Dissecting unlinkability

Dissecting Unlinkability

Mayla Brusó

A catalogue record is available from the Eindhoven University of Technology Library.
ISBN: 978-90-386-3649-8

This work is supported by the research program Sentinels (www.sentinels.nl) as project PEARL (07639).

Dissecting Unlinkability

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, Prof. Dr. Ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties, in het openbaar te verdedigen
op woensdag 18 juni 2014 om 16:00 uur

door

Mayla Brusó

geboren te Venetië, Italië

Dit proefschrift is goedgekeurd door de promotor:

Prof. Dr. S. Etalle

Copromotor:
Dr. J. I. den Hartog

ACKNOWLEDGEMENTS

I would like to thank everyone who has supported me during my Ph.D. studies. First, I want to thank my supervisor Sandro Etalle for giving me the opportunity to pursue a Ph.D. at TU/e, for his help and for all the valuable comments and suggestions on my papers and on this dissertation. I thank my daily supervisors Jerry den Hartog and Kostas Chatzikokolakis for their guidance and support. And thank you all for your friendship and good advice.

I want to express my gratitude to the members of my doctoral committee, Emile Aarts, Sjouke Mauw, Milan Petković, Erik Poll and Berry Schoenmakers, for assessing my thesis and providing valuable feedback.

I had a great time in Eindhoven thanks to all the friends I made in the past years. In particular I want to thank Alessandro, Antonio A, Antonio C, Alberto, Bruno, Christiane, Çiçek, Costas, Daniela, Dave, Daniel, Elisa, Eric, Fred, Gaetan, Giovanni, Giulia, Henk, Irene, Jan-Jaap, Jing, Jolande, Juan Carlos, Lida, Mark, Nicola, Patrizia, Richard, Sabine, Shona, Tanir, Vinicius. Thanks a lot for all the fun, the coffee breaks, the dinners and the chats! Thanks also to the other members of the SEC, CC, CASA and DAM groups: Anders, Anita, Benne, Boris, Dan, Dion, Fatih, Jan-Willem, José, Michael, Maxim, Meilof, Patricio, Patrick, Peter B, Peter S, Peter vL, Relinde, Rob, Ruben, Ruud, Sebastiaan, Sokratis, Tanja, Tanya, Thijs, Wil, Yael. You all made TU/e such a *gezellig* working place.

I thank all my new colleagues at Riscure for giving me the opportunity to work on new challenging projects in such a pleasant environment.

Un enorme grazie a Luca per essermi sempre stato vicino nonostante la distanza che ci separa. Grazie per tutti i bei momenti, i discorsi profondi e quelli insensati che hanno reso tanto speciale la nostra amicizia!

Vorrei ringraziare tutta la mia famiglia per il costante supporto. In particolare, grazie mamma e papá per tutto il vostro amore e il vostro appoggio, senza i quali non avrei mai potuto raggiungere questo traguardo.

Infine, il ringraziamento piú grande va a Damiano (aka Antonino o Jaba). Non ci sono parole per descrivere quanto tu abbia straordinariamente migliorato la mia vita ogni giorno durante tutti questi anni.

CONTENTS

1

INTRODUCTION

Radio-frequency identification (RFID) systems are a wireless technology for automatic identification of objects or persons. They consist of a set of tags, a set of readers and a backend. Tags are typically simple battery-less devices consisting of a tiny chip and an antenna. The antenna is needed to emit radio signals while the chip is used both for computation and storing data. This data always includes a unique identification number. Readers are devices that, on the one hand, can wirelessly connect to tags using radio waves and, on the other hand, typically communicates to a backend system through a wired network. The backend system contains all the information needed to identify the tags in the system. Basically, each reader serves as an intermediary between the tags to be identified and a backend. The sequence of messages that are exchanged between these agents is called an *identification protocol*. At the end of a protocol execution, the backend system should have correctly identified the tag running the protocol through a reader, and therefore the person or the object that carries the tag.

RFID technology is already employed in a wide range of applications. The main advantage of RFID systems, compared to other identification systems, is that tags can do computation and a line of sight is not required to identify items. The possibility of performing computation is vital to guarantee privacy while the fact that no line of sight is needed implies that the reader does not even need to "see" a tag to identify it, but only needs it to be in its proximity. For example, if a book has a bar code on its inside back cover, one needs to find it and place it in front of a bar code reader in order to retrieve the corresponding information. If a book contains instead an RFID tag, one only needs to put it close to a reader, which can read the tag information through the materials of the book. The fact that there is no need

of a line of sight also implies that many items can be identified at the same time. Thanks to these advantages, the adoption of RFID technology allows to speed up many existing applications and enables new ones. For example, they are already used in libraries to manage the inventory, in supply chain management for process automation, in passports for a better border control, in cars to automatically pay road tolls without stopping, in badges to let only authorised people access office buildings, in pets to identify their owner, or even in the body of human beings to retrieve their medical history. All this information used to be manually collected by means of slow and error-prone processes, while the data collection in RFID systems is fully automated and precise, requiring basically no human intervention.

As a new application, we expect RFID tags to replace the bar code labels in a wide range of products in the near future, when their price will drop sufficiently to be suitable also for inexpensive products. In this case, tags may be employed by different users for different purposes. A manufacturer or a retailer may use tags for managing their inventory and to retrieve asset information, and a customer for payment and further future services. For instance, consider a food product carrying an RFID tag whose unique identification number is linked to data such as its manufacturer information, its production and expiry dates, its country of origin, and so on. This information may be used by its manufacturer and its retailer to instantly know the amount of each product and to control that products meet the requirements to be sold, resulting in an enhancement of the supply chain efficiency. Then, customers may verify that products satisfy their needs (for instance, one may check whether the food product is both vegan and gluten-free), and would no longer have to queue at the checkout counter to have their products scanned one by one. In fact, a person would only need to walk in proximity of a reader with his shopping trolley, and all the items and corresponding prices would be immediately detected. Also, the fridge where the customer stores his food may be equipped with an RFID reader that displays which products are about to expire and must be used as soon as possible.

Finally, there is a range of RFID applications that are not widely employed yet, although the technology to implement them is already available. We already mentioned that RFID tags are still too expensive to be used as a replacement for all bar codes, but they are certainly not too expensive to be used in high-priced products. Nevertheless, this and similar solutions are not widely adopted yet. This is because the wireless nature of RFID systems, while being their main selling point, is also one of their main vulnerabilities. Tags are extremely easy to access, also for unauthorised users trying to obtain

information about the product a tag is attached to or about the person who is carrying it. For example, a person who buys an expensive coat may forget to remove the RFID tag and be traceable due to some tag vulnerability. Also, it has been proposed to use RFID tags in banknotes to avoid their counterfeit and to monitor illegal activities. In this scenario, a person with a reader may be able to instantly know whether someone is carrying banknotes or to trace banknotes, e.g. to link a person to his purchases. These unacceptable privacy violations are a main hindrance to further RFID adoption. Therefore it is necessary to find solutions to avoid such information disclosure. In the following sections we discuss these privacy issues in more details and our approach to solving them.

1.1 Privacy in RFID systems

As it appears in the previous section, RFID systems can raise a number of privacy issues. An RFID protocol may leak too much data allowing tracing of tags, which results in disclosing private information about the person or object carrying them. For instance, if a person carries a traceable RFID tag (for example sewed to the person's coat), then one may learn information such as home address, work address, habits, etc., just by following that specific tag.

In order to avoid such privacy violations, it is essential to clarify which are the privacy guarantees that an identification protocol should satisfy. Formalising the concept of privacy in the context of RFID systems is not trivial though. Privacy definitions typically require that a protocol does not leak a specific piece of information. However, privacy in RFID systems is only guaranteed when a protocol does not leak *any information* that may help to distinguish a tag from another, thus trace a specific tag. This intuition has been formalised in different models [38, 3, 18, 27, 4, 32], leading to several definitions whose relationship is unclear. Although such definitions claim to guarantee the same privacy property, in practice they seem to provide different privacy guarantees, but it is difficult to compare them because they are modelled in different settings. Thus, the complete understanding of privacy in this context remains an open problem.

To prevent privacy violations, protocols must adopt some privacy preserving solutions. Unfortunately, the standard techniques used in identification systems are unsuitable for RFID systems, due to their uncommon features, such as their mobile nature and the resource limitations of RFID tag chips. A tag chip offers a little amount of data storage and a small set of basic opera-

tions. This means that a tag cannot store much more than its unique identifier and it is not powerful enough to execute all the steps of a standard privacy protocol. Therefore, privacy can only be achieved in RFID systems by developing ad hoc protocols that meet all the RFID constraints. However, several protocols proposed in the past years have been proven to fall short of their privacy goals, sometimes after their implementation in real systems, causing consumers to lose trust in this technology. The shortage of formal solutions to validate RFID protocols is thus another open problem to investigate.

In the next section we introduce a range of privacy notions and discuss the problems deriving from their violation in more detail.

1.2 Unlinkability and related notions

Sensitive data is typically not stored in RFID tags, but rather in the backend system. The information that allows a backend system to associate such data to a specific RFID tag is the tag unique identifier, which often is the most valuable information stored in a tag. Clearly, an identification protocol must avoid the public disclosure of such information, otherwise an attacker, namely a user who wants to illegitimately access user data, may obtain private information about the person or object carrying an RFID tag, as explained in the previous section. More formally, a protocol must first of all prevent an attacker from being able to infer whether two or more protocol executions belong to the same tag. This cornerstone property is known as *unlinkability*, and, in the context of RFID systems, it is the most important privacy notion. It can also be seen as a special form of anonymity. Anonymity is a more general privacy property that holds when a protocol prevents an attacker from being able to uniquely identify a user. A user, in the case of RFID systems, corresponds to a tag. Instead, unlinkability is violated when an attacker can distinguish between tags. Therefore, a protocol may not guarantee this property even if anonymity holds, i.e. when the protocol does not leak tag identifiers. For example, a protocol that discloses the session number of an RFID tag is already an issue, because an attacker can trace a tag by querying the tags in the range of its reader and following the one that emits the expected sequential session number.

Sometimes, RFID systems must satisfy privacy properties that are stronger than unlinkability, because user privacy must be guaranteed even when an attacker is able to tamper with tags. So far, the attacker we mentioned could only eavesdrop on the wireless channel between legitimate tags and readers,

and query tags with his own reader, but he could not physically obtain a tag. Unfortunately, in many practical situations an attacker can easily get hold of a tag in most RFID systems. Sometimes the tag is simply transferred, e.g. when the attacker buys a second-hand product. Otherwise, an attacker may illegitimately get temporary access to a tag. Suppose that a person loses his wallet, which contains a card with an RFID installed in it. An attacker finds the wallet and tampers with the RFID tag, obtains its unique identifier, and then gives the wallet to the lost-and-found office. By using the tag unique identifier, it may be possible to retroactively trace the RFID tag or to trace its location in the future. In other words, one may learn where the owner of the wallet has been and may trace him once he has been given back his wallet. To capture this, the attacker must be provided with stronger capabilities, i.e. he can obtain a tag internal state, and see whether he can link it to other legitimate sessions. A protocol that prevents the attacks described above guarantees the privacy properties of forward and backward privacy. Intuitively, forward privacy holds if an attacker cannot infer whether some past protocol executions belong to a tag whose identifier has been disclosed. Symmetrically, backward privacy holds if privacy is protected even after the disclosure of a tag identifier.

The privacy concerns raised by the use of the RFID technology are clear, but there is no agreement in the literature on the concept of unlinkability. A variety of existing definitions differ in model and strength [38, 3, 18, 27, 4, 32], and are apparently unrelated to each other. Thus, when a protocol is proven to guarantee some definition of unlinkability, it may actually not satisfy some other definition. As a result, different protocols that in principle should guarantee the same property provide, in practice, different privacy guarantees. Therefore, a first open problem is the lack of clarity on the exact definition of unlinkability. With a full understanding of the notion of unlinkability, we can provide the formal basis to express privacy properties and verify protocols.

Developing protocols that satisfy all these privacy properties is not an easy task and even the smallest mistake in the protocol design may lead to severe consequences. Consider the case of e-passports, passports containing an RFID tag that stores sensitive information which is claimed to be protected from unauthorised access. However, some protocol vulnerabilities that lead to the violation of unlinkability have been found in [20]. Many other identification protocols have been proven not to guarantee privacy, and every new protocol may potentially have design flaws. Hence, a second open problem is the lack of tools to verify whether identification protocols guarantee what

they promise. Therefore, after understanding which are exactly the privacy guarantees that RFID systems should provide, we must find a way to formally define and verify privacy properties taking into account all the unique features of RFID systems, otherwise a range of promising new RFID applications will never be employed.

1.2.1 Unlinkability in other contexts

Unlinkability is not a new concept to the security world. In fact, already in 1985, [19] showed the importance of unlinkability in credential systems. The problem was that organisations could exchange information about individuals, and instantly link their records to those held by other organisations, leading to severe privacy violations.

There are many other applications where unlinkability is desirable. For example, in e-voting systems, there are typically two phases that should never be linked: during the initial phase, an election authority certifies that a ballot is valid (the voter can vote and did not already vote), and, in a later phase, the votes are sent. If unlinkability does not hold for these two phases of the voting protocol, a voter's identity and vote can be linked, resulting in a violation of the voter's privacy. Unlinkability may be desirable also in web applications to avoid that they use the information stored on computers to link current activities of a user with previous actions.

