. So they are equivalent by the condition Swapping and (DE_1).

As there are only two agents in the system, these sets together cover all traces.

Note that any trace in EE has the last two transitions linked. By weak unlinkability, each mapping must be equivalent to a mapping in which these two transactions are not linked, i.e. a mapping not in EE. Similarly for DE_1 at the last and third last and DE_2 at the second last and third last positions. Therefore in each set there is a mapping that is related to a mapping outside the set, and thus in one of the other two sets. As we have only three sets and equivalence is transitive this is sufficient to conclude that all mappings are equivalent.

Case with more than two agents

Base cases. The base cases are the followings:

- $n = 1, 2$: With at most two transactions there are at most two agents involved. The proof remains the same as in the case with two agents.
- $n = 3$: There are five possible mappings: (a_1, a_1, a_1), (a_1, a_1, a_2), (a_1, a_2, a_1), (a_1, a_2, a_2), (a_1, a_2, a_3). We have to show that they are all equivalent:
  - The condition Extension I applied on the case $n = 2$ gives $(a_1, a_1, a_2) \sim (a_1, a_2, a_3)$.
  - The condition Swapping applied on the last two positions of these two traces gives $(a_1, a_2, a_1) \sim (a_1, a_2, a_3)$ (note that swapping has no effect on (a_1, a_2, a_3) due to the renaming to the canonical form). If we apply the condition again to now swap the first two positions we obtain: $(a_1, a_2, a_2) \sim (a_1, a_2, a_3)$ (note the renaming from (a_2, a_1, a_1) to canonical form (a_1, a_2, a_2)).
  - (a_1, a_1, a_1) must be equivalent to one of the other four (equivalent) mappings by weak unlinkability.

Inductive case. For the inductive case we have to show that all the mappings with length n are equivalent. We use a_i^m for a sequence of m times the agent a_i. The induction hypothesis is that all equal-length mappings with length up to $n - 1$ are equivalent. We give four sets of mappings that are all equivalent to $({a_1}^n)$ and thus to each other.

EE Mappings ending in two transactions with the same agent (π^{-2}, x, x). They are equivalent to each other (including $({a_1}^n)$) according to the induction assumption plus condition Extension II.

FR Mappings ending in a transactions with a fresh agent (π^{-1}, y), $y \notin \pi^{-1}$. They are equivalent to each other according to the induction assumption plus condition Extension I. They are equivalent to $({a_1}^n)$ because: $(a_1^{n-4}, a_1, a_1, a_2, a_2) \sim (a_1^{n-4}, a_2, a_3, a_1, a_1)$ by (EE) so, by using the condition Swapping twice to move the second last transaction to position $n - 4$ and twice more to move the last transaction to place $n - 3$, we get: $(a_1^{n-4}, a_2, a_2, a_1, a_1) \sim (a_1^{n-4}, a_1, a_1, a_2, a_3)$ with the former in EE and the latter in FR.

ST Mappings of the form (a_1^k, a_2, a_1^l) $(k + l = n - 1)$ are equivalent to $({a_1}^n)$. This is clear for $l \geq 2$ by (EE) and for $l = 0$ by (FR). For $l = 1$ consider: $(a_1^{n-2}, a_1, a_2) \sim (a_1^{n-2}, a_2, a_3)$ (by (FR)). By using the condition Swapping on position $n-1$ we get also $(a_1^{n-2}, a_2, a_1) \sim (a_1^{n-2}, a_2, a_3)$ (note that swapping has no effect on the latter mapping due to the use of normal forms by renaming), with the former in ST and the latter in FR.

SW Swaps of mappings equivalent to $({a_1}^n)$. If a mapping π is equivalent to $({a_1}^n)$ then we have $\pi \sim (a_1^{k-1}, a_2, a_1^{n-k})$ by (ST) so also $sw_k(\pi) \sim ({a_1}^k, a_2, a_1^{n-k-1})$ by the condition Swapping. But then $sw_k(\pi) \sim ({a_1}^n)$ by (ST).

These sets contain all traces; any trace that is not in FR is of the form (π_1, x, π_2, x) with $x \notin \pi_2$ which we obtain from (π_1, π_2, x, x) in (EE) by swapping π_2 times. Therefore we can conclude that all the mappings are equivalent, therefore game-based unlinkability holds. Note that there is no need to use the unbounded number of agents condition for the game-based definition.

Now we have to prove that weak unlinkability implies strong unlinkability under the conditions. From the previous proof, we know that the conditions, with the exception of the unbounded number of agents condition, are sufficient to prove that all the traces generated by a given protocol are equivalent under any attacker strategy. The existence of an unbounded number of agents implies that among the equivalent traces there is always the one where all the transactions are not linked, which is the requirement for strong unlinkability to hold. □

Theorem 5 (Unification of inseparability)**.** *If a protocol guarantees the following conditions:*

- *Renaming*
- *Extension II*

then it satisfies weak inseparability iff it satisfies strong inseparability.

Again, under these conditions, both properties require all mappings of the same length to be equivalent; in particular they are equivalent to one where all the transactions are linked.

Proof. We want to prove that, under the above conditions, whenever a protocol guarantees weak inseparability it also guarantees strong inseparability. Therefore, in our trace-based model we need to prove that all the traces with the same length generated by a protocol are equivalent. The proof is by induction on the number of sessions (n).

Base case. ($n = 2$) We have to prove that all the traces in T such that $|\pi_\tau| = 2$ are equivalent under any attacker strategy σ. The only possible mappings are $\pi = (a_1, a_1)$ and $\pi' = (a_1, a_2)$. They are trivially equivalent by weak inseparability. Therefore, all the traces are equivalent for $n = 2$.

Inductive case. ($n > 2$) By inductive hypothesis all the traces $\tau \in \mathrm{T}$ such that $|\pi_\tau| \leq n - 1$ are equivalent under any attacker strategy σ. Therefore, by the condition Extension II, we have that all the traces $\tau \in \mathrm{T}$ such that $|\pi_\tau| = n$ and $\pi_\tau(p_{n-1}) = \pi_\tau(p_n)$ are equivalent under any attacker strategy σ. By weak inseparability, the remaining traces $\tau \in \mathrm{T}$ such that $|\pi_\tau| = n$ and $\pi_\tau(t_{n-1}) \neq \pi_\tau(p_n)$ must be equivalent to a trace where the last two transactions are linked, namely to the ones of the previous case. Therefore, by transitivity of $\sim$ we can conclude that all the traces such that $|\pi_\tau| = n$ are equivalent. □

The previous theorems compare properties of each family (unlinkability or inseparability). We now show that unlinkability and inseparability also coincide under certain conditions.

Theorem 6 (Unification of strong properties)**.** *If a protocol guarantees the following conditions:*

- *Unbounded number of agents*
- *Renaming*

then all the strong properties (strong unlinkability, game-based unlinkability, strong inseparability) coincide.

It is worth noting that this result uses weaker assumptions than the previous ones; indeed, the conditions Swapping and Extension I and II are not needed. An unbounded number of agents is required to guarantee the existence of the traces where all transactions are mapped to different agents (for strong unlinkability), while the condition Renaming implies that agents are not observationally different (for strong inseparability and game-based unlinkability). Example 3 (with two distinguishable types of tags) shows that a protocol, without the condition Renaming, can satisfy strong unlinkability while violating game-based unlinkability and strong inseparability. On the other hand, Example 2 shows that, without the unbounded number of agents, a protocol can violate strong unlinkability while satisfying game-based unlinkability and strong inseparability.

Finally, we can state the result we were aiming at.

Corollary 1. *If a protocol guarantees all the following conditions:*

- *Unbounded number of agents*
- *Renaming*
- *Swapping*
- *Extension I and II*

then all the forms of unlinkability and inseparability coincide.

This result shows that all the privacy definitions of Section 4 coincide under a set of conditions.

6 RFID systems: single-step protocols

In Section 5 we showed that, under some conditions, all the unlinkability and inseparability definitions coincide. In this section, we show that these conditions are indeed satisfied by a generic class of "single-step" protocols [7]. To this end, we express such protocols in the applied pi calculus, using the model of [3], which defines an instantiation of our model, i.e. it provides a concrete set of traces and an equivalence relation between them.

Single-step protocols consist of a single message from a tag to a reader. A protocol of this class is shown in Figure 1. Each tag is initialized with an internal state S_0, which contains a secret s that is unique to that tag (e.g. a nonce or private key). At each run of the protocol, the tag computes a function $O(S)$, where S is its current state, and outputs the result to the reader. This message should contain sufficient information for the reader to identify the tag. Then, the tag computes a second function $U(S)$ and updates its current state with the result. Although the protocol is restricted to a single message, O and U can be arbitrary, using any cryptographic operation (provided that it can be modelled by an equational theory in the applied pi calculus).

As discussed in [7], two well-known protocols from the literature, namely the OSK protocol [22] and the basic hash protocol of [30], fall in this class.

6.1 Modelling single-step protocols

We model single-step protocols in the applied pi calculus [1], a language for describing concurrent processes and their interaction. It extends the pi calculus [20] adding the possibility to model cryptographic primitives using a signature and an equational theory. A detailed description of the calculus is available in [1]; here, we only assume a basic understanding of the calculus, technical details are only needed to follow the proofs.

Tags are modelled as processes in the calculus, using a public channel c to communicate with the reader. The main challenge comes from the fact that tags are *stateful*. To model their state, we use an internal channel w. The content of the state is available to the tag by a sub-process $\overline{w}\langle S\rangle$ running in parallel to it, so the tag can read the state by an input on w. Therefore, a tag execution can be modelled in the applied pi calculus as follows:

$$TagExec \triangleq c(\_).\ w(x).\ \nu\tilde{\rho}.\ \overline{c}\langle O(x)\rangle.\ \overline{w}\langle U(x)\rangle$$

The tag is first triggered by an input on the public channel c (but without reading a value, which is denoted by the use of "_" instead of a variable). Then, it reads the current state x by an input on w and outputs $O(x)$ on a public channel. $\nu\tilde{\rho}$ denotes the possibility of generating fresh nonces for the output. Finally, the tag outputs $U(x)$ on w, updating its state with the new value.

A complete tag starts with an initial state S_0, containing the tag secret s, and can execute an unbounded number of sessions.

$$Tag \triangleq \nu s.\nu w.(\overline{w}\langle S_0\rangle \mid !TagExec)$$

Note that the secret is private to the tag, thus we use a freshly generated name s. We also restrict w so that only the tag can access its state.

Finally, the complete system P consists of an unbounded number of tags: $P \triangleq !Tag$. Note that the reader in single-step protocols is passive: it only triggers the tag. Since c is public, the tag can be triggered by any external process, so we can simply omit the reader altogether.

6.2 Instantiating our trace model

The system P can perform labelled transitions, according to the semantics of the applied pi calculus. We denote by $\stackrel{\alpha}{\Longrightarrow}$ a sequence of internal transitions, followed by the visible transition α, followed again by internal transitions. A trace is a sequence

$$tr = P \stackrel{\alpha_1}{\Longrightarrow} P_1 \stackrel{\alpha_2}{\Longrightarrow} P_2 \stackrel{\alpha_3}{\Longrightarrow} \ldots \stackrel{\alpha_q}{\Longrightarrow} P_q$$

Two traces are equivalent, denoted by $tr_1 \sim_{tr} tr_2$ if they contain the same transitions, and all the intermediate processes are *statically* equivalent according to the definition of [1], which states that the processes provide the attacker with the same static knowledge. We refer to [3] for the formal definition of $\sim_{tr}$.

To instantiate our trace model, we need do define a set of agents A, a set of strategies Σ such that a strategy σ together with a mapping π give rise to a trace $\tau = (\pi, \sigma)$, and an equivalence relation $\sim$ between traces.

The agents $A = \{a_i | i \in \mathbb{N}\}$ correspond to the tags of the system. Note that in the applied pi calculus model, we use replication to denote an unbounded number of tags. We identify the tags by their secret s, which is restricted inside the replication, thus it is unique for each tag. When a_i is spawned we denote its secret by s_i.

Since tags in single-step protocols have no input, the only thing that the attacker can decide is how many transactions she will run, and how many protocol executions she will trigger in each transaction. Thus, a strategy σ is a sequence $\sigma = (\sigma_1, \ldots, \sigma_k)$ such that k is the number of transactions the attacker performs, and σ_i the number of executions that she triggers in the transaction i. A mapping π determines which tag will participate in each transaction, for example $\pi(p_2) = a_3$ means that a_3 is the tag involved in the second transaction p_2.

Given a strategy σ and a mapping π, we can define a unique trace $tr(\pi, \sigma)$ starting from P. In this trace, the tag $\pi(p_1)$ is first spawned and runs σ_1 executions. Then, the tag $\pi(p_2)$ is spawned (unless $\pi(p_1) = \pi(p_2)$, meaning that it had already been spawned) and runs σ_2 executions, and so on. Finally, we define trace equivalence as follows:

$$(\pi, \sigma) \sim (\pi, \sigma') \quad \text{iff} \quad tr(\pi, \sigma) \sim_{tr} tr(\pi, \sigma')$$

Now that we have a concrete trace model for single-step protocols, we can show that they satisfy all the conditions of Section 5.1.