Given the importance of unlinkability in such a variety of contexts, we keep our definitions in Chapter 3 general. Although we have been inspired by definitions in the RFID literature, our results may be also used to study and compare definitions given in a different context. The reason why we focused on RFID systems (and designed an RFID model in Chapter 4) is that, due to their mobile nature, protecting privacy is of paramount importance in this setting.

1.3 Research question

Given the characteristics of RFID systems and the related challenges listed in the previous sections, the aim of this thesis is:

To provide a formal framework in which privacy properties can be specified and verified

To achieve this goal we have to answer two research questions. As explained in Section 1.2, there is no agreement on the exact definition of privacy notions in the context of RFID systems. Therefore, our first objective is to understand the intuition behind all these definitions and compare them to capture their differences. Thus, our first research question is:

> *How can we define a unifying framework to model and compare different privacy definitions?*

The answer to our first research question helps clarify the notion of privacy. Then, our second objective is to provide a model tailored to RFID systems where we can define and verify, in particular, the strongest of the forms of privacy that we model in our unifying framework. Hence, our second research question is:

> *How can we implement a model in which we can verify whether a protocol guarantees unlinkability and its related notions?*

In the following chapters, we investigate these topics by modelling privacy first in an abstract and then in a concrete model. The abstract model allows us to reason about the privacy guarantees behind existing definitions. On the other hand, the concrete model allows to describe a protocol and to formally verify it. Also, we show how our solutions can be applied to real protocols.

1.3.1 Contributions

In order to answer the research questions presented in Section 1.3, we propose formal techniques to define and study a range of privacy properties. Our work gives rise to the following contributions:

- A general model for the *specification* and *comparison* of privacy properties [13, 12] that helps us capture the intuition behind existing definitions of unlinkability.
- A model tailored to RFID systems that allows us to *describe identification protocols* and use formal techniques to define and *verify* privacy properties [11, 10].

The general nature of the model that we present in Chapter 3 enables easy specification of privacy properties, abstracting away from non-relevant technical aspects of more concrete systems. This model captures existing definitions from very different settings in a single unifying model, so their privacy

guarantees can be compared. This work allows us to answer the first research question, since it results in a better understanding of the notions of unlinkability, forward and backward privacy. We show that the existing definitions are different while they claim to express the same properties. Therefore we study necessary and sufficient conditions that hold under most realistic situations and under which the different definitions of the properties do coincide. Finally, in Chapter 5, we show that a large class of protocols satisfies these conditions.

In Chapter 4, we present a concrete model tailored to RFID systems. Our study on unlinkability, forward and backward privacy described above allows us to identify the privacy guarantees that are desirable for an identification protocol. Based on the strongest of these privacy guarantees, we define the properties in our concrete model taking into account specific RFID features. The more concrete model allows us to analyse protocols and verify whether they satisfy unlinkability, forward and backward privacy, as required by the second research question. Because these definitions are not trivial to prove, we express equivalent and simpler conditions for a class of basic protocols introduced in Chapter 5. Also, we study some protocols from the literature checking whether these conditions hold, providing proofs if they do.

In general, this thesis contributes to the field of formal verification. It provides techniques and tools to analyse identification protocols, with particular focus on RFID systems. The formalisms that we use to express protocols and define properties are very intuitive and precise, as our work is in a symbolic setting. We can directly apply our results to existing protocols or even to general cases such as the class of protocols that we study in Chapter 5. Finally, our definitions can be easily expressed in existing tools like ProVerif [8] in order to automatically verify certain protocols, as we do for our case studies.

1.4 Plan of the thesis

The remainder of this thesis is structured as follows.

Chapter 2 presents an overview of the epistemic logic and the applied pi calculus, the ingredients used to model unlinkability and other privacy notions in the remainder of this thesis.

Chapter 3 introduces a general model where we capture several definitions of unlinkability from the literature to compare the intuition behind them and to fully understand the privacy guarantees that they offer. Then, we

formalise the stronger properties of forward and backward privacy. This is a joint work with K. Chatzikokolakis, S. Etalle, and J. den Hartog, published in [13], later extended in [12].

Chapter 4 presents a model where we redefine unlinkability, forward and backward privacy, but this time in a more concrete and specific way. In fact, this model is an instance of the model in Chapter 3, meant for protocol verification rather than properties comparison. Here, we take into account RFID features in the protocol and properties descriptions and provide definitions of those model elements that were left abstract in Chapter 3. This work is the result of a collaboration with K. Chatzikokolakis and J. den Hartog [11], later extended in [10].

Chapter 5 presents a class of single-step protocols, namely identification protocols consisting of one message from a tag to a reader. First, we discuss the privacy guarantees that single-step protocols provide with respect to the definitions given in Chapter 3. This work is presented in [13, 12]. Then, we provide necessary and sufficient conditions for these protocols to satisfy all the privacy definitions given in Chapter 4. Finally, we prove that some protocols from the literature satisfy these conditions and, therefore, guarantees the privacy definitions of Chapter 4. These results are discussed in [11, 10]

Chapter 6 presents our conclusions and provides possible directions for future work.

2

PRELIMINARIES

The objectives of this thesis are to clarify the notion of privacy in the context of RFID systems and to provide a model for RFID systems that supports defining and verifying privacy properties. To achieve these goals we first need to introduce epistemic logic (Section 2.1) and the applied pi calculus (Section 2.2) that we use in the next chapters.

2.1 Epistemic logic with public announcements

Epistemic logic with public announcements [30] is a logic modelling agent knowledge. This logic allows us to define and compare privacy definitions in a very intuitive manner, namely in terms of an attacker's knowledge. In this section we provide the syntax and semantics of the epistemic logic that we use in this thesis.

Given a set of propositional constants (atoms) P, the set $\mathfrak{L}(P)$ of epistemic formulas φ, ϕ, ... 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 negation *not* of a formula, the conjunction *and* and the disjunction *or* of two formulas, 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. We only consider the knowledge of a single agent, namely the global attacker. Thus, we simplify the notation by omitting the agent identifier in the knowledge formula $K\varphi$. Finally, $[\varphi]\psi$ means "after φ

is revealed, ψ holds".

The semantics of epistemic logic 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.

Intuitively, $s_1 \sim s_2$ means that, from the attacker's point of view, the two states are indistinguishable.

For a Kripke structure M and a state $s \in S$ the semantics of the logic is given by

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

$M, s \vDash p$ means that M satisfies p at state s. The rules for $\wedge, \vee$ and $\neg$ are standard. The knowledge formula $K\varphi$ holds when the attacker knows φ at state s, which is true if and only if 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 by M by restricting it only to states satisfying φ, i.e. having state space $S' = \{s \in S \mid M, s \vDash \varphi\}$. Intuitively, revealing $[\varphi]$ in a state where φ holds, restricts the model to a smaller one where φ always holds. Thus, $[\varphi]\psi$ is true if ψ holds in the restricted model.

From now on we simply write $s \vDash \varphi$ for $M, s \vDash \varphi$.

2.2 Applied pi calculus

The applied pi calculus [1] is a language for the description of concurrent processes and their interaction. Similarly to the pi calculus [31], it provides simple syntax and semantics for the description of cryptographic protocols, but adds the possibility to model cryptographic primitives through a signature and an equational theory.

2.2.1 Syntax

The ingredients to define terms in the applied pi calculus are a set of names, a set of variables and a signature Σ, which consists of a finite set of function symbols. Given these ingredients, the set of *terms* can be defined by the grammar shown in Figure 2.1.

$T ::=$	**Terms**
$a, b, c, \ldots$	name
x, y, z	variable
$f(T_1, \ldots, T_l)$	function application

FIGURE 2.1: Terms of the applied pi calculus

A set of terms contains names, variables and all the terms that can be built up by applying function symbols to other terms. In particular, f is a function symbol that ranges over Σ and has an arity of l. A function symbol with arity 0 is a constant symbol. The signature Σ is equipped with an equational theory, i.e. an equivalence relation $=_{\mathrm{E}}$ on terms that is typically assumed to be closed under one-to-one substitution of names [1]; in this thesis we assume a slightly stronger property, namely that $=_{\mathrm{E}}$ is closed under one-to-one substitution of fresh variables for names. This means that from an equation, say $\mathbf{f}(n) =_{\mathrm{E}} \mathbf{g}(\mathbf{g}(n))$, we can not only deduce that $\mathbf{f}(m) =_{\mathrm{E}} \mathbf{g}(\mathbf{g}(m))$ for any name m, but also $\mathbf{f}(x) =_{\mathrm{E}} \mathbf{g}(\mathbf{g}(x))$. Note that this stronger property holds for all theories generated by a finite number of axioms, which is the usual way of generating theories. We finally assume that $=_{\mathrm{E}}$ is not the trivial theory equating all terms.

The signature together with the equational theory are used to model cryptographic primitives. For instance, to model symmetric encryption we may include in Σ the function symbols $\mathbf{encrypt}$ and $\mathbf{decrypt}$, both with arity 2. Typically, the parameters of $\mathbf{encrypt}$ are two terms corresponding to a cleartext and a symmetric key, while the ones of $\mathbf{decrypt}$ are a ciphertext and a

$P, Q, R ::=$	**plain processes**
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	restriction
if $M = N$ then P else Q	conditional
$u(x).P$	message input
$\overline{u}\langle N \rangle.P$	message output
$A, B, C ::=$	**extended processes**
P	plain process
$A \mid B$	parallel composition
$\nu n.A$	name restriction
$\nu x.A$	variable restriction
$\{^{M}/_{x}\}$	active substitution

FIGURE 2.2: Syntax of the applied pi calculus

symmetric key. If the key used to decrypt a ciphertext corresponds to the key used for its encryption, the decryption function $\mathbf{decrypt}$ should return the correct cleartext. To model this, we can define the following equation in our equational theory

$$\mathbf{decrypt}(\mathbf{encrypt}(x, k), k) =_{\mathrm{E}} x$$

Metavariables u, v, w are used for both names and variables. We denote by $\widetilde{n}, \widetilde{x}, \widetilde{M}$ a (possibly empty) sequence of names, variables and terms, respectively. We write $\{\widetilde{y}\}$ for the set of elements in a sequence $\widetilde{y}$. We write $M \trianglelefteq N$ if and only if M is a subterm of N and $M \triangleleft N$ if and only if $M \trianglelefteq N$ and $M \neq N$, and $M =\!\trianglelefteq N$ if and only if $\exists M' : M =_{\mathrm{E}} M' \trianglelefteq N$ and $M =\!\triangleleft N$ if and only if $\exists M' : M =_{\mathrm{E}} M' \triangleleft N$.

The syntax for *processes* is shown in Figure 2.2. The null process does nothing. In the parallel composition of two processes $P \mid Q$, the processes P and Q are executed concurrently. The replication process $!P$ behaves as the parallel composition of an infinite number of copies of P. We write $\prod_{i=1}^{n} P_i$ for the parallel composition of n processes $P_1, \ldots, P_n$. The restriction process $\nu n.P$ creates a new private name n and continues as P. We write $\nu\widetilde{n}$ or $\nu n_1, n_2, \ldots$ for a sequence of restrictions $\nu n_1.\nu n_2. \ldots$. The standard primi-

tive if-then-else tests for the equality (according to the equational theory) of M and N. The input process $u(x).P$ awaits a message from channel u; when a message is received, the process continues as P, where all the occurrences of x are replaced by the received message. Symmetrically, the output process $\overline{u}\langle N\rangle.P$ is the process that sends a message N on the channel u and then continues as P.
Figure 2.2 also presents the syntax for *extended processes*, which extends the syntax for plain processes with active substitutions. An active substitution $\{^{M}/_{x}\}$ replaces a variable x with a term M and models the information known to the environment. We assume that substitutions are cycle-free. We use $\{^{\widetilde{M}}/_{\widetilde{x}}\}$ for a sequence of substitutions. Finally, we sometimes write $N[^{M}/_{x}]$ for the term obtained by substituting M for x in N. Note that $N[^{M}/_{x}]$ substitutes a term for a variable in a term, while an active substitution affects all the processes that are in its scope.

Names and variables have a scope that is delimited by restrictions. An input $u(x).P$ also implicitly restricts x. Given an extended process A, the set $\mathbf{bn}(A)$ contains all the names that are bound by a restriction in A, while the set $\mathbf{bv}(A)$ contains all the variables bound by a restriction or an input process. Symmetrically, the sets $\mathbf{fn}(A)$ and $\mathbf{fv}(A)$ contain all the names and variables, respectively, that are not bound in A. Note that an active substitution $\{^{M}/_{x}\}$ does not bind names or variables (the variable x is free). We say that an extended process is closed when its variables are either bound or defined by an active substitution.

Finally, every extended process can be mapped into a *frame*, denoted by ψ and φ, by replacing all the plain processes with the null process. A frame is an extended process built up from 0 and active substitutions of the form $\{^{M}/_{x}\}$ by parallel composition and restriction, and it collects the static knowledge that the corresponding extended process outputs to its environment. The domain of a frame ψ, written $\mathrm{dom}(\psi)$, is the set of variables x in the substitutions $\{^{M}/_{x}\}$ that are not under restriction. The domain of an extended process is that of its frame. A frame ψ is in *canonical form* if and only if $\psi = \nu\widetilde{n}.\{^{\widetilde{M}}/_{\widetilde{x}}\}$ where $\mathbf{fv}(\widetilde{M}) = \emptyset$ and $\{\widetilde{n}\} \subseteq \mathrm{fn}(\widetilde{M})$.

2.2.2 Operational semantics

The operational semantics of the applied pi calculus describes the transitions that extended processes can perform and it is given in terms of two relations: *structural equivalence* $\equiv$ and *internal reduction* $\rightarrow$.