Theorem 7. *Single-step protocols satisfy all the following conditions:*

- *Unbounded number of agents*
- *Renaming*
- *Swapping*
- *Extension I and II*

Therefore all the unlinkability and inseparability properties coincide.

Proof. A single-step protocol in the concrete model provides traces of the form:

$$\begin{array}{lllll} P & \stackrel{c(-)}{\Longrightarrow} & A_{1,1} & \stackrel{\nu x_{1,1}.\overline{c}\langle x_{1,1}\rangle}{\Longrightarrow} & A'_{1,1} \\ & \cdots & & & \\ & \stackrel{c(-)}{\Longrightarrow} & A_{i,j} & \stackrel{\nu x_{i,j}.\overline{c}\langle x_{i,j}\rangle}{\Longrightarrow} & A'_{i,j} \\ & \cdots & & & \\ & \stackrel{c(-)}{\Longrightarrow} & A_{n,\sigma_n} & \stackrel{\nu x_{n,\sigma_n}.\overline{c}\langle x_{n,\sigma_n}\rangle}{\Longrightarrow} & A'_{n,\sigma_n} \end{array}$$

where:

- the indexes (i, j) are used to indicate the i-th transaction at the j-th query;
- $A_{i,j}$ corresponds to the process obtained after triggering the tag during the specific session (i, j);

- $A'_{i,j}$ corresponds to the process obtained after the tag has sent out the output in the session (i, j);
- $x_{i,j}$ is the variable that contains the term sent over the network.

We have to prove that single-step protocols guarantee all the conditions of Section 5.1.

First we define our notation and some lemmas used later on in the proofs of the conditions. We use $\Phi_{\pi,\sigma}(k)$ to denote the frame produced by a trace (π, σ) at the k-th transaction:

$$\Phi_{\pi,\sigma}(k) = \prod_{i=1}^{\sigma_k} \nu\tilde{\rho}.\{^{O(U^{I(k,i)}(s_{\pi_k}))}/_{x_{k,i}}\}$$

where $I(k, i) = i - 1 + \sum_{\pi_j^{j<k}=\pi_k} \sigma_i$ is the number of sessions executed before the i-th session of the k-th transaction by the agent π_k.
We use π^l and σ^l as shorts for $sw_l(\pi)$ and $sw_l(\sigma)$ respectively, denoting that the l-th agent and the l-th strategy have been swapped with the $l + 1$-st respectively.
From now on, we will consider single-step protocols as modelled in Section 6.

The first lemma simplifies the concept of trace equivalence, so that to prove trace equivalence it suffices to prove that the traces can produce the same inputs and outputs and that the final processes are statically equivalent.

Lemma 1. *Consider two processes A and B in canonical form that process the same sequences of inputs and outputs. If every last pair of processes produced by the traces that A and B can generate are statically equivalent, then the corresponding traces are equivalent.*

Proof. Consider two traces τ_A and τ_B, generated by the processes A and B such that

$$\tau_A = A \stackrel{\alpha_1}{\Longrightarrow} A_1 \stackrel{\alpha_2}{\Longrightarrow} \ldots \stackrel{\alpha_{n-1}}{\Longrightarrow} A_{n-1} \stackrel{\alpha_n}{\Longrightarrow} A_n$$

$$\tau_B = B \stackrel{\alpha_1}{\Longrightarrow} B_1 \stackrel{\alpha_2}{\Longrightarrow} \ldots \stackrel{\alpha_{n-1}}{\Longrightarrow} B_{n-1} \stackrel{\alpha_n}{\Longrightarrow} B_n$$

Each transition can only augment a frame, namely it can only add variables, but it cannot destroy them or change their content. In practice, $\phi(A_i)$ always has all the variables in $\phi(A_{i-1})$ and possibly more. Clearly, all the terms bounded to the variables in $\phi(A_{i-1})$ do not contain any of the new variables present in the frame $\phi(A_i)$. We want to prove that if the frames $\phi(A_n)$ and $\phi(B_n)$ are equivalent then the traces τ_A and τ_B are equivalent, namely that also the processes obtained in the previous steps are statically equivalent:

$$\forall i : \phi(A_i) \approx_s \phi(B_i)$$

The processes A and B have the following form:

$$\nu\tilde{s}.(!P_1, \ldots, !P_p)$$

where $\tilde{s}$ is the sequence of all the names restricted in $!P_1, \ldots, !P_p$ and the subprocesses P_i represent the type of agent in the system.

The traces τ_A and τ_B generate a sequence of n frames. Each of the α transitions either produce a new substitution or does not change the frames. When the frames do not change from the $i-1$-st to the i-th transition, it is trivial to show that the $i-1$-st transition produced a pair of statically equivalent frames, since the i-th pair is statically equivalent by inductive hypothesis. Thus, for the sake of simplicity, in the rest of the proof we only consider the transitions that produce a substitution. Therefore, the n-th frames correspond to:

$$\phi(A_n) = \nu\tilde{s}.\{\{{}^{T_1}/_{x_1}\}, \ldots, \{{}^{T_n}/_{x_n}\}\}$$
$$\phi(B_n) = \nu\tilde{s}.\{\{{}^{T'_1}/_{x_1}\}, \ldots, \{{}^{T'_n}/_{x_n}\}\}$$

where T_i and T'_i indicate the terms produced by the i-th transitions of the traces τ_A and τ_B.
By hypothesis, the processes A_n and B_n are statically equivalent, namely $\phi(A_n) \approx_s \phi(B_n)$. We prove by induction on the number of transitions $i \leq n$ that all the frames are statically equivalent. By inductive hypothesis, we know that:

$$\phi(A_i) \approx_s \phi(B_i)$$

This can be written as:

$$\nu\tilde{s}.(\varphi_{A_{i-1}} \mid \{{}^{T_i}/_{x_i}\}) \approx_s \nu\tilde{s}.(\varphi_{B_{i-1}} \mid \{{}^{T'_i}/_{x_i}\})$$

where $\varphi_{A_{i-1}}$ and $\varphi_{B_{i-1}}$ contains all the substitutions generated until the $i-1$-st transition and $\tilde{n}$ contains all the restricted names, including possible names generated by the i-th transition. Static equivalence is closed under the application of closing evaluation contexts, therefore we can restrict the variable x_i to obtain the following equivalent frames:

$$\nu x_i.\nu\tilde{s}.(\varphi_{A_{i-1}} \mid \{{}^{T_i}/_{x_i}\}) \approx_s \nu x_i.\nu\tilde{n}.(\varphi_{B_{i-1}} \mid \{{}^{T'_i}/_{x_i}\})$$