Structural equivalence $\equiv$ is the smallest equivalence relation on extended

$$A \mid 0 \equiv A \qquad \text{PAR-0}$$
$$A \mid (B \mid C) \equiv (A \mid B) \mid C \qquad \text{PAR-A}$$
$$A \mid B \equiv B \mid A \qquad \text{PAR-C}$$
$$!P \equiv !P \mid P \qquad \text{REPL}$$
$$\nu n.0 \equiv 0 \qquad \text{NEW-0}$$
$$\nu u.\nu v.A \equiv \nu v.\nu u.A \qquad \text{NEW-C}$$
$$\nu x.\{^M/_x\} \equiv 0 \qquad \text{ALIAS}$$
$$\{^M/_x\} \mid A \equiv \{^M/_x\} \mid A\{^M/_x\} \qquad \text{SUBST}$$
$$A \mid \nu u.B \equiv \nu u.(A \mid B) \qquad \text{NEW-PAR}$$
$$\text{when } u \notin \mathbf{fv}(A) \cup \mathrm{fn}(A)$$
$$\{^M/_x\} \equiv \{^N/_x\} \qquad \text{REWRITE}$$
$$\text{when } M =_{\mathrm{E}} N$$

FIGURE 2.3: Structural equivalence

processes that is closed under α-conversion on both names and variables, is closed under application of evaluation contexts, and satisfies the rules of Figure 2.3.

The rules for parallel composition (PAR-0, PAR-A, PAR-C), replication (REPL) and restriction (NEW-0, NEW-C) are standard. ALIAS enables the introduction of an arbitrary active substitution. SUBST describes the application of an active substitution to a process that is in contact with it. NEW-PAR allows to move a restriction νu without changing the meaning of the process. REWRITE allows to replace a term for another term that is equivalent to it in the specified equational theory. It can be shown that any frame φ is structurally equivalent to a frame φ' in canonical form.

Internal reduction ($\rightarrow$) is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts such that

$$\overline{a}\langle x\rangle.P \mid a(x).Q \rightarrow P \mid Q \qquad \text{COMM}$$
$$\text{if } M = M \text{ then } P \text{ else } Q \rightarrow P \qquad \text{THEN}$$
$$\text{if } M = N \text{ then } P \text{ else } Q \rightarrow Q \qquad \text{ELSE}$$

for any ground terms M and N such that $M \neq_{\mathrm{E}} N$. The rule COMM represents a communication on channel a defined on variables. The rules THEN and

ELSE compare terms and continue as P if the terms are equivalent in the equational theory, as Q otherwise. They may require the application of active substitutions in the context to ensure that M and N are ground terms.

2.2.3 Observational equivalence

Several privacy properties can be formalised in terms of observational equivalence between processes. Two processes are equivalent if an external observer cannot tell them apart. To understand observational equivalence we need to introduce the concept of *context*, that is an extended process $C[-]$ with a hole replacing an extended sub-process. A context $C[-]$ represents an environment, thus it may be used to model an attacker. An *evaluation context* is a context whose hole is not under a replication, a conditional, an input, or an output process. A context $C[-]$ *closes* A when $C[A]$ is closed. The definition of observational equivalence contains the syntax $A \Downarrow a$ for "A can send a message on a channel a", that is when $A \rightarrow^* C[\overline{a}\langle M\rangle.P]$, where $\rightarrow^*$ indicates zero or any finite number of steps $\rightarrow^*$, for some evaluation context C that does not bind a.

DEFINITION 2.2.1. *Observational equivalence ($\approx$) is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A\mathcal{R}B$ implies*

1. *if $A \Downarrow a$ then $B \Downarrow a$;*
2. *if $A \rightarrow^* A'$ then $B \rightarrow^* B'$ and $A'\mathcal{R}B'$ for some B';*
3. *$C[A]\mathcal{R}C[B]$ for all closing evaluation contexts C.*

Hence, two processes are observationally equivalent when, for all closing evaluation contexts, they can output on the same channel and perform the same reduction steps, which must lead to two observationally equivalent new processes.

2.2.4 Static equivalence

Observational equivalence may be difficult to handle in proofs. Therefore, in this thesis we use labelled bisimilarity that is proven to be equivalent to observational equivalence ($\approx_l = \approx$) in [1]. To define labelled bisimilarity we need to introduce the concept of static equivalence.

Static equivalence compares the static knowledge that is output by two extended processes to their environment.

DEFINITION 2.2.2. *Two terms M and N are* equal in the frame φ, *written* $(M = N)\varphi$, *if and only if* $\varphi \equiv \nu\widetilde{n}.\sigma$, $M\sigma = N\sigma$, *and* $\{\widetilde{n}\} \cap \{fn(M) \cup fn(N)\} = \emptyset$ *for some names* $\widetilde{n}$ *and substitution* σ.

Two closed frames φ, ψ are statically equivalent, written $\varphi \approx_s \psi$, when $\mathrm{dom}(\varphi) = \mathrm{dom}(\psi)$ *and, for all terms M and N, we have $(M = N)\varphi$ if and only if $(M = N)\psi$.*

Two closed extended processes are statically equivalent, written $A \approx_s B$, if the frame of A is statically equivalent to the frame of B.

Intuitively, two closed extended processes are statically equivalent when their static knowledge cannot be distinguished. Their static knowledge is the information that is already known to the environment and it does not include the information that the processes may send after some transitions. In fact, static equivalence does not consider the dynamic behaviour of the processes.

2.2.5 Labelled operational semantics and equivalence

The labelled operational semantics extends the operational semantics of Section 2.2.2 and defines a relation $A \xrightarrow{\alpha} A'$, where α is a label that corresponds to an input $a(M)$ where M is a term, or to an output $\overline{a}\langle u\rangle$ or $\nu u.\overline{a}\langle u\rangle$, where u is a variable or a name. This semantics allows us to reason about processes interaction with their environment. In addition to the operational semantics rules of Section 2.2.2, it also satisfies the rules of Figure 2.4. The rule IN inputs a term M from the channel a while OUT-ATOM and OPEN-ATOM model the output of a name or variable u on the channel a. In particular, OPEN-ATOM may be used to output a restricted channel (when u is a name) as well as a term (when u is a variable that is bound to a term). The rule SCOPE allows a process under a restriction to perform a labelled transition step when the restriction does not appear in the label. PAR and STRUCT deal with the parallel composition and the structural equivalence of processes, respectively. The assumption $\mathbf{bv}(\alpha) \cap \mathbf{fv}(B) = \mathbf{bn}(\alpha) \cap \mathbf{fn}(B) = \emptyset$ in the rule PAR prevents the transition $A \xrightarrow{\alpha} A'$ from changing the meaning of the process B.

A labelled bisimilarity $\approx_l$ between extended processes requires the processes to be statically equivalent and to perform the same reduction steps, which must lead to two statically equivalent new processes.

DEFINITION 2.2.3. *A labelled bisimilarity ($\approx_l$) is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A\mathcal{R}B$ implies*

1. $A \approx_s B$;

$$a(x).P \xrightarrow{a(M)} P\{^{M}/_{x}\} \qquad \text{IN}$$

$$\overline{a}\langle u\rangle.P \xrightarrow{\overline{a}\langle u\rangle} P \qquad \text{OUT-ATOM}$$

$$\frac{A \xrightarrow{\overline{a}\langle u\rangle} A' \qquad u \neq a}{\nu u.A \xrightarrow{\nu u.\overline{a}\langle u\rangle} A'} \qquad \text{OPEN-ATOM}$$

$$\frac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'} \qquad \text{SCOPE}$$

$$\frac{A \xrightarrow{\alpha} A' \qquad \mathbf{bv}(\alpha) \cap \mathbf{fv}(B) = \mathbf{bn}(\alpha) \cap \mathbf{fn}(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B} \qquad \text{PAR}$$

$$\frac{A \equiv B \qquad B \xrightarrow{\alpha} B' \qquad B' \equiv A'}{A \xrightarrow{\alpha} A'} \qquad \text{STRUCT}$$

FIGURE 2.4: Labelled semantics rules

2. *if* $A \rightarrow A'$*, then* $\exists B'$ *such that* $B \rightarrow^* B'$ *and* $A' \mathcal{R} B'$*;*

3. *if* $A \xrightarrow{\alpha} A'$ *and* $\mathbf{fv}(\alpha) \subseteq \mathrm{dom}(A)$ *and* $\mathbf{bn}(\alpha) \cap \mathbf{fn}(B) = \emptyset$*, then* $B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'$ *and* $A' \mathcal{R} B'$ *for some* B'*.*

2.3 Conclusions

In this chapter we presented the epistemic logic and the applied pi calculus, the basis that we use to formalise privacy models in the remainder of the thesis. The epistemic logic is used in Chapter 3 to define a general formal model and several privacy properties. In Chapter 4, we use the applied pi calculus to develop an RFID model where the privacy properties are redefined taking into account specific RFID aspects. Finally, in Chapter 5, we use both the resulting models to study and verify the properties for a class of protocols.

3

FORMALISATION OF PRIVACY PROPERTIES

As explained in the introduction, given the lack of agreement on the exact definition of privacy, our first goal is to clarify which are the privacy guarantees that an identification protocol should satisfy. In this chapter we provide a unifying framework that we use to model and compare different definitions of the unlinkability property. All these definitions are expressed in terms of an attacker's knowledge using epistemic logic. Also, we give examples to show that all these definitions are different, but we demonstrate that, in many practical situations, they coincide. Finally, we provide definitions for the concepts of forward and backward privacy, which both are stronger than unlinkability.

3.1 An abstract trace-based model

In this section we present our abstract formal model, which will be the basis for formalising several privacy properties using epistemic logic in Section 3.2.

An RFID system consists of a number of tags, readers and backend systems, which interact to execute the steps of an identification protocol. The purpose of an identification protocol is to identify a tag to a backend system. The reader, which must be in the proximity of the tag, is used as an intermediary. Communication between tags and readers takes place over an insecure (wireless) channel, where 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 is an abstract collection of protocol runs consisting of one or multiple messages. A transaction starts when the attacker gains access to a tag 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 tag participating in the transaction does not change. For example, the attacker may see a tag at the entrance of a building at 12:00 and can interact with it. The attacker can query the tag many times, and he is sure that the same tag responds every time, since there is no other tag around. When he loses proximity to the tag, this transaction ends. Then, at the same or some other location, the attacker once again sees a tag and can interact with it in a different transaction. The tag can be either the same as before, or a different one.

The goal of the attacker is to trace tags by linking transactions. In our model, the attacker's approach is captured by the concept of *strategy*. For example, the attacker might passively eavesdrop the session of the first agent he 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. Note that when the model captures an RFID system, the agents correspond to tags.

For simplicity, we assume that all interactions happen between the attacker and the tags (honest sessions are also allowed, since the attacker can forward messages between agents) as in the Dolev-Yao model.

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 details, we consider a protocol as a set of traces composed of strategies and agent mappings, and we define privacy properties based on the attacker's ability to distinguish traces.

DEFINITION 3.1.1. *A system is a tuple* $(A, \Sigma, \mathrm{T}, \sim)$ *where*

- $A = \{a_1, a_2, \ldots\}$ *is a (possibly infinite) set of agents;*
- Σ *is a set of strategies; each strategy* $\sigma \in \Sigma$ *has a length* $|\sigma|$*;*
- $\mathrm{T} = \{(\pi, \sigma) \mid \sigma \in \Sigma, \pi \in \Pi^{|\sigma|}\}$*, where* $\Pi^{|\sigma|}$ *is a set of agent mappings of length* $|\sigma|$*, is the set of traces; each trace* $\tau \in \mathrm{T}$ *is a pair* (π, σ) *where* $\sigma \in \Sigma$ *is a strategy and* $\pi \in \Pi^{|\sigma|}$ *is a sequence of agents;*
- $\sim$ *is an equivalence relation on* T *such that* $(\pi_1, \sigma_1) \sim (\pi_2, \sigma_2) \Rightarrow \sigma_1 = \sigma_2$.

A strategy $\sigma \in \Sigma$, chosen by the attacker, describes what the attacker does in each transaction. We do not specify this further than the number of transactions $|\sigma|$ in the strategy.

To still be able to talk about specific transactions we assume an ordered set of transaction names $\{p_1, \ldots, p_n\}$. We interpret $\pi \in \Pi^n$ as a mapping function from $\{p_1, \ldots, p_n\}$ to A. We use $Dom_{(\pi,\sigma)}$ to denote the domain of this function, i.e. $\{p_1, \ldots, p_{|\sigma|}\}$.

A strategy σ and a mapping π together form a trace τ, which represents a complete execution of the system.

Finally, $\sim$ captures which traces are indistinguishable for the attacker. An equivalence between traces is a fundamental concept underlying all the unlinkability definitions 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, 11]) or using the concept of "reinterpretation" [38]. In computational models, the equivalence is stated in terms of the inability of the attacker to distinguish the two cases in the corresponding game.

We write S^n to denote a sequence of n elements from the set S. We use S^* for a sequence of an arbitrary number of elements from the set S.

In our model, a protocol is an abstract object that describes the behaviour of the agents in a system. Given a protocol description, the corresponding set T contains 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 agents he was interacting with.

We sometimes use π_τ to emphasise that π belongs to the trace τ. For a trace $\tau = (\pi, \sigma)$ we define A_τ and A_π as the image of π (i.e. the set of agents involved in the trace). We write Π for $\cup_{n \geq 1} \Pi^n$.

We use $\sim$ also for mappings as follows:

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

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 Chapter 5). 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 Chapter 5 to study a class of protocols.

3.2 Interpretation of existing 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. We also express the definitions in terms of trace equivalence, which provides the restrictions on the traces for the privacy properties to hold more explicitly.

First, we state the definition of *weak unlinkability* from [38, 3]. This definition requires that for every pair of transactions (p, p') the attacker cannot *know* whether they are linked.

Second, we show that there are several definitions in the literature that are stronger than weak unlinkability. Therefore, we first study the notion of *strong unlinkability* inspired by the work 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*. These definitions, which often appear in the literature in the computational setting [18, 27, 4, 32], are stronger than weak unlinkability, but are incomparable to strong unlinkability. 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 and we call them *two-agents game unlinkability* and *three-agents game unlinkability*.

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 all the definitions in our trace-based model, and then we compare them 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 3.1 gathers all the resulting epistemic notions.

Property	*For any trace τ the attacker cannot infer*
Weak U $\neg K(link(p, p'))$	that two transactions p, p' are linked
Strong U $\neg K(anyLink(\tau))$	the existence of linked transactions
Two-agents game U $[\pi_a \vee \pi_{a'}]\neg K\pi_a$	the mapping π_a even when $\pi_a \vee \pi_{a'}$ is revealed
Three-agents game U $[\pi_{a,a} \vee \pi_{a_1,a_2}]\neg K\pi_{a,a}$	the mapping $\pi_{a,a}$ even when $\pi_{a,a} \vee \pi_{a_1,a_2}$ is revealed
Weak I $\neg K(unlink(p, p'))$	that two transactions p, p' are unlinked
Strong I $\neg K(anyUnlink(\tau))$	the existence of unlinked transactions

TABLE 3.1: Epistemic definitions of unlinkability (U) and inseparability (I)

3.2.1 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 3.1.

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 attacker cannot distinguish between those executions.

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. The set of atomic propositions P and the assignment function $f : \mathrm{T} \to P$ are built as follows:

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

We use two types of propositions: $\pi \in \Pi$ simply denotes that the mapping of

a trace is π; $link(p, p')$ denotes that the transactions p, p' are linked, and it is true in a trace $\tau = (\pi, \sigma)$ if and only if both transactions are mapped to the same agent in that trace.

3.2.2 Weak Unlinkability

The first definition that we study is the one of weak unlinkability of van Deursen et al. [38] and Arapinis et al. [3]. Although presented in different models, the two definitions are similar in nature. They require that, given a trace with two linked transactions (sent by the same agent), an equivalent trace must exist where the corresponding transactions are not linked. This can be expressed using epistemic logic as follows.

DEFINITION 3.2.1 (Weak unlinkability). *A protocol* T *guarantees weak unlinkability if and only if*

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

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 the traces τ and all the pairs of distinct transactions, there must exist an equivalent trace $\tau' \sim \tau$ in which the corresponding transactions are mapped to two different agents. Hence, the above definition can be written as follows.

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

This formulation of unlinkability corresponds exactly to the ones of [38, 3]. Note that if $\tau \vDash \neg link(p, p')$ then we can simply take $\tau' = \tau$.

The weakness of this definition lies in the existential 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 following example.

Example 1. Let T be the set of traces of a given protocol that guarantees weak unlinkability in a system with at least three agents. 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 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, but τ_1 and τ_2 are indistinguishable. Weak unlinkability is satisfied, in fact it is easy to see that each pair of linked transactions in a trace is unlinked in another. Still, when the attacker sees τ_1 (or τ_2), he knows that p_1 is linked to either p_2 or p_3 in the trace. Formally, $\tau_1 \vDash K(link(p_1, p_2) \vee link(p_1, p_3))$.

3.2.3 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. A simplified form of their definition, which uses the applied pi calculus, requires that

$$\begin{aligned} !\,T \approx !\,T_s \quad \text{where} \\ T = \nu m.\ init.\ !main \\ T_s = \nu m.\ init.\ main \end{aligned} \tag{3.1}$$

where T represents an agent carrying out an initialisation 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 because it is directly expressible 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

$$anyLink(\tau) = \bigvee_{p \in Dom_\tau} \bigvee_{p' \neq p \in Dom_\tau} link(p, p'). \tag{3.2}$$

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

DEFINITION 3.2.2 (Strong unlinkability). *We say that a protocol* T *guarantees strong unlinkability if and only if*

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

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

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

This formulation corresponds exactly to the equivalence (3.1) since τ' is a trace that can be produced by the process $!T_s$, where no agent executes more than one transaction.

Clearly, strong unlinkability is stronger than weak unlinkability, as stated below.

THEOREM 3.2.1. *Strong unlinkability implies weak unlinkability.*

PROOF. We need to prove that strong unlinkability implies weak unlinkability, namely that

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

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

$$\forall P_i : \bigvee_i K(P_i) \Rightarrow K \bigvee_i P_i.$$

For all $\tau \in \mathrm{T}$ we have

$$\begin{array}{lll} \tau \vDash \neg K(anyLink(\tau)) & & \\ \equiv & \text{[Def. 3.2]} & \tau \vDash \neg K \bigvee_p \bigvee_{p' \neq p} link(p, p') \\ \Rightarrow & \text{[Prop. } K\text{]} & \tau \vDash \neg \bigvee_p K \bigvee_{p' \neq p} link(p, p') \\ \equiv & & \forall p \in Dom_\tau : \tau \vDash \neg K \bigvee_{p' \neq p} link(p, p') \\ \Rightarrow & \text{[Prop. } K\text{]} & \forall p \in Dom_\tau : \tau \vDash \neg \bigvee_{p' \neq p} K(link(p, p')) \\ \equiv & & \forall p, p' \in Dom_\tau, p \neq p' : \tau \vDash \neg K(link(p, p')). \end{array}$$

Therefore strong unlinkability implies weak unlinkability. Note that this result is also proven in [3] in their setting. □

3.2.4 Game-based definitions of privacy

We now give two definitions of privacy based on privacy games. We believe that these two definitions cover most privacy definitions from the literature and show that they are equivalent in our model. Then, we give a simpler translation based on trace equivalence and demonstrate that it is also equivalent.

In all the game-based 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.

The definitions can be split in two categories. The first is related to the definition given by Ohkubo et al. [33], variations of which can be found also in [18, 27, 4, 34], while the second corresponds to the definitions given in [18, 32]. 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 two-agents game 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. 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 he 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:

- $\pi_x \in \Pi_x^n$ is a partial mapping $(A \cup \{x\})^n$. We interpret π_x as a partial mapping from transactions to agents or a variable x. This mapping represents the situation in which the attacker knows the identities of the agents involved in all the transactions of a trace, except for the ones mapped to x.
- Π_x denotes the set of all partial mappings (for all n).
- π_a is a complete mapping obtained from π_x by mapping to an agent a all the transactions previously mapped to the variable x.

In our model, we formalise the unlinkability game by requiring that the attacker cannot infer whether he 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 T guarantees *two-agents game unlinkability* if and only if

$$\forall \tau \in \mathrm{T},\ a, a' \in A, a \neq a', \pi_x \in \Pi_x : \tau \vDash [\pi_a \vee \pi_{a'}] \neg K(\pi_a). \tag{3.3}$$

Note that, although the only forbidden knowledge concerns π_a, (3.3) 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 (3.3) as

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

We now turn our attention to the three-agent game type of game [18, 32]. As for the previous game, during the first phase the attacker is allowed to interact with all the agents. Then, he selects three agents a, a_1, a_2 to be challenged on. 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 he can infer whether x and y are linked.

Again, we need to introduce some notation

- $\pi_{x,y} \in \Pi^n_{x,y}$ is a dual variable partial mapping $(A \cup \{x, y\})^n$. We interpret $\pi_{x,y}$ as a partial mapping from transactions to agents or variables x and y. This represents the situation in which the attacker knows the identities of the agents involved in all the transactions except for the ones mapped to 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 the transactions previously mapped to the variable x and an agent b to the variable y.

We require that the attacker cannot infer whether he is given a mapping $\pi_{a,a}$ or π_{a_1,a_2}. Formally, we say that a protocol T guarantees three-agents game unlinkability if and only if

$$\forall \tau \in \mathrm{T},\ a, a_1, a_2 \in A, a_1 \neq a_2,\ \pi_{x,y} \in \Pi_{x,y} : \\ \tau \vDash [\pi_{a,a} \vee \pi_{a_1,a_2}] \neg K(\pi_{a,a}). \quad (3.4)$$

In terms of equivalence of traces, (3.4) can be expressed as

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

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

DEFINITION 3.2.3 (Game-based unlinkability). *We say that a protocol* T *guarantees game-based unlinkability if and only if*

$$\forall \pi, \pi' \in \Pi, |\pi| = |\pi'| : \pi \sim \pi'. \quad (3.5)$$

Each of the referenced works uses a variant of either (3.3) or (3.4), while [18] 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 3.2.3.

THEOREM 3.2.2. *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.*

The proof of Theorem 3.2.2 can be found in Appendix A.1.

From now on we will only use the definition of game-based unlinkability. However, by Theorem 3.2.2, all the results that involve game-based unlinkability in the following hold also for two-agents game unlinkability and three-agents game unlinkability.

As one may expect, game-based unlinkability is stronger than weak unlinkability, as stated below. Hence, we consider game-based unlinkability as a strong definition of unlinkability.

THEOREM 3.2.3. *Game-based unlinkability implies weak unlinkability.*

PROOF. It follows directly from the definition of game-based unlinkability. In fact, it implies the equivalence of *all* the traces, thus also weak unlinkability, which only requires the equivalence of *some* traces. □

In Section 3.3.1 it is shown that weak unlinkability actually differs from the strong and game-based definitions of unlinkability, and that strong unlinkability is incomparable to game-based unlinkability.

3.2.5 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. To model this property

we need to introduce the concept of *inseparability*, which requires that the attacker cannot infer whether two specific transactions are *not* linked. As for unlinkability, we introduce a weak and a strong form of this privacy property.

DEFINITION 3.2.4 (Weak inseparability). *We say that a protocol* T *guarantees weak inseparability if and only if*

$$\forall \tau \in \mathrm{T},\ p, p' \in Dom_\tau, p \neq p' : \tau \vDash \neg K(unlink(p, p')) \tag{3.6}$$

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, (3.6) corresponds to

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

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 3.2.5 (Strong inseparability). *A protocol* T *guarantees strong inseparability if and only if*

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

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

Similarly to unlinkability and weak inseparability, also the definition of strong inseparability can be expressed in a direct way in terms of trace equivalence

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

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 every trace is equivalent 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, but about the attacker seeing a difference between two observations, therefore it offers both "unlinkability" and "inseparability" guarantees. Note that this implication would not hold in systems that limit the number of sessions that a tag can execute. In this case, a trace run by the same tag would not exist for a strategy that executes more sessions than the limit imposed by the system, hence strong inseparability would not hold, while game-based unlinkability could still be guaranteed. However, we do not consider this possibility, since we assume that a tag may run an infinite amount of protocol executions in our model.

THEOREM 3.2.4. *Game-based unlinkability implies strong inseparability, which implies weak inseparability.*

PROOF. The first part of the theorem states that the game-based definition of unlinkability implies strong inseparability. 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. Hence, they are also 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 (see proof of Theorem 3.2.1). □

3.3 Comparison between definitions

In Section 3.2 we gave three definitions of unlinkability as well as two of inseparability. Here, we show that these properties are in general different by providing examples that illustrate such differences. However, these examples have features that are unlikely to be found in practice. Therefore, we investigate conditions under which some or all of the properties become equivalent. In Chapter 5 we also formally prove that a class of protocols satisfies all these conditions and we argue that most RFID protocols actually satisfy them, at least in their abstract form.

3.3.1 Protocols where the properties do not coincide

This section lists some examples of RFID protocols that guarantee only some of the properties described in Section 3.2. The examples are constructed and use artificial settings. Indeed, the differences between the properties arise from features of these examples that are unlikely to be present in realistic applications.

In the next section, we identify some conditions under which weak and strong properties become equivalent. The examples correspond to or are variations of the OSK protocol for RFID systems [33]. In the protocol, each tag is initialised with a unique secret. During each session, the tag calculates the hash $\mathbf{g}$ of the current secret and outputs it. Then, it updates the state using a different hash function $\mathbf{h}$. The OSK protocol guarantees strong unlinkability under certain assumptions, as we demonstrate in Chapter 5, where we also give a more detailed description of the protocol.

Example 2 (System with a bounded number of tags). Consider the OSK protocol in a system with a bounded number of tags. Note that, though the number of agents in the model is infinite, the number of tags in a system is determined by T, the set of traces of the protocol, which may limit it.

- ✔ Game-based unlinkability: all the traces of equal length produced by the OSK protocol are equivalent, therefore game-based unlinkability holds.

- ✘ Strong unlinkability: the original OSK protocol is considered to guarantee strong unlinkability; however, when the number of tags in the system is bounded, the attacker who sees a number of sessions greater than the number of agents knows about the existence of linked sessions, although he cannot point to any specific message. In Chapter 5 we actually prove that the OSK protocol guarantees strong unlinkability in the model of [3], but this is because this model implicitly assumes the existence of an unbounded number of tags.

- ✔ Weak unlinkability: this property holds because the attacker, who does not know any tag secret, is unable to link two specific messages.

- ✔ Strong inseparability: 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 trace length; therefore strong inseparability holds.

- ✔ Weak inseparability: the protocol guarantees strong inseparability, which implies weak inseparability.

Example 3 (System with several "types" of tags). Consider the OSK protocol in a system where there are two types of tags, $Type_1$ and $Type_2$. We assume that the attacker can distinguish these two types, for example because the tags have different technical characteristics.