Since the variable x_i is not used in $\varphi_{A_{i-1}}$ nor in $\varphi_{B_{i-1}}$, we can move its restriction, obtaining two structurally equivalent frames:

$$\nu\tilde{s}.(\varphi_{A_{i-1}} \mid \nu x_i.\{{}^{T_i}/_{x_i}\}) \approx_s \nu\tilde{n}.(\varphi_{B_{i-1}} \mid \nu x_i.\{{}^{T'_i}/_{x_i}\})$$

By structural equivalence $\nu x_i.\{{}^{T_i}/_{x_i}\} = \nu x_i.\{{}^{T'_i}/_{x_i}\} = 0$; also, the parallel composition of a process with the 0 process is structurally equivalent to the process, therefore we have:

$$\nu\tilde{s}.(\varphi_{A_{i-1}}) \approx_s \nu\tilde{n}.(\varphi_{B_{i-1}})$$

Which corresponds exactly to:

$$\phi(A_{i-1}) \approx_s \phi(B_{i-1})$$

Thus, we can conclude that all the pairs of processes in τ_A and τ_B are statically equivalent. □

Lemma 2. *Let*

$$\phi_A = \nu\tilde{\rho}.\{\varphi_1, {}^{M_1}/_{x_1}, \ldots, {}^{M_m}/_{x_m}, {}^{N_1}/_{x_{m+1}}, \ldots, {}^{N_n}/_{x_{m+n}}, \varphi_2\}$$
$$\phi_B = \nu\tilde{\rho}.\{\varphi'_1, {}^{M'_1}/_{x_1}, \ldots, {}^{M'_m}/_{x_m}, {}^{N'_1}/_{x_{m+1}}, \ldots, {}^{N'_n}/_{y_{m+n}}, \varphi'_2\}$$

be frames in canonical form such that $\phi_A \approx_s \phi_B$, and Θ a substitution function such that:

$$\Theta(x_i) = \begin{cases} x_{((i+m-1) mod (n+m))+1} & 1 \leq i \leq n+m \\ x_i & \textit{otherwise} \end{cases}$$

If $\phi'_A = \phi_A\Theta$ and $\phi'_B = \phi_B\Theta$ then $\phi'_A \approx_s \phi'_B$.

Proof. By hypothesis we know that

$$dom(\phi_A) = dom(\phi_B) \wedge \forall M, N : (M = N)\phi_A \text{ iff } (M = N)\phi_B$$

and we want to prove that

$$dom(\phi'_A) = dom(\phi'_B) \wedge \forall M, N : (M = N)\phi'_A \text{ iff } (M = N)\phi'_B$$

The domains coincide since Θ preserves the variable names. From the hypothesis we know that

$$\forall M, N : M\phi_A = N\phi_A \text{ iff } M\phi_B = N\phi_B$$

Consider now the frame $\phi'_A = \phi_A\Theta$ and the terms M, N. Since $M\Theta$ and $N\Theta$ are also terms and $\phi_A \approx_s \phi_B$ then by hypothesis it holds that

$$\forall M, N : M\Theta\phi_A = N\Theta\phi_A \text{ iff } M\Theta\phi_B = N\Theta\phi_B$$

But $\Theta\phi_A$ is ϕ'_A and $\Theta\phi_B$ is ϕ'_B therefore we have that

$$\forall M, N : M\phi'_A = N\phi'_A \text{ iff } M\phi'_B = N\phi'_B$$

which corresponds to $\phi'_A \approx_s \phi'_B$.

□

We are now ready to demonstrate that the conditions hold.

SSP: Condition Unbounded number of agents. We have to show that for each n it is possible to create a trace with n transactions run by n different agents. This can simply be done by spawning n copies of the agent process in the system P.

SSP: Condition Renaming. We have to show that $\pi \sim \pi[s'/s]$ for all possible mappings π of traces in T and tags $s \in A_\pi$ and $s' \notin A_\pi$. In the applied pi calculus it is always possible to α-rename a name or a variable in a process without changing its semantics. Consider a trace $\tau \in \mathrm{T}$ where the final process has the following frame:

$$\nu s_1. \ldots . \nu s_i. \ldots . \nu s_{|A_\pi|}. \nu \tilde{w}. \prod_{k=1}^{n} \Phi_{\pi,\sigma}(k)$$

If we α-rename all the occurrences of the secret s_i with $s'_i \notin A_\tau$ in the trace, then we obtain a new frame which is statically equivalent to the original one:

$$\nu s_1. \ldots . \nu s'_i. \ldots . \nu s_{|A_\pi|}. \nu \tilde{w}. \prod_{k=1}^{n} \Phi_{\pi[s'/s],\sigma}(k)$$

SSP: Condition Swapping. We have to prove that, given two equivalent mappings π_A and π_B, for all l such that $\pi_A(l) \neq \pi_A(l+1)$ and $\pi_B(l) \neq \pi_B(l+1)$ also the swapped mappings are equivalent, i.e. $\pi_A^l \sim \pi_B^l$. We only need to show that the traces lead to the same inputs and outputs and the final frames produced by the swapped mappings are statically equivalent. Then, we can use Lemma 1 to conclude the proof. Since the attacker strategy is common to both the swapped traces, and it leads to the same inputs and outputs on the public channel c, the two processes evolve in the same way. The static equivalence to prove is the following:

$$\forall l \in \{1,\ldots,n-1\}, \sigma : \nu\tilde{s}.\nu\tilde{w}.\prod_{k=1}^{n}\Phi_{\pi_A^l,\sigma}(k) \approx_s \nu\tilde{s}.\nu\tilde{w}.\prod_{k=1}^{n}\Phi_{\pi_B^l,\sigma}(k)$$

Consider the frames $\Phi_{\pi_A^l,\sigma}(l)$ and $\Phi_{\pi_A^l,\sigma}(l+1)$. Since single-step protocols have no input and the tags swapped have a different internal secret, if we swap both the mappings and the attacker strategies we obtain two frames that have the same terms assigned to different variables. Such substitution of variable can be obtained by the application of the function Θ that renames the variables $(x_{l,1},\ldots,x_{l,\sigma_l},x_{l+1,1},\ldots,x_{l+1,\sigma_{l+1}})$ into $(x_{l+1,1},\ldots,x_{l+1,\sigma_{l+1}},x_{l,1},\ldots,x_{l,\sigma_l})$. Therefore we obtain that:

$$\Phi_{\pi_A^l,\sigma}(l) = \Phi_{\pi_A,\sigma^l}(l+1)\Theta \tag{6}$$

$$\Phi_{\pi_A^l,\sigma}(l+1) = \Phi_{\pi_A,\sigma^l}(l)\Theta \tag{7}$$

The same reasoning can be applied to the trace τ_B. Given that Θ is a substitution that preserve the variable names and does not affect any term, we can apply it to π_A and π_B obtaining (by Lemma 2):

$$\begin{aligned}\forall l \in &\{1,\ldots,n-1\}, \sigma :\\ &\nu\tilde{s}.\nu\tilde{w}.(\textstyle\prod_{k=1}^{l-1}\Phi_{\pi_A,\sigma^l}(k) \mid \Phi_{\pi_A,\sigma^l}(l) \mid \Phi_{\pi_A,\sigma^l}(l+1) \mid \prod_{k=l+2}^{n}\Phi_{\pi_A,\sigma^l}(k))\Theta \approx_s\\ &\nu\tilde{s}.\nu\tilde{w}.(\textstyle\prod_{k=1}^{l-1}\Phi_{\pi_B,\sigma^l}(k) \mid \Phi_{\pi_B,\sigma^l}(l) \mid \Phi_{\pi_B,\sigma^l}(l+1) \mid \prod_{k=l+2}^{n}\Phi_{\pi_B,\sigma^l}(k))\Theta\end{aligned}$$