- ✘ Game-based unlinkability: this property is violated, because the attacker can trivially distinguish some traces from others (e.g. one completely linked from one executed by tags of different type).

- ✔ Strong unlinkability: it holds because the attacker cannot infer 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 definition of unlinkability are incomparable.

- ✔ Weak unlinkability: the protocol guarantees strong unlinkability, which implies weak unlinkability.

- ✘ Weak inseparability: when two transactions of different types are observed, the attacker knows that the transactions cannot come from the same tag, thus weak inseparability is violated.

- ✘ Strong inseparability: the protocol does not guarantee weak inseparability, which implies that strong inseparability is also violated.

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). Consider a case where the number of protocol sessions belonging to tags of $Type_2$ in the trace is greater than the number of tags of $Type_2$. Then strong unlinkability is violated while weak unlinkability still holds.

Example 4 (System forcing the execution of a single type of tag per trace). Consider the OSK protocol in a system where there are two distinguishable types of tags, $Type_1$ and $Type_2$. We assume that the system allows only one type of tag within a trace.

- ✘ Game-based unlinkability: it does not hold, because a trace run by tags of type $Type_1$ is always distinguishable from a trace run by tags of type $Type_2$.

- ✔ Strong unlinkability: it holds because the attacker cannot infer the existence of any linked transactions since all transactions of the same type could come from different tags.
- ✔ Weak unlinkability: the protocol guarantees strong unlinkability, which implies weak unlinkability.
- ✔ Strong inseparability: it holds, because for each type of tag there always exists a trace entirely executed by a single agent of that type.
- ✔ Weak inseparability: the protocol guarantees strong inseparability, which implies weak inseparability.

Example 5 (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.

- ✘ Game-based unlinkability: since not all mappings are equivalent to each other due to the observable action, game-based unlinkability is violated.
- ✘ Strong unlinkability: the attacker knows that the session that made the reader beep must be linked to a past session of the trace, thus strong unlinkability is violated. For example, 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.
- ✔ Weak unlinkability: it holds because the beep does not allow the attacker to point to any two specific sessions and tell that they are linked.
- ✘ Weak inseparability: it is violated because the trace which causes the reader to beep at the third session tells the attacker that the first two sessions are *not* linked.
- ✘ Strong inseparability: this variation of the OSK protocol does not guarantee weak inseparability, which implies that strong inseparability is also violated.

Example 6 (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 and satisfies exactly

the same properties. However, the protocol fails to achieve some privacy goals for a different reason, hence the example leads to the definition of a different condition in the next section.

- ✘ Game-based unlinkability: as for the previous example, it is violated because the protocol leads to distinguishable observations.
- ✘ Strong unlinkability: it is violated. Consider a trace that causes a reader to beep after four or more sessions. This trivially implies that there must exist a link.
- ✔ Weak unlinkability: it is satisfied because no observation gives the attacker information about the messages that may be linked in a trace.
- ✘ Weak inseparability: since a beep during the third session means that the first three sessions are not linked, weak inseparability is violated.
- ✘ Strong inseparability: this variation of the OSK protocol does not guarantee weak inseparability, which implies that strong inseparability is also violated.

Example 7 (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.

- ✘ Game-based unlinkability: again, this protocol violates game-based unlinkability because it leads to distinguishable observations.
- ✘ Strong unlinkability: when a reader beeps the attacker knows that there is a link, thus strong unlinkability is violated.
- ✔ Weak unlinkability: it is satisfied because no observation tells to the attacker which specific messages are linked when a reader beeps.
- ✘ Strong inseparability: it is violated, because a beep means that at least two tags were involved in that trace.
- ✔ Weak inseparability: the attacker cannot point to two sessions and claim that they are unlinked; therefore, weak inseparability still holds.

Table 3.2 summarises only the negative relationships between the properties and lists the references to the corresponding examples. Figure 3.1 shows the relation between all the properties.

Now that we have shown that the privacy properties are different, we are ready to introduce the conditions under which they coincide.

Negative relationship			*References*
Weak unlinkability	$\not\Rightarrow$	Strong unlinkability	Examples 2,5,6,7
Weak unlinkability	$\not\Rightarrow$	Game-based unlinkability	Examples 3,5,6,7
Strong unlinkability	$\not\Rightarrow \setminus \not\Leftarrow$	Game-based unlinkability	Examples 2,3
Weak inseparability	$\not\Rightarrow$	Strong inseparability	Example 7
Strong unlinkability	$\not\Rightarrow$	Strong inseparability	Example 3
Strong inseparability	$\not\Rightarrow$	Game-based unlinkability	Example 4

TABLE 3.2: Negative relationship between properties

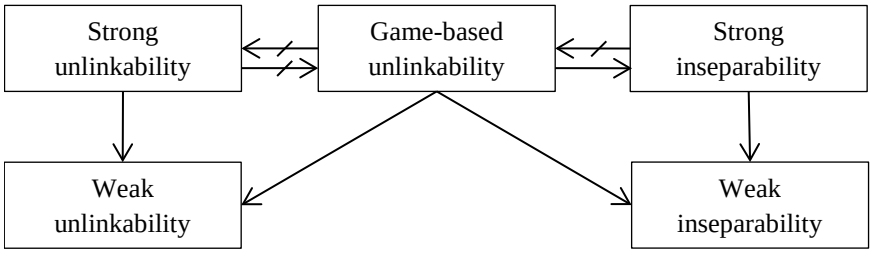

FIGURE 3.1: Relationship between properties

3.3.2 Conditions

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 that satisfy all the privacy properties defined in this chapter must contain an unbounded number of agents and a trace where each transaction is executed by different agents should always exists.

DEFINITION 3.3.1 (Unbounded number of agents). *We say that a protocol has an unbounded number of agents if and only if*

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

Clearly, an unbounded number of agents cannot exist in any real identification system. However, identification systems usually include a large number of agents, therefore an attacker cannot communicate with all the agents in a limited amount of time in a realistic situation. 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.

Condition: Renaming

As shown in Example 3, having multiple distinguishable types of agents can be problematic. For instance, observing two transactions run by two tags with distinguishable features clearly violates inseparability, as we can conclude that 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* the 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 3.3.2 (Renaming). *Let π be a mapping. 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 if and only if $\pi \sim \pi[a'/a]$ 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.

Note that in the remainder of the thesis, we assume that this condition holds and we write all the mappings in a normalised form, sorting the agents by their order of appearance: the first agent is always a_1, the next agent that differs 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 normalisation is that the number of normalised traces is always finite for any given length, even when the number of agents is infinite.

Condition: Swapping

The swapping condition requires that different agents act in an independent way and the execution of one agent should not affect the execution of another. We express it as follows. Consider $\pi_1 = (\ldots, a_1, a_2, \ldots)$ and $\pi_2 = (\ldots, a_3, a_4, \ldots)$, two mappings where the k-th (a_1 in π_1 and a_3 in π_2) and $(k+1)$-st (a_2 in π_1 and a_4 in π_2) transactions involve different agents (which could also appear elsewhere in the mappings). Now assume that $\pi_1 \sim \pi_2$, i.e. 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. The transactions of a_1, a_3 should not depend on whether a_2, a_4 were previously executed and vice versa. Thus, π_1' and π_2' should also be indistinguishable. Formally, we require the following condition:

DEFINITION 3.3.3 (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*

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

We say that a protocol satisfies the condition Swapping if and only if

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

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

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

Note that this condition is violated by the protocol in the Example 5. 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 $(\_, \_, \_)$. The reader beeps if it sees a linked transaction, but only if at least two agents

have identified themselves to the system. With a mapping (a_1, a_1, a_2) the reader sees a link during the second transaction, but does not beep because it has seen only one agent. Then, during the third transaction a second agent appears, but that specific transaction is not linked to any past transaction, thus the reader does not beep. Clearly, the mapping (a_1, a_2, a_3) causes no beep either, since it has no linked transactions at all. By swapping the second and third transactions of these mappings, we obtain (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 behaviour of the reader depends on the given sequence of agents. Note that the swapping of the mapping (a_1, a_2, a_3) is identical to the original mapping because we write it in canonical form, instead of (a_1, a_3, a_2).

Conditions: Extension I and II

Finally, we introduce two conditions related to extending existing mappings. The first condition 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 transaction run by a new agent to two indistinguishable traces should not make them distinguishable. This is formally stated by the following condition:

DEFINITION 3.3.4 (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 and only 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 6. Consider a passive attacker that eavesdrops three legitimate sessions: if the attacker observes no beep, he 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), since a reader beeps only when a third tag appears in the trace. If we add a new tag to all the mappings, the equivalence of these four cases 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 is not satisfied.

For the second extension condition, consider two equivalent mappings π_1 and 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. We recall that a transaction contains an arbitrary number of sessions of a tag. Intuitively, since the attacker could not distinguish between the mappings π_1 and π_2, he cannot distinguish between their extension either, because he does not gain any new knowledge by querying twice the same agent in the last two transactions. Basically, what the attacker can do in the transaction p_{n+1} in the extended mappings to disclose information could already be done in p_n in the mappings π_1 and π_2. Thus, we require the following condition:

DEFINITION 3.3.5 (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 and only 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 7. The traces with mappings (a_1, a_1) and (a_1, a_2) are not distinguishable. However, their extensions, 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 trace makes the reader beep.

3.3.3 Equivalence results

We are now ready to demonstrate that, under the conditions stated in the previous section, all the definitions of unlinkability coincide. Moreover, we prove that, under different subsets of these conditions, the definitions of inseparability coincide as well as the strong definitions of both unlinkability and inseparability.

THEOREM 3.3.1 (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, which corresponds to the definition of game-based unlinkability. Given that we assume an infinite amount of agents, the trace where all the transactions are not linked always exists and is equivalent to all the other traces, implying strong unlinkability.

The complete proof of Theorem 3.3.1 can be found in Appendix A.2.

THEOREM 3.3.2 (Unification of inseparability). *If a protocol guarantees the following conditions:*

- *Renaming*
- *Extension II*

then it satisfies weak inseparability if and only if it satisfies strong inseparability.

Again, under these conditions, both properties require all the mappings of the same length to be equivalent. In particular they are equivalent to one where all the transactions are linked, which corresponds to the definition of strong inseparability.

The proof of Theorem 3.3.2 can be found in Appendix A.3.

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

THEOREM 3.3.3 (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 3.3.2. Indeed, the conditions Swapping and Extension I and II are not needed.

PROOF. Under the conditions Unbounded number of agents and Renaming all the strong properties coincide. An unbounded number of agents is required to guarantee the existence of the traces where all the transactions are mapped to different agents. In fact, game-based unlinkability and strong inseparability both imply the equivalence of the possible traces in a system. If the number of agents is infinite and there always exists a trace run by different agents, game based unlinkability and strong inseparability imply strong unlinkability. Instead, the condition Renaming implies that the agents cannot be distinguished. When strong unlinkability and the renaming condition hold, each trace is equivalent to a trace run by agents executing no more than one session and the equivalence to a trace executed by any single agent is always guaranteed. Thus they imply game-based unlinkability and strong inseparability. If these two conditions do not hold, it is easy to see that strong unlinkability does not coincide to game-based unlinkability and strong inseparability. This is shown in Example 3 that describes a system with two distinguishable types of tags, hence a system where the condition Renaming does not hold. In the example, the protocol satisfies strong unlinkability while it violates game-based unlinkability and strong inseparability. On the other hand, Example 2 shows that, without an unbounded number of agents, a protocol may violate strong unlinkability while satisfying game-based unlinkability and strong inseparability. □

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

COROLLARY 3.3.4. *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 3.2 coincide under the set of conditions of Section 3.3.2.

3.4 Formalisation of forward and backward privacy

In this section we discuss how our results on unlinkability can easily be adapted to define stronger privacy properties such as forward and backward privacy. These properties are 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. This corresponds to the attacker physically obtaining a tag by, for example, stealing it or simply buying it because attached to a second-hand item, and then being able to extract the content stored in the tag memory. Also, this attacker is able to extract the content stored in a tag memory. 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. These properties are fundamental in mobile systems such as RFID systems. In fact, a protocol that does not guarantee forward or backward privacy allows the attacker to trace a tag by linking it to all the past or future steps of an agent. Therefore, by tampering with a tag, an attacker may not only obtain the secret data stored in the tag, but also get to know the locations where the owner of the tag was or is going to be (by observing future transactions), even if the RFID protocol in use guarantees unlinkability.

A first definition of forward privacy in the context of RFID systems can be found in [33], while [24] 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 he disclosed.

Forward and backward privacy can be formalised in terms of traces in our model. Similarly to unlinkability, forward and backward privacy are guaranteed if the attacker cannot infer whether a specific transaction in a trace is linked, but they involve a notion of time and require stronger assumptions.

We show that, by assuming a stronger attacker and restricting the analysis to specific parts of the protocol traces, it is possible to prove forward and backward privacy.

We assume that the attacker strategy for a certain transaction discloses the secret of the agent executing it, and we call that transaction p_d, where d indicates that this transaction is the d-th in the trace. The strategy for this transaction corresponds to the attacker obtaining a tag and tampering with it in order to trace back the steps of its owner (e.g. the attacker stealing a tag). In our model it corresponds to revealing part of the mapping of a trace. Since forward privacy 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. 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 he cannot tell whether the agent that runs p_d has ever executed a session in the past.

DEFINITION 3.4.1 (Forward privacy). *Consider a protocol* T. *For each trace* $\tau \in \mathrm{T}$*, let* p_d *be a transaction such that its attacker strategy* $\sigma(p_d)$ *discloses the secret data stored in* $\pi(p_d)$*'s tag memory.*
We say that the protocol T *guarantees weak forward privacy if and only if*

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

We say that the protocol T *guarantees strong forward privacy if and only if*

$$\forall \tau \in \mathrm{T}, p_d \in Dom_\tau : \tau \vDash \neg K \bigvee_{\substack{p_i \in Dom_\tau \\ 1<i<d}} link(p_d, p_i).$$

Intuitively, these definitions imply that, if the agent whose secret has been disclosed executed transactions before p_d, then the attacker cannot link those specific transactions when weak forward privacy holds, or cannot infer the existence of a link to past transactions at all when strong forward privacy is satisfied.

Example 8. A well-known protocol that guarantees forward privacy is the OSK protocol [33] described in Section 3.3.1. To prevent the attacker from linking a secret to sessions executed before its disclosure, the protocol updates the secret after each session. To prove that this protocol guarantees strong forward privacy in our model, it is sufficient to show that each linked trace is equivalent to a trace where the transactions involved are unlinked, as

described in Definition 3.4.1. 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. According to the definition of strong forward privacy, the attacker should not be able to link the secret to the first transaction. This holds for the OSK protocol because the attacker cannot obtain any information from the current state, which corresponds to the hash $\mathbf{h}(s)$ of a_1's previous secret s, that helps him link it to the hashed value s. On the other hand, once the attacker obtains the secret, he is able to trace it in the future, i.e. he knows that the third transaction is executed by a different agent. Therefore, in our model, the trace τ must be equivalent to a trace such that the second transaction is not linked to the first transaction nor to the third one. Formally, it must be that $\tau \sim \tau_1 = ((a_1, a_2, a_1), \sigma)$ or $\tau \sim \tau_2 = ((a_1, a_2, a_3), \sigma)$ for forward privacy to hold.

With respect to unlinkability, the definitions of forward privacy limit the range of the analysis to a specific subset of transactions (those executed before p_d). However, the attacker strategy for unlinkability should never allow the attacker to disclose a tag secret by tampering with the tag, while it must for breaking forward privacy. As expected, weak forward privacy is stronger than weak unlinkability. Weak unlinkability holds when each *specific* pair of transactions are indistinguishable while weak forward privacy holds when each transaction p_d (that discloses the secret of the corresponding tag) is indistinguishable from *any* other transaction executed before p_d. Instead, in this model strong forward privacy is incomparable to strong unlinkability and game-based unlinkability. On the one hand, the attacker strategy is assumed to be stronger than the one used in the strong notions of unlinkability. On the other hand, in the definition of strong forward privacy we need to check whether each transaction is not linked to the other transactions, while the strong notions of unlinkability are more general and require the equivalence of all traces.

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 3.2.1, which can be found in Section 3.2.3). Also, weak and strong forward privacy coincide under the conditions given in Section 3.3.2, because weak forward privacy is stronger than weak unlinkability, hence, together with all the conditions, it implies the equivalence of all the traces by Theorem 3.3.1, thus also strong forward privacy.

Symmetrically to forward privacy, backward privacy checks whether the secret information is linked to *future* transactions. As in all the previous works

on this property (e.g. [24, 22]), we need to assume that, subsequently to p_d, a synchronisation session in a transaction p_s ($s > d$) takes place. A synchronisation session is a session executed among honest parties only, meaning that the attacker cannot eavesdrop it. The synchronisation session is needed to avoid the traceability of the agent, whose secret knowledge is now shared with the attacker. The session missed by the attacker gives the agent an opportunity to secretly update its internal state. In our model, after the session p_d, the attacker is able to link the transactions executed by the agent $\pi(p_d)$. We assume that he models his strategy to ignore a legitimate session that belongs to $\pi(p_d)$ during the s-th transaction. Note that in this model the attacker is very strong, since he can choose which session he is going to miss. We say that backward privacy holds if the attacker cannot infer whether p_d is linked to a specific transaction (weak form) or to any other transaction (strong form) executed after p_s.

DEFINITION 3.4.2 (Backward privacy). *Consider a protocol* T. *For each trace* $\tau \in \mathrm{T}$*, let* p_d *be the transaction such that its attacker strategy* $\sigma(p_d)$ *discloses the secret data stored in* $\pi(p_d)$*'s tag memory, and let* p_s *be the transaction such that* $s \in [d+1, \ldots, |\tau|]$*,* $\pi(p_s) = \pi(p_d)$ *and* $\sigma(p_s)$ *allows the execution of a synchronisation session.*
We say that the protocol T *guarantees weak backward privacy if and only if*

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

We say that the protocol T *guarantees strong backward privacy if and only if*

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

If some transactions are executed by the agent $\pi(p_d)$ after the restore session in p_s then the attacker should not be able to link those specific transactions when weak backward privacy holds, or to infer the existence of a link when strong backward privacy holds.

Example 9. 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 in our model this protocol guarantees strong backward privacy, it is sufficient to show that for any trace, if a

tag whose secret has been disclosed executes other transactions, then there always exists an equivalent trace where all the transactions happening after the restore session are run by other tags. 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 cannot 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 $\tau \sim \tau_2 = ((a_1, a_2, a_1, a_3), \sigma)$.

Backward privacy and unlinkability are related as forward privacy is related to unlinkability.

It is easy to prove that strong backward privacy implies weak backward privacy (it suffices to follow the technique used in the last three steps of the proof of Theorem 3.2.1, which can be found in Section 3.2.3). Also, weak and strong backward privacy coincide under the conditions given in Section 3.3.2, because weak backward privacy is stronger than weak unlinkability, hence, together with all the conditions, it implies the equivalence of all the traces by Theorem 3.3.1, thus also strong backward privacy.

Below we discuss related work before giving the conclusions.

3.5 Related work

The concept of *unlinkability*, sometimes called *untraceability* or simply *privacy*, has widely been modelled in the RFID literature in terms of games in a computational setting [18, 27, 33, 4, 34, 40]. More recently, several works focused on unlinkability properties for RFID systems in a symbolic setting [38, 39, 2, 3]. Van Deursen et al. [38] propose a definition of untraceability in a trace-based model. Arapinis et al. [2, 3] formalise protocols in the applied pi-calculus and define weak and strong unlinkability in terms of trace equivalence and observational equivalence, respectively.

Our work makes direct use of several definitions of unlinkability from the literature. As explained in detail in Section 3.2, we express the notion of weak unlinkability of [38, 3], strong unlinkability of [2, 3], and game-based definitions of [18, 27, 33, 4, 34, 32]. 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 and compared.

Epistemic models have been used in the past to formalise privacy. Similarly to our work, [25] gives general privacy definitions for a multiagent system using a modal logic of knowledge. It considers different levels of strength for anonimity, providing some probabilistic definitions as well. In [17], epistemic logic is used to give intuitive definitions of privacy in voting systems, with the applied pi calculus as an underlying model. Similarly, [21] proposes a framework in which protocols are expressed in a process language, and security properties are defined in a logic that uses both temporal and epistemic operators. The properties considered in the above works are quite different form the unlinkability properties that we consider in this thesis. Moreover, the above works are involved with the mechanics of the corresponding formalisms, while we try to completely abstract away from concrete models, considering a system as an abstract set of traces.

Several other privacy definitions have also been studied in the literature. A logic approach has been followed in [37], where an axiom system is defined to reason about anonymity, and in [26] 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, [25] and [37] study anonymity, namely a property that ensures that the identity of the agent which executes some action remains hidden from other observers.

The work in [35] gives a terminology for anonymity, unlinkability, unobservability and pseudonymity. 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 [23], [36], and [7] 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 by following the approach used in [25], that we leave as future work.

3.6 Conclusions

In this chapter we study the privacy notion of unlinkability. We capture several definitions from the literature in a simple abstract model based on epistemic logic, obtaining natural and intuitive definitions in terms of the attacker's knowledge. We also identify inseparability, a notion dual to unlinkability. Moreover, we show that these privacy definitions are different in general, but do coincide in systems satisfying a set of simple conditions. Finally, we model forward and backward privacy, two privacy properties that are strictly related to the notion of unlinkability and assume a stronger attacker. In the next chapter we translate the strongest forms of unlinkability, forward and backward privacy into an RFID model. Moreover, in Chapter 5 we demonstrate that the conditions of Section 3.3.2 are satisfied for a class of protocols in the model by Arapinis et al. [3].

4

MODELLING PRIVACY FOR RFID SYSTEMS

The privacy definitions given in Chapter 3 provide us with a full understanding of the unlinkability property and its related notions. We also argued that most protocols either guarantee the strong forms of unlinkability, forward and backward privacy, or do not guarantee even their weak forms. Hence, in this chapter we focus on the strong forms of these notions and show how to translate them into a concrete model suitable for RFID systems.

First, we provide a model in a symbolic setting to formally describe RFID protocols. Then, we focus on three privacy properties corresponding to *strong unlinkability*, *strong forward privacy* and *strong backward privacy*. In particular, we follow an approach inspired by the game-based definitions of [18, 32], which are described in 3.2.4. We do not define game-based unlinkability because it coincides with the notion of strong unlinkability in this model, due to its assumptions.

4.1 A concrete RFID model

In the context of RFID systems, the notion of *unlinkability* [18], also called untraceability [38, 2], indistinguishability [33] or simply privacy [32] has been widely studied. This research has also led to the introduction of related notions, such as *forward privacy* [33] and *backward privacy* [29]. Our main purpose is to create a model that combines together different features that belong to the existing models that captures the aforementioned notions. In particular, these features are the clarity of description provided by a formal language, a precise formalisation of privacy properties, the rigour of the

proofs, and the possibility to automatically verify protocols by means of existing tools like ProVerif [8].

We start by modelling protocols tailored to the characteristic features of RFID. The applied pi calculus provides an elegant framework for modelling protocols. It allows specifying the interaction between the various agents, using the communication primitives of the calculus. It also supports defining cryptographic operations by using a suitable equational theory.

To model RFID protocols, we first need to identify which are typically the agents involved in a protocol execution and which are their main characteristics. As we discussed in Chapter 1, RFID systems consist of several tags and readers that communicate wirelessly. The readers might communicate with a centralised backend database, typically through a secure channel. In order to achieve privacy, RFID protocols need the tags to store information (a state), either sent by a reader (e.g. some random number to randomise the outcome of the next session) or initially put in their memory (e.g. a secret which is updated at each run of the protocol using a hash chain). This is required in order to provide the same functionality of a protocol where the agents have no state, while using only the simple cryptographic primitives supported by the limited resources of the tags. Each tag typically has a state that consists of a unique secret s, shared with the backend database (and in certain systems also with the reader), and other information, which together are used to identify tags and possibly update its state after each execution.

We are now ready to formalise all these RFID aspects. First we show how to model a tag state, then a tag session and a complete tag which is initialised and offers multiple tag sessions. Finally, we discuss the synchronisation between different tag sessions and model a complete RFID system with multiple tags, reader and backend.

The state of a tag is made available through a restricted local channel w, which represents the tag memory. When a tag needs to read its state, it performs an input $w(x)$ that binds to x the content of the memory w. Similarly, a tag state M (a term that typically contains or is based on the initial secret of the tag) can be initialised or updated by performing an output, that we define as

$$St(w, M) \triangleq \overline{w}\langle M\rangle.$$

This process may be used to update a tag internal state or to make its state available after it has been read.

We define a *tag session* as $P(w, c)$, a process that communicates with the rest of the system using two channels. The restricted channel w corresponds to the tag memory, which can be used as described before. The channel c is

public and represents the wireless means of communication between a *specific* tag and *any* reader. We refer to this channel as a *tag interface*. The concept of interface plays a role similar to the one of the transactions in Section 3.1. Indeed, both an interface and a transaction model the fact that the attacker is aware that he is continuously communicating with the same tag while querying it through the same interface or within the same transaction, respectively. The intuition behind the concept of interface is that an attacker, who obtains proximity to a tag and queries it wirelessly an arbitrary number of times, knows that he accesses the same tag each time. For example, an attacker can query a tag at the entrance of a building multiple times when there is only one person in the range of his reader. Clearly, he knows that all the protocol executions are run with the same tag. On the other hand, a tag may be accessed by multiple interfaces. For instance, an attacker might query a tag at the entrance of a building (interface c_1) and a tag at the entrance of another one (interface c_2). This gives him two interfaces to freely communicate with a tag, however it could be either the same tag in both cases (tag sessions $P(w, c_1)$ and $P(w, c_2)$) or different ones (tag sessions $P(w_1, c_1)$ and $P(w_2, c_2)$).

An RFID protocol description may include cryptographic primitives. These primitives are modelled in the applied pi calculus, as usual, by using a signature and an equational theory. The example protocols analysed in this thesis only use hash functions as cryptographic primitives. We need these hash functions to be one-way (they cannot be inverted) and collision free. To this end, we model them as unary function symbols, $\mathbf{h}$ or $\mathbf{g}$, with no equations. We discuss hash functions in more detail in Section 5.3.1. Other cryptographic primitives will require proper equations. Several of them are discussed in [1].

For reasons explained in detail in Section 4.2.1, we need to introduce a synchronisation mechanism between tags in order to describe a complete protocol session. We require certain tags of interest to be *synchronised*, so that they can execute only one session at a time. Synchronisation is achieved using a channel t, which plays the role of a *synchronisation token*. A tag can only run a session (and access its state) once it obtains the token with an input on t, and must release the token with an output on t at the end of that session.