Since Θ does not affect the first $l-1$ and the last $n-l+2$ transactions, we have that:

$$\begin{aligned}&\forall l \in \{1,\ldots,n-1\}, \sigma :\\ &\nu\tilde{s}.\nu\tilde{w}.(\textstyle\prod_{k=1}^{l-1}\Phi_{\pi_A,\sigma^l}(k) \mid \Phi_{\pi_A,\sigma^l}(l)\Theta \mid \Phi_{\pi_A,\sigma^l}(l+1)\Theta \mid \prod_{k=l+2}^{n}\Phi_{\pi_A,\sigma^l}(k)) \approx_s\\ &\nu\tilde{s}.\nu\tilde{w}.(\textstyle\prod_{k=1}^{l-1}\Phi_{\pi_B,\sigma^l}(k) \mid \Phi_{\pi_B,\sigma^l}(l)\Theta \mid \Phi_{\pi_B,\sigma^l}(l+1)\Theta \mid \prod_{k=l+2}^{n}\Phi_{\pi_B,\sigma^l}(k))\end{aligned}$$

Given (6) and (7) and the fact that $\forall l \in \{1,\ldots,n-1\}, i \in \{1,\ldots,n\} \setminus \{l,l+1\} : \Phi_{\pi_A,\sigma^l}(i) = \Phi_{\pi_A^l,\sigma}(i) \wedge \Phi_{\pi_B,\sigma^l}(i) = \Phi_{\pi_B^l,\sigma}(i)$ we obtain:

$$\begin{aligned}&\forall l \in \{1,\ldots,n-1\}, \sigma :\\ &\nu\tilde{s}.\nu\tilde{w}.(\textstyle\prod_{k=1}^{l-1}\Phi_{\pi_A^l,\sigma}(k) \mid \Phi_{\pi_A^l,\sigma}(l) \mid \Phi_{\pi_A^l,\sigma}(l+1) \mid \prod_{k=l+2}^{n}\Phi_{\pi_A^l,\sigma}(k)) \approx_s\\ &\nu\tilde{s}.\nu\tilde{w}.(\textstyle\prod_{k=1}^{l-1}\Phi_{\pi_B^l,\sigma}(k) \mid \Phi_{\pi_B^l,\sigma}(l) \mid \Phi_{\pi_B^l,\sigma}(l+1) \mid \prod_{k=l+2}^{n}\Phi_{\pi_B^l,\sigma}(k))\end{aligned}$$

This can be written as:

$$\forall l \in \{1,\ldots,n-1\}, \sigma : \nu\tilde{s}.\nu\tilde{w}.\prod_{k=1}^{n}\Phi_{\pi_A^l,\sigma}(k) \approx_s \nu\tilde{s}.\nu\tilde{w}.\prod_{k=1}^{n}\Phi_{\pi_B^l,\sigma}(k)$$

which is exactly $\pi_A^l \sim \pi_B^l$. Therefore, we can conclude that the condition Swapping holds.

SSP: Condition Extension I. We have to show that two traces (π_A, σ) and (π_B, σ) in T preserve the equivalence under any attacker strategy σ when we add a new transaction at the end of the traces, which is executed by a new tag. Formally, we need to prove that

$$\forall \sigma : \nu\tilde{s}.\nu\tilde{w}.(\prod_{k=1}^{n+1}\Phi_{extn(\pi_A),\sigma}(k)) \approx_s \nu\tilde{s}.\nu\tilde{w}.(\prod_{k=1}^{n+1}\Phi_{extn(\pi_B),\sigma}(k))$$

The left hand side of the equivalence can be written as follows:

$$\nu\tilde{s}.\nu\tilde{w}.(\prod_{k=1}^{n}\Phi_{\pi_A,\sigma}(k) \mid \Phi_{extn(\pi_A),\sigma}(n+1))$$

since the first n mappings corresponds to the original ones.
We know that the last transaction, by hypothesis, belongs to a fresh new tag never appeared in the first n transactions. This means that its secret and internal memory restricted by $\nu\tilde{s}$ and $\nu\tilde{w}$ do not belong to $\prod_{k=1}^{n}\Phi_{\pi_A,\sigma}(k)$, which contains restrictions that do not belong to $\Phi_{extn(\pi_A),\sigma}(n+1)$. Therefore, we can separate the restrictions obtaining an equivalent frame:

$$\nu\tilde{s}.\nu\tilde{w}.\prod_{k=1}^{n}\Phi_{\pi_A,\sigma}(k) \mid \nu s.\nu w.\Phi_{extn(\pi_A),\sigma}(n+1)$$

The same can be done for the right hand side of the equivalence.
We know by hypothesis that:

$$\forall \sigma : \nu\tilde{s}.\nu\tilde{w}.(\prod_{k=1}^{n}\Phi_{\pi_A,\sigma}(k)) \approx_s \nu\tilde{s}.\nu\tilde{w}.(\prod_{k=1}^{n}\Phi_{\pi_B,\sigma}(k))$$

Moreover the frames $\nu s.\nu w.\Phi_{extn(\pi_A),\sigma}(n+1)$ and $\nu s.\nu w.\Phi_{extn(\pi_B),\sigma}(n+1)$ are identical under any attacker strategy σ. Therefore, also the mappings $extn(\pi_A)$ and $extn(\pi_B)$, obtained by parallel composition of these frames, are equivalent.