Example 10. In the OSK protocol [33], as depicted in Figure 4.1, each tag initially stores a secret s that is shared with a backend. The protocol uses two hash functions, $\mathbf{h}$ and $\mathbf{g}$. The hash function $\mathbf{h}$ is used to update the secret at each run of the protocol while $\mathbf{g}$ is used to "encrypt" the output. The tag sends $\mathbf{g}(s_i)$, where s_i is its current secret and then updates its secret using $\mathbf{h}$, giving

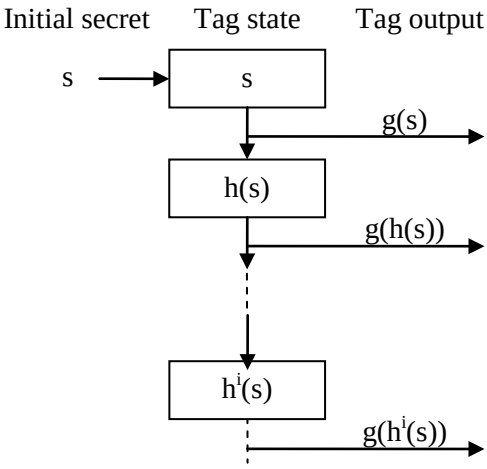

FIGURE 4.1: The OSK protocol

$s_{i+1} = \mathbf{h}(s_i)$. Basically, the combination of the two hash functions allows generating an output that looks random to an attacker. If the protocol would use $\mathbf{h}$ only, an attacker could link different sessions by applying $\mathbf{h}$ to one of the observed protocol outputs. On the other hand, by applying only $\mathbf{g}$ to a static secret, a tag protocol output would be constant and it would be trivial to find a link between sessions of the same tag. We can model a tag session of the OSK protocol in the applied pi calculus as

$$P(w, c) \triangleq c(\_).t(\_).w(x).\bar{c}\langle \mathbf{g}(x) \rangle.(\bar{t}\langle \_ \rangle \mid St(w, \mathbf{h}(x))).$$

We use $c(\_)$ and $\bar{c}\langle \_ \rangle$ to respectively denote an input and an output on channel c if the transmitted value is unimportant. Thus, the first part of the process, i.e. the input on c, simply triggers the execution of the protocol and corresponds to the reader asking "Who are you?" to the tag. Then, the input on t corresponds to giving the synchronisation token to the tag, which can now execute the protocol. Thus, the tag reads the current content of its state through an input on its state channel w and outputs the hash $\mathbf{g}$ of the current state on the public channel c. Finally, it puts both the token $\bar{t}\langle \_ \rangle$ and a new state $St(w, \mathbf{h}(x))$ back to the system. By doing so, the token is available again to the system and can be consumed by one of the synchronised tags that has been triggered by a reader.

So far we have modelled a *single tag execution*. To model a *complete tag* we need to initialise the state and run an unbounded number of executions. Let

$InitSt(w, s)$ be a process that initialises a tag state, where w is the channel used to read the state and s is the unique secret of the tag. For example, the process

$$InitSt(w, s) \triangleq \overline{n}\langle s\rangle.St(w, s).$$

registers the secret to the database through a private channel n (restricted by process modelling the system) and then stores s in the state. A complete tag is modelled as

$$Tag(c) \triangleq \nu w.\nu s.\big(InitSt(w, s) \mid !P(w, c)\big).$$

$Tag(c)$ models a tag with interface c. It can perform an unbounded number of protocol executions, starting from the initial state $InitSt(w, s)$.

A complete RFID system consists of several tags. Therefore, we define

$$ReplTag \triangleq !\,\nu c.\nu t.\overline{an}\langle c\rangle.(Tag(c) \mid \overline{t}\langle\_\rangle).$$

$ReplTag$ models an unbounded number of tags, each one with its own interface and token. A new channel c is created by each replicated copy and its name is announced on the public channel an to make it available to the environment (thus, to the attacker). The token is used to avoid that the same tag executes more than one session at a time.

We also need to define processes modelling readers and backend system in order to include all the parties involved in the protocol description. We define the processes $Reader$ and $Backend$. They model the protocol steps executed by readers and backend, similarly to the process $P(w, c)$ that specifies instead the protocol steps executed by tags. Note that in this thesis we assume that reader and backend system together have enough information to identify a tag in the system, but it is worth mentioning that in some privacy-friendly schemes not even the backend system uniquely identify tags in order to achieve stronger privacy guarantees (see [15, 16]).

All the parties (tags, readers and backend system) may need to share some private channel. We define $\widetilde{n}$ as such sequence of restricted names. For example, $\widetilde{n}$ may contain a channel name that is used to register a tag to the backend system in the initialisation phase.

Finally, a complete RFID system can be modelled as

$$\nu\widetilde{n}.(ReplTag \mid Reader \mid Backend).$$

Now that we have all the ingredients to formalise an RFID system, we are ready to define the privacy properties corresponding to strong unlinkability, forward and backward privacy, presented in the previous chapter.

4.2 Unlinkability

The idea behind our definition of unlinkability is inspired by the three-agents game unlinkability definition discussed in Section 3.2.4. In this type of game, the attacker wins if he can infer whether two given agents represent one or two different tags. In our model, we use tag interfaces to represent a public channel that connects *any* reader to a *specific* tag. Then, our definition requires that two linked interfaces (corresponding to one tag) are not distinguishable from two independent interfaces (corresponding to two tags).

In order to formally define unlinkability, we need to introduce some notation. As discussed in the previous section, we denote by $P(w, c)$ the process modelling a single tag session. The channel c is the tag interface, i.e. the public channel that it uses to communicate with the environment, and it corresponds to its interface. Since all the tags execute the same protocol, they are only distinguished by their state.

Consider $P(w, c_1)$ and $P(w, c_2)$. Since both processes are connected to the same state through the channel w, they model two executions of the *same tag*, but on *different interfaces*, e.g. a tag appearing at the entrance of building A and at the entrance of building B. Hence, we define a tag with multiple interfaces as

$$Tag(c_1, c_2) \triangleq \nu w.\nu s.\big(InitSt(w, s) \mid !P(w, c_1) \mid !P(w, c_2)\big)$$

This process represents a single tag as it restricts a single channel w and a single secret s. On the other hand, the tag has two public interfaces c_1 and c_2. Both interfaces can be accessed to execute a protocol session an unbounded number of times. Since both interfaces access the same state, interacting with any of them at a particular moment will give the same outcome. Figure 4.2 shows a graphical representation of the difference between $Tag(c_1, c_2)$ (two interfaces for a tag) and $Tag(c_1) \mid Tag(c_2)$ (two independent interfaces for two tags).

We are now ready to define unlinkability.

DEFINITION 4.2.1 (Unlinkability). *We define a system context as*

$$C[\,] = \nu\widetilde{n}.\big(\nu t.([\,] \mid \overline{t}\langle\_\rangle) \mid ReplTag \mid Reader \mid Backend\big).$$

An RFID protocol satisfies unlinkability if and only if

$$C[Tag(c_1, c_2)] \approx C[Tag(c_1) \mid Tag(c_2)].$$

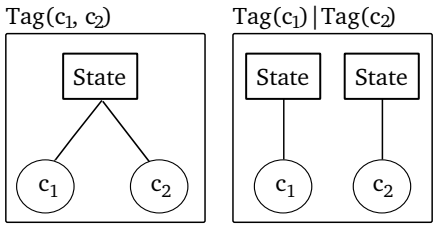

FIGURE 4.2: A tag with two interfaces and a tag with two independent interfaces

If a protocol satisfies the above definition of unlinkability, then the attacker cannot link two executions on different interfaces, since he cannot distinguish whether the interfaces correspond to the same tag or two different tags. The tags are compared in a context modelling a complete RFID system, which potentially allows the attacker to exploit information from other tags, the reader(s) or the backend database. Note that the attacker is not modelled explicitly, but he is considered part of the environment. Observational equivalence guarantees that no environment will be able to distinguish these two cases. Note also that the two tags are synchronised on a shared channel t, the reason for which is explained below.

The intuition behind Definition 4.2.1 corresponds to the one that inspired our definition of strong unlinkability (Definition 3.2.2). In fact, both these definitions require the attacker to be unable to infer whether any two transactions are linked. By Theorem 3.3.3, Definition 4.2.1 also implies the notion of strong inseparability presented in Section 3.2.5. Strong inseparability and strong unlinkability coincide when the conditions *Unbounded number of agents* and *Renaming* hold, and they are always satisfied in our model in the applied pi calculus. In fact, the number of tags in the system is unbounded and $alpha$-renaming ensures the renaming of a tag secret into another. However, we do not formally prove the equivalence of the definitions 4.2.1 and 3.2.2, and we leave it as future work.

4.2.1 Synchronisation issues

Our definition of unlinkability prevents the tags running on the interfaces c_1 and c_2 from executing more than one protocol session at a time. This is achieved by using a token t that does not allow a session to start until a tag has successfully updated its state. Without a token, the resulting definition of

unlinkability would be too strong in our opinion. To understand the type of attack that it would capture, consider a scenario in which an attacker starts communicating with a tag using the interface c_1 (e.g. at location A). In the middle of the session he stops, leaving the tag in an intermediate state, in which only part of the protocol has been executed. He then accesses a tag using a different interface c_2 (e.g. at a different location B) and tries to run the protocol again. If c_2 corresponds to the same tag then the protocol cannot start, because the tag is waiting to complete its previous session. If it is a different tag then it can start the protocol normally. Thus, the attacker can decide whether c_1, c_2 correspond to the same tag, violating unlinkability.

In practice, this type of attack is usually prevented by some property of the tag that we do not model explicitly. For example, a passive tag (without battery) will switch off when the tag is moved away from the reader, and before the attacker is able to start a session on a different interface. Even self-powered tags are often programmed to run each session for a small amount of time, and then switch off automatically.

To exclude these attacks, we restrict our attacker model by requiring that the attacker cannot execute sessions on c_1 and c_2 at the same time. A session on interface c_1 must be completed before a session on c_2 can start, and vice versa. This is achieved by using a single token to synchronise both interfaces. By using this restriction, Definition 4.2.1 ensures that the attacker cannot distinguish a tag running two sessions from two independent interfaces by *forcing* the two sessions to overlap as in the attack described above. Note that we do not need to synchronise all the tags in the system, but only the ones communicating on the "challenge" interfaces c_1 and c_2. Indeed, the tags in $ReplTag$ have their own private tokens, thus they can be executed autonomously.

We should also point out that our definition does not really depend on the synchronisation requirement. If we want to capture the attack described above, e.g. to validate a mechanism of a protocol to prevent it, we can remove all the occurrences of the channel t from our model, and still use Definition 4.2.1 to express unlinkability. In this case, $Tag(c_1) \mid Tag(c_2)$ can always run two sessions on c_1 and c_2 in parallel, since the tags are independent. However, $Tag(c_1, c_2)$ might not be able to do so: if the first session does not update the state immediately, the second will block when it tries to read it.

4.3 Forward privacy

Forward privacy only differs from unlinkability in the attacker's capabilities. Hence, we can easily adapt Definition 4.2.1 to give a further ability to the attacker: he is now able to tamper with one of the two tag interfaces he is given and retrieve the information stored in the state of the corresponding tag. After that, a tag clearly becomes traceable. However, forward privacy requires that the attacker is still unable to trace protocol sessions occurred before the tag was broken. To capture this notion, once the tag state is revealed, the interfaces c_1, c_2 can no longer be queried. Thus, the attacker can only use information obtained in the past sessions to distinguish between linked and independent interfaces. He can still, however, communicate with all the other tags of the system.

We define

$$\begin{aligned} Break^f(w) &\triangleq br(\_).t(\_).w(x).\overline{br}\langle x\rangle \\ BrTag^f(c) &\triangleq \nu w.\nu s.\big(InitSt(w,s) \mid !P(w,c) \mid Break^f(w)\big) \\ BrTag^f(c_1,c_2) &\triangleq \\ &\quad \nu w.\nu s.\big(InitSt(w,s) \mid !P(w,c_1) \mid !P(w,c_2) \mid Break^f(w)\big) \end{aligned}$$

$Break^f(w)$ is a process that models the attacker capability to reveal the state stored in the tag memory w. The attacker triggers this action by an input on the public channel br. Then, the token t is consumed, disabling both the interfaces c_1 and c_2, thus the corresponding tags are no longer available. The state content stored in w is read and sent to the attacker on the public channel br.

$BrTag^f(c)$ models a tag with a single interface c which can be tampered with. It is similar to $Tag(c)$, but allows to trigger $Break^f(w)$, a subprocess which reveals the tag state and blocks the two tag interfaces by consuming the token.

Similarly, $BrTag^f(c_1, c_2)$ models a tag with two interfaces, the state content of which can be revealed by $Break^f(w)$.

We can now define forward privacy.

DEFINITION 4.3.1 (Forward Privacy). *Let $C[\,]$ be a system context as in Definition 4.2.1. An RFID protocol satisfies forward privacy if and only if*

$$C[BrTag^f(c_1,c_2)] \approx C[BrTag^f(c_1) \mid Tag(c_2)].$$

The definition is similar to the one of unlinkability: an attacker should not be able to distinguish a tag with two interfaces from two separate tags.

The difference is the possibility to tamper with one of the tags and read its state. Also, after the state is revealed, the interfaces c_1 and c_2 can no longer be queried.

Intuitively, forward privacy is a stronger property since it gives more capabilities to the attacker. In our model, this implication can be formally proven.

PROPOSITION 4.3.1. *If a protocol satisfies forward privacy (Definition 4.3.1) then it also satisfies unlinkability (Definition 4.2.1).*

PROOF. The processes in the definition of forward privacy are the same as the ones of unlinkability, except for the subprocess $Break^f(w)$. By restricting br, the channel name used to reveal a tag state, we can easily obtain the equivalence of the unlinkability definition.