SSP: Condition Extension II. As in the previous proof, we have to show that if we add a new transaction to two equivalent traces (π_A, σ) and (π_B, σ) in T, then their equivalence is preserved under any attacker strategy σ. The difference is that now the last transaction does not belong to a new tag but to the tag which executed the last transaction in the original traces. Formally, we want to prove that if $\pi_A \sim \pi_B$ then also $extl(\pi_A) \sim extl(\pi_B)$.
Consider any pair of mappings $extl(\pi_A)$ and $extl(\pi_B)$ built up from the two equivalent mappings π_A and π_B. For any possible attacker strategy σ applied to the mappings $extl(\pi_A)$ and $extl(\pi_B)$ there always exists an attacker strategy σ' such that:

- $|\sigma'| = |\sigma| - 1$
- $\sigma'_i = \begin{cases} \sigma_i & i < |\sigma'| \\ \sigma_n + \sigma_{n+1} & i = |\sigma'| \end{cases}$

such that $(\pi_A, \sigma') \sim (\pi_B, \sigma')$ by hypothesis. Since (π_A, σ') and (π_B, σ') produce the same frames as $(extl(\pi_A), \sigma)$ and $(extl(\pi_B), \sigma)$ respectively, then it must be that also $(extl(\pi_A), \sigma) \sim (extl(\pi_B), \sigma)$ for any attacker strategy σ. □

We can conclude that all the forms of unlinkability and inseparability defined in Section 4 coincide for the class of single-step protocols. As a consequence, if any of these properties is proven to hold for a single-step protocol, by Theorem 7 all the unlinkability and inseparability properties should be satisfied.

7 Relations with forward and backward privacy

In this section we discuss how our results on unlinkability can easily be adapted to define and prove stronger privacy guarantees. We show that, assuming a stronger attacker and restricting the analysis to specific parts of traces in the set generated by a protocol, it is possible to verify the stronger goals of forward and backward privacy.

Forward and backward privacy are privacy definitions strongly related to the concept of unlinkability. In cryptography, they are well-known under the names of forward and backward security, respectively.

Both properties assume a stronger attacker with respect to the definition of unlinkability, namely an attacker who is able to disclose all the secret information of an agent at a chosen time. Forward privacy is achieved when the disclosure of an agent secret does not help the attacker in linking the agent to any of its past sessions. Symmetrically, backward privacy states that an attacker should not be able to link the agent secret to any future session. Clearly, with the coming of mobile technology, these properties became fundamental for the location privacy of agents. In fact, a protocol that guarantees forward or backward privacy prevents an attacker from tracing back or forward all the steps of an agent. A first definition of forward privacy in the context of RFID systems can be found in [22], while [14] first modelled backward privacy. Both papers give definitions in terms of games, which consist of a challenge to the attacker who has to guess if there exists a session (during the execution of the system) that belongs to the RFID tag whose secret she disclosed.

Forward and backward privacy can be formalized in terms of traces in our model. Similarly to unlinkability, backward and forward privacy are guaranteed if the attacker cannot tell whether a specific transaction in a trace is linked. On the other hand, they involve a notion of time and require stronger assumptions. First of all, we need to assume that the attacker strategy for a certain transaction discloses the secret of the agent executing it, and we call that transaction p_d (i.e. the d-th transaction in a trace). Since the forward privacy analysis concerns only *past* transactions, we check whether p_d can be linked to transactions executed before it only. We express this property in two different forms, namely a weak and a strong form inspired by the corresponding definitions of unlinkability. Intuitively, weak forward privacy requires that the attacker cannot link p_d to a specific past transaction, while strong forward privacy holds when the attacker cannot tell whether p_d is linked to *any* previous transaction, meaning that she cannot tell whether the agent who runs p_d has ever executed a session in the past.

Definition 12 (Forward privacy)**.** Consider a set of traces T. For each trace $\tau \in \mathrm{T}$, let p_d be a transaction such that its attacker strategy $\sigma(p_d)$ discloses $\pi(p_d)$ secret information.
We say that a protocol generating the set of traces T guarantees weak forward privacy iff

$$\forall \tau \in \mathrm{T}, p_d \in Dom_\tau, i < d : \tau \vDash \neg K_a(link(p_d, p_i))$$

We say that a protocol generating the set of traces T guarantees strong forward privacy iff

$$\forall \tau \in \mathrm{T}, p_d \in Dom_\tau : \tau \vDash \neg K_a \bigvee_{\substack{p_i \in Dom_\tau \\ i<d}} link(p_d, p_i)$$

Intuitively, these definitions imply that, if the agent whose secret has been disclosed executed transactions before p_d, there must exist a trace τ' in T such that $\tau' \sim \tau$ and the corresponding transactions (or any previous transaction for strong forward privacy) are not linked to p_d. It is easy to prove that strong forward privacy implies weak forward privacy; it suffices to follow the technique used in last three steps of the proof of Theorem 2.

Example 7. A well-known protocol that guarantees forward privacy is the OSK protocol [22] described in Section 4.6. To prevent the attacker from linking a secret to sessions executed before its disclosure, the protocol updates the secret after each execution. To prove that this protocol guarantees strong forward privacy in our model, we should show that each linked trace is equivalent to a trace where the transactions involved are unlinked, as described in Definition 12. Consider, for example, a trace $\tau = (\pi, \sigma)$ produced by the OSK protocol such that $\pi = (a_1, a_1, a_2)$. Let $d = 2$, meaning that the attacker strategy causes the secret disclosure at the second transaction, so that the attacker knows that it was executed by the agent a_1. In practice, the attacker cannot link the secret to the first transaction, because it is not possible to calculate the inverse function h^{-1} of a_1's current secret. On the other hand, once the attacker obtains the secret, she is able to trace it in the future, i.e. she knows that the third transaction is executed by a different agent, thus the protocol does not guarantee backward privacy. In our model, all these results imply that τ must be equivalent to a trace such that the second transaction is not linked to the first transaction nor to the third one (the attacker can distinguish the second transaction from the third one). Formally, we have that $\tau \sim \tau_1 = ((a_1, a_2, a_1), \sigma)$ or $\tau \sim \tau_2 = ((a_1, a_2, a_3), \sigma)$. This suffices to have both forms of forward privacy. Note that, if h^{-1} exists then τ would only be equivalent to itself, and the protocol would not guarantee forward privacy.

Symmetrically, backward privacy analysis checks whether the secret information is linked to *future* transactions. As in all previous work on this property (e.g. [14, 12]), we need to assume that, subsequently to p_d, a synchronization session in a transaction p_s ($s > d$) takes place. A synchronization session is a session executed among honest parties only, meaning that the attacker is not supposed to eavesdrop it. The synchronization session is needed to avoid the traceability of the agent, whose secret knowledge is now shared with the attacker. The session ignored by the attacker is needed to give the agent an opportunity to secretly update its internal state. In our model, after the disclosing session p_d, the attacker is able to link the transactions belonging to the agent $\pi(p_d)$. We assume that she models her strategy to ignore a legitimate session that belongs to $\pi(p_d)$ during the s-th transaction. Thus, backward privacy holds if the attacker cannot tell if p_d is linked to a specific transaction (weak form) or to any other transaction (strong form) executed after p_s.

Definition 13 (Backward privacy)**.** Consider a set of traces T. For each trace $\tau \in \mathrm{T}$, let p_d be the transaction such that its attacker strategy $\sigma(p_d)$ discloses $\pi(p_d)$ secret information, and let p_s be the transaction such that $s \in [d+1, \ldots, |\tau|]$, $\pi(p_s) = \pi(p_d)$ and it executes a synchronization session.
We say that a protocol generating the set of traces T guarantees weak backward privacy iff

$$\forall \tau \in \mathrm{T}, p_d \in Dom_\tau, s > d, i > s : \tau \vDash \neg K_a(link(p_d, p_i))$$

We say that a protocol generating the set of traces T guarantees strong backward privacy iff

$$\forall \tau \in \mathrm{T}, p_d \in Dom_\tau, s > d : \tau \vDash \neg K_a \bigvee_{\substack{p_i \in Dom_\tau \\ i>s}} link(p_d, p_i)$$

If some transactions are executed by the agent $\pi(p_d)$ after the restore session in p_s in a trace $\tau \in \mathrm{T}$, there must exist a trace $\tau' \in \mathrm{T}$ such that $\tau' \sim \tau$ and the corresponding transactions (or any future transaction for strong backward privacy) are executed by other agents. It is easy to prove that strong backward privacy implies weak backward privacy; it suffices to follow the technique used in last three steps of the proof of Theorem 2.

Example 8. Consider a variation of the OSK protocol that sends a random number in plain text together with the hash of the secret, and then updates the secret by hashing it with the random number. This protocol prevents the attacker from linking a secret to transactions executed after the restore session, because the attacker, by missing a random number, would not be able to calculate the next secret. To prove that this protocol guarantees strong backward privacy in our model, we should show the equivalence of traces as described in Definition 13. Consider, for example, a trace $\tau = (\pi, \sigma)$ produced by this protocol such that $\pi = (a_1, a_2, a_1, a_1)$. Let $d = 1$, meaning that the attacker strategy causes the secret disclosure at the first transaction, and $s = 3$, namely the restore session happens at the third transaction. The attacker should not be able to link the secret to the fourth transaction, because it is not possible to calculate the new secret without the random number sent during the third transaction. In our model, this means that τ must be equivalent to a trace such that the third transaction is not linked to the fourth transaction. Thus, it must be that $\tau \sim \tau_1 = ((a_1, a_2, a_1, a_2), \sigma)$ or $\tau \sim \tau_2 = ((a_1, a_2, a_1, a_3), \sigma)$. This suffices to have both forms of backward privacy.

These results complete our study of unlinkability, providing a full description of the conditions that a protocol should guarantee in order to avoid any information disclosure, even when a stronger attacker is assumed.

8 Related work

Our work makes direct use of several definitions of unlinkability from the literature. As explained in detail in Section 4, we express the notion of weak unlinkability of [27, 3], strong unlinkability of [2, 3], and game-based definitions of [10, 18, 22, 4, 23, 21]. While all these works have given their own definitions of privacy properties in a very specific context, ours provides a more general and abstract framework where all the other definitions can be captured.

Epistemic models have been used in the past to formalize privacy. Similarly to our work, [15] gives general privacy definitions for a multiagent system using a modal logic of knowledge. The paper considers different levels of strength for unlinkability, providing some probabilistic definition as well. In [9], epistemic logic is used to give intuitive definitions of privacy in voting systems, with the applied pi calculus as the underlying model. Similarly, [11] proposes a framework in which protocols are expressed in a process language while security properties in a logic with both temporal and epistemic operators. The properties considered in the above works are quite different than the unlinkability properties that we consider in this paper. Moreover, the

above works are involved with the mechanics of the corresponding formalisms, while we try to completely abstract away from the concrete model, viewing a system as an abstract set of traces.

Several other definitions of unlinkability have also been studied in the literature. A logic approach has been followed in [26], where an axiom system is defined to reason about anonymity, and in [16] that expresses privacy properties using logic and models the system through other formalisms, like CSP, combining two different techniques. As in our work, logic is used to define in a natural way privacy properties, while having an abstract model applicable to any real system. However, while our work focuses on unlinkability, [15] and [26] study anonymity, namely a property that ensures that the identity of the agent which executes some action remains hidden from other observers. [24] proposes a terminology for anonymity, unlinkability, unobservability and pseudonimity. While it aims at clarifying terminology at an informal level, our work aims instead at comparing definitions of unlinkability in a unifying formal model.

Finally, other papers introduce the notion of unlinkability using approaches based on information theory. Examples are [13], [25], and [6] that give probabilistic descriptions of unlinkability, quantifying the linkability of items in the system. Our work does not provide any probabilistic definition, but this would be possible following the approach used in [15], that we leave as future work.

9 Conclusions and future work

In this paper we studied the privacy notion of unlinkability. We captured several definitions from the literature into a simple abstract model based on epistemic logic, obtaining natural and intuitive definitions in terms of the attacker's knowledge. We also identified inseparability, a notion dual to unlinkability, in weak and strong forms. Moreover, we showed that these privacy definitions are different in general, but do coincide in systems satisfying a set of conditions. Finally, we proved that the conditions hold for a class of identification protocols.

As future work, we plan at investigating probabilistic descriptions of unlinkability. We also plan at developing a more concrete model in the style of [11]. This work bridges the gap between operational semantics and epistemic logic, offering a combined framework where it is possible to easily model the behavior of a protocol in a process language with an operational semantics and reason about properties expressed in a rich epistemic temporal logic. This would allow to automatically verify privacy properties.

10 Tables and figures

Table 1: Epistemic definitions of unlinkability and inseparability

	Property	*Epistemic formula*	*For any trace τ the attacker cannot infer*
a.	*Weak unlinkability*	$\neg K(link(p, p'))$	that two messages p, p' are linked
b.	*Strong unlinkability*	$\neg K(anyLink(\tau))$	the existence of linked messages
c.	*Two-agents game unlinkability*	$[\pi_a \vee \pi_{a'}]\neg K\pi_a$	the mapping π_a even when $\pi_a \vee \pi_{a'}$ is revealed
d.	*Three-agents game unlinkability*	$[\pi_{a,a} \vee \pi_{a_1,a_2}]\neg K\pi_{a,a}$	the mapping $\pi_{a,a}$ even when $\pi_{a,a} \vee \pi_{a_1,a_2}$ is revealed
e.	*Weak inseparability*	$\neg K(unlink(p, p'))$	that two messages p, p' are unlinked
f.	*Strong inseparability*	$\neg K(anyUnlink(\tau))$	the existence of unlinked messages

Table 2: Negative relationship between properties

Relationship			*References*
Weak unlinkability	$\not\Rightarrow$	Strong unlinkability	Examples 2,4,5,6
Weak unlinkability	$\not\Rightarrow$	Game-based unlinkability	Examples 3,4,5,6
Strong unlinkability	$\not\Leftarrow\not\Rightarrow$	Game-based unlinkability	Examples 2,3
Weak inseparability	$\not\Rightarrow$	Strong inseparability	Example 6

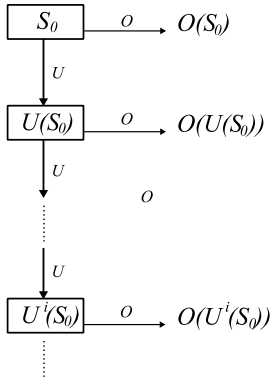

Figure 1: A single-step protocol.

Bibliography

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. of POPL*, pages 104–115, 2001.

[2] M. Arapinis, T. Chothia, E. Ritter, and M. Ryan. Untraceability in the applied pi-calculus. In *Proc. of ICITST*, pages 1–6. IEEE, 2009.

[3] M. Arapinis, T. Chothia, E. Ritter, and M. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proc. of CSF*, pages 107–121. IEEE Computer Society, 2010.

[4] G. Avoine. Adversary model for radio frequency identification. Technical Report LASEC-REPORT-2005-001, Swiss Federal Institute of Technology, Lausanne, Switzerland, 2005.

[5] G. Avoine. *Cryptography in Radio Frequency Identification and Fair Exchange Protocols*. PhD thesis, EPFL, Lausanne, Switzerland, 2005.

[6] R. Berman, A. Fiat, and A. Ta-Shma. Provable unlinkability against traffic analysis. In *Proc. of Financial Cryptography*, LNCS, pages 266–280. Springer, 2004.

[7] M. Brusó, K. Chatzikokolakis, and J. den Hartog. Formal verification of privacy for rfid systems. In *Proc. of CSF*, pages 75–88. IEEE Computer Society, 2010.

[8] M. Burmester, T. V. Le, and B. de Medeiros. Provably secure ubiquitous systems: Universally composable rfid authentication protocols. In *Proc. of Securecomm*, pages 1–9, 2006.

[9] R. Chadha, S. Delaune, and S. Kremer. Epistemic logic for the applied pi calculus. In *Proceedings of IFIP*, volume 5522 of *LNCS*, pages 182–197, Lisbon, Portugal, 2009. Springer.

[10] C. Chatmon, T. van Le, and M. Burmester. Secure anonymous RFID authentication protocols. Technical Report TR-060112, Florida State University, Tallahassee, Florida, USA, 2006.

[11] F. Dechesne, M. R. Mousavi, and S. Orzan. Operational and epistemic approaches to protocol analysis: Bridging the gap. In *Proc. of LPAR*, volume 4790 of *LNCS*, pages 226–241, 2007.

[12] T. Dimitriou. Rfid-dot: Rfid delegation and ownership transfer made simple. In *Proc. of 4th International Conference on Security and Privacy in Communication Networks*, pages 1–8. ACM, 2008.

[13] M. Franz, B. Meyer, and A. Pashalidis. Attacking unlinkability: The importance of context. In *Proc. of Privacy Enhancing Technologies*, volume 4776 of *LNCS*, pages 1–16. Springer, 2007.

[14] F. D. Garcia and P. van Rossum. Modeling privacy for off-line rfid systems. In *Proc. of CARDIS*, volume 6035 of *LNCS*, pages 194–208. Springer, 2010.

[15] J. Y. Halpern and K. R. O'Neill. Anonymity and information hiding in multiagent systems. In *Proc. of CSFW*, pages 75–88. IEEE Computer Society, 2003.

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

[17] A. Juels, P. F. Syverson, and D. V. Bailey. High-power proxies for enhancing rfid privacy and utility. In *Proc. of PET*, volume 3856 of *LNCS*, pages 210–226. Springer, 2005.

[18] A. Juels and S. A. Weis. Defining strong privacy for rfid. In *Proc. of PerCom Workshops*, pages 342–347. IEEE Computer Society, 2007.

[19] J.-J. C. Meyer and W. v. d. Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press, New York, NY, USA, 2004.

[20] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts i and ii. *I and II. Information and Computation*, 100:1–77, 1989.

[21] K. Nohl and D. Evans. Privacy through noise: a design space for private identification. In *Annual Computer Security Applications Conference (ACSAC 2009)*, 2009.

[22] M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *Proc. of RFID Privacy Workshop*, 2003.

[23] K. Ouafi and R. C.-W. Phan. Privacy of recent rfid authentication protocols. In *Proc. of ISPEC*, volume 4991 of *LNCS*, pages 263–277. Springer, 2008.

[24] A. Pfitzmann and M. Kohntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. *Designing Privacy Enhancing Technologies*, page 19, 2001.

[25] S. Steinbrecher and S. Kopsell. Modelling unlinkability. In *Proc. of Privacy Enhancing Technologies*, volume 2760 of *LNCS*, pages 32–47. Springer, 2003.

[26] P. F. Syverson and S. G. Stubblebine. Group principals and the formalization of anonymity. In *Proc. of World Congress on Formal Methods*, volume 1708 of *LNCS*, pages 814–833, 1999.

[27] T. van Deursen, S. Mauw, and S. Radomirovic. Untraceability of rfid protocols. In *Proc. of WISTP*, volume 5019 of *LNCS*, pages 1–15. Springer, 2008.

[28] T. van Deursen and S. Radomirovic. Algebraic attacks on rfid protocols. In *Proc. of WISTP*, volume 5746 of *LNCS*, pages 38–51. Springer, 2009.

[29] S. Vaudenay. On privacy models for rfid. In *Proc. of ASIACRYPT*, volume 4833 of *LNCS*, pages 68–87. Springer, 2007.

[30] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels. Security and privacy aspects of low-cost RFID systems. In *Proc. of SPC*, volume 2802 of *LNCS*, pages 201–212. Springer, 2003.

MEALS Partner Abbreviations

SAU: Saarland University, D

RWT: RWTH Aachen University, D

TUD: Technische Universität Dresden, D

INR: Institut National de Recherche en Informatique et en Automatique, FR

IMP: Imperial College of Science, Technology and Medicine, UK

ULEIC: University of Leicester, UK

TUE: Technische Universiteit Eindhoven, NL

UNC: Universidad Nacional de Córdoba, AR

UBA: Universidad de Buenos Aires, AR

UNR: Universidad Nacional de Río Cuarto, AR

ITBA: Instituto Técnológico Buenos Aires, AR