Starting from the definition of forward privacy, we restrict the name br, obtaining by structural congruence

$$\nu br.\nu\widetilde{n}.(\nu t.(BrTag^f(c_1,c_2) \mid \overline{t}\langle\_\rangle) \mid ReplTag \mid Reader \mid Backend) \approx$$
$$\nu br.\nu\widetilde{n}.(\nu t.(BrTag^f(c_1) \mid Tag(c_2) \mid \overline{t}\langle\_\rangle) \mid ReplTag \mid Reader \mid Backend).$$

Note that, br appears inside $Break^f(w)$ only, which is contained in the processes $BrTag^f(c_1,c_2)$ and $BrTag^f(c_1)$ in the left and right hand side of the equivalence, respectively. Thus, we can move the restriction νbr inside such processes, obtaining in both sides

$$\nu br.Break^f(w) = \nu br.br(\_).t(\_).w(x).\overline{br}\langle x\rangle$$

which is equivalent to the null process 0, because this process can perform no steps. Therefore, we have that

$$\nu br.BrTag^f(c_1,c_2) \approx Tag(c_1,c_2)$$
$$\nu BrTag^f(c_1) \approx Tag(c_1)$$

From these equivalences we can conclude that the equivalence implies

$$\nu\widetilde{n}.(\nu t.(Tag(c_1,c_2) \mid \overline{t}\langle\_\rangle) \mid ReplTag \mid Reader \mid Backend) \approx$$
$$\nu\widetilde{n}.(\nu t.(Tag(c_1) \mid Tag(c_2) \mid \overline{t}\langle\_\rangle) \mid ReplTag \mid Reader \mid Backend).$$

which is the definition of unlinkability. □

4.3.1 Synchronisation issues

The definition of forward privacy uses a token to synchronise the interfaces c_1, c_2. Similarly to the case of unlinkability, the use of this token is not essential for the definition. A non-synchronised version can be defined if one wants to be sensitive to synchronisation attacks. However, in the case of forward privacy, the token is also used to block the two interfaces after breaking a tag and revealing its state to the attacker. Thus, a non-synchronised version would be technically more involved than merely removing all the occurrences of t. A non-synchronised version could be achieved by using two tokens t_1 and t_2, one for each interface. Since each interface has its own token, there is no actual synchronisation between them. Then, $Break^f(w)$ can consume both tokens t_1 and t_2, blocking the two interfaces before revealing the state to the attacker.

4.4 Backward privacy

Backward privacy allows the attacker to tamper with a given tag and to retrieve its internal state, as for forward privacy. Once a tag state is disclosed, the tag becomes traceable and remains traceable as long as the attacker is given the possibility to eavesdrop on the public channel. In fact, the attacker has the same knowledge as the tag, therefore he can use the information of the next legitimate session to calculate the new tag state as the tag does. For this reason we must assume that a complete (restore) execution of the protocol takes place between legitimate users while the attacker is not eavesdropping on the wireless channel. Backward privacy is guaranteed if the attacker cannot link the tag internal state to future sessions run after the restore session.

In the definition of forward privacy (Definition 4.3.1), we block the interfaces c_1 and c_2 to be able to set our focus on the tag state and its past sessions only, so that the next sessions do not affect the validity of the definition. For backward privacy the definition should only check whether the tag state can be linked to its future sessions, rather than to its past sessions, otherwise it would imply our definition of forward privacy, although they should not be related. Also, once the attacker knows a tag state, he can certainly distinguish it from other tags at least until the restore session takes place, thus he can infer whether the interfaces are linked. However, this should not imply a violation of backward privacy.

To avoid these issues, we need to limit the attacker abilities by running a restore session immediately after the tag state disclosure, and to block the

second interface until the restore session is completed. As a result, the tag state would not provide him with any information that is useful to distinguish between the cases of linked and independent interfaces *before* the tag state disclosure.

We define:

$$
\begin{aligned}
Restore(w) &\triangleq \overline{t}\langle\_\rangle \mid P(w, rc) \mid \overline{rc}\langle\_\rangle.rc(\_).\overline{br}\langle c_2\rangle \\
Break^b(w) &\triangleq br(choice).\text{if } (choice = break) \\
&\qquad \text{then } t(\_).w(x).\overline{br}\langle x\rangle.(St(w, x) \mid Restore(w)) \\
&\qquad \text{else } \ \overline{br}\langle c_2\rangle \\
BrTag^b(c) &\triangleq \nu w.\nu s.\big(InitSt(w, s) \mid !P(w, c) \mid Break^b(w)\big) \\
BrTag^b(c_1, c_2) &\triangleq \nu w.\nu s.\big(InitSt(w, s) \mid !P(w, c_1) \mid !P(w, c_2) \mid Break^b(w)\big)
\end{aligned}
$$

The processes $BrTag^b(c)$ and $BrTag^b(c_1, c_2)$ are the same as the processes $BrTag^f(c)$ and $BrTag^f(c_1, c_2)$ in the definition of forward privacy, except for the subprocess $Break^b(w)$. The processes $BrTag^b(c)$ and $BrTag^b(c_1, c_2)$ are triggered when the attacker sends a message on the channel br. The attacker can choose between breaking the tag on c_1 to try to violate backward privacy or breaking no tag to try to violate unlinkability. In both cases, the process $Break^b(w)$ does not permanently disable the interfaces c_1 and c_2 as for forward privacy, but rather gives the attacker access to the challenge interface c_2. In the first case, the attacker sends a message "break" on the public channel br, and on br he obtains the content of the memory w (execution of the then-branch of the conditional statement). The process $Break^b$ executes the restore session before enabling the "challenge" interface c_2, otherwise the attacker can trivially infer whether the interfaces c_1 and c_2 are linked, because he can link the tag state to the sessions executed before the restore session. To achieve this, the token t is immediately consumed, blocking the interface c_1 (c_2 is already restricted by the system context, see Definition 4.4.1). Then, the process reads the memory w and gives its content to the attacker on the public channel br. The tag state is put back in w and the restore session process $Restore(w)$ is triggered. $Restore(w)$ releases the token t and allows one execution of the protocol on the private channel rc (also restricted by the system context), so the attacker cannot eavesdrop any message communicated during that session. The restore session is triggered by the subprocess $\overline{rc}\langle\_\rangle.rc(\_).\overline{br}\langle c_2\rangle$, that reads (and discard) the output of the restore session, and finally, when the restore session is terminated, releases the second interface c_2. Note that this subprocess should be part of

the $Reader$ process. For simplicity we include it in the $Restore(w)$ process, since we never explicitly model a reader. In the second case, when the attacker does not want to break the tag, he is challenged to violate unlinkability. To do this, he must send any message that does not coincide with $break$ on the public channel br. The process $Break^b(w)$ enables c_2, but the tag state in w is not revealed. In the definitions, this corresponds to the case in which the test $(choice = break)$ fails. Then the else-branch of the conditional statement is executed, and it only publishes the name of the second interface c_2. Since the $Break^b(w)$ process appears once in the processes, it cannot be triggered again to obtain the tag state. Intuitively, if the attacker immediately sends a message that is not $break$ as an input to br, he obtains exactly the processes $Tag(c)$ and $Tag(c_1, c_2)$.

DEFINITION 4.4.1 (Backward Privacy). *We define a system context as*

$$C[\,] = \nu\widetilde{n}.\nu rc.\big(\nu t.\nu c_2.([\,] \mid \overline{t}\langle\_\rangle) \mid ReplTag \mid Reader \mid Backend\big).$$

An RFID protocol satisfies backward privacy if and only if

$$C[BrTag^b(c_1, c_2)] \approx C[BrTag^b(c_1) \mid Tag(c_2)].$$

Again, the attacker is required to distinguish between these two processes, modelling linked and independent interfaces. The main difference with respect to the definitions of unlinkability and forward privacy is that the "challenge" to the attacker does not start at the first run of the protocol, but only when the attacker requests the second interface. In fact, until the attacker triggers the process $Break^b(w)$, the two sides of the equivalence model exactly the same situation, i.e. an arbitrary amount of independent tags on different interfaces running an arbitrary number of sessions each. If the attacker decides to trigger the process $Break^b(w)$ to obtain the secret of the tag using the interface c_1, he cannot use interface c_2 until the restore session is completed. If the attacker triggers the process only to reveal the name of the challenge interface, he is forbidden from tampering with the tag on c_1. This ensures that, if the protocol violates unlinkability, also our definition of backward privacy is violated. Without the else-branch, unlinkability and backward privacy would be incomparable, because the attacker knowledge would differ: while unlinkability provides the attacker with a sequence of protocol sessions from two interfaces, backward privacy would let the attacker query only one interface in the beginning and would prevent him from eavesdropping the restore session, providing the attacker with a different sequence of protocol sessions.

Similarly to forward privacy, backward privacy is stronger than unlinkability, but is unrelated to forward privacy. The first checks whether a secret can be linked with future sessions, while the second checks for links with past sessions. For example, the OSK protocol guarantees forward privacy, as we prove in Section 5.3.3, but not backward privacy. In fact, the protocol updates the secret by hashing it at every run, thus, even if the attacker misses the restore session, he can simply apply the hash function twice to the tag secret to obtain the current secret. On the other hand, backward privacy does not in general imply forward privacy. For instance, a protocol that updates a tag secret at any run by adding to it a random number sent in plain text may satisfy backward privacy, because the information that the attacker loses during the restore session is needed to calculate the new secret. But forward privacy would not necessarily be guaranteed, since the function may be reversible. The attacker could collect all the random numbers and states sent in the past sessions of a tag (before revealing the tag state), which may suffice to calculate all the past secrets and, thus, to link a tag secret to its past sessions.

PROPOSITION 4.4.1. *If a protocol satisfies backward privacy (Definition 4.4.1) then it also satisfies unlinkability (Definition 4.2.1).*

PROOF. Starting from the definition of backward privacy, we restrict the name br and output a message $noBreak$ on br, obtaining by structural congruence

$$\begin{aligned}&\nu br.\nu\widetilde{n}.\nu rc.(\nu t.\nu c_2.(BrTag^b(c_1,c_2) \mid \overline{br}\langle noBreak\rangle \mid \bar{t}\langle\_\rangle) \mid \\ &\qquad\qquad ReplTag \mid Reader \mid Backend) \approx \\ &\nu br.\nu\widetilde{n}.\nu rc.(\nu t.\nu c_2.(BrTag^b(c_1) \mid Tag(c_2) \mid \overline{br}\langle noBreak\rangle \mid \bar{t}\langle\_\rangle) \mid \\ &\qquad\qquad ReplTag \mid Reader \mid Backend).\end{aligned}$$

The process $Break^b(w)$ inside the processes $BrTag^b(c_1,c_2)$ and $BrTag^b(c_1)$ synchronises with $\overline{br}\langle noBreak\rangle$ by a silent action. Since $(noBreak \neq break)$, the process $Break^b(w)$ executes the else-branch, which enables the interface c_2. The $Break^b(w)$ process, after these transitions, reduces to 0, and by structural equivalence we have that $\nu br.0 \equiv 0$. Also, the restriction νrc can be removed since rc only appears in $Restore(w)$ in the then-branch of $Break^b(w)$, which is not chosen in this case. We obtain

$$\begin{aligned}&\nu\widetilde{n}.(\nu t.(Tag(c_1,c_2) \mid \bar{t}\langle\_\rangle) \mid ReplTag \mid Reader \mid Backend) \approx \\ &\nu\widetilde{n}.(\nu t.(Tag(c_1) \mid Tag(c_2) \mid \bar{t}\langle\_\rangle) \mid ReplTag \mid Reader \mid Backend).\end{aligned}$$

The resulting equivalence is the definition of unlinkability, thus we can conclude that it is implied by the definition of backward privacy. □

4.4.1 Synchronisation issues

The definition of backward privacy uses a token t to synchronise the interfaces c_1, c_2. As for unlinkability, a version without token can be implemented in order to capture more attacks. Note that the token here is not used as an interface-blocking mechanism. The second interface is simply blocked by a restriction in the context.

4.5 Related work

Several papers [27, 33, 4, 5, 34, 14, 40] analyse privacy properties for RFID systems, at various levels of formality. Most of them, however, define privacy in a computational setting, typically in terms of games. On the other hand, our work takes place in a symbolic setting using the formal language of the applied pi calculus. In Section 3.2.4 we briefly describe two types of indistinguishability games found in the literature [33, 18, 27, 4, 34, 32] given in a computational setting, which inspired our gagme-based definition (Definition 3.2.3) that is in a symbolic setting. The advantages of using a symbolic model are the clarity, the rigour of the proofs and the possibility of automatically verifying protocols using tools like ProVerif [8]. On the other hand, a symbolic analysis might miss attacks that exploit weaknesses of the cryptographic primitives. In the resource constrained setting of RFID systems the use of good cryptography is not trivial. Instead, several ad-hoc cryptographic solutions have been employed and some have already been proven to be insecure. Unfortunately, our model cannot capture all the details of their implementation.

The closest work to ours is the one of Arapinis et al. [2], who independently developed a definition of untraceability in the applied pi calculus, extended later in [3] where a more detailed trace-based model is provided with definitions that are similar in nature to [2]. In [2], the authors define the properties of strong and weak untraceability, as discussed in Chapter 3. The former is a strong property requiring a given RFID system to be equivalent to one where an infinite amount of tags executes only one session. This is possible because, in their model, the attacker cannot choose which tag to communicate with. Instead, he might get a response from any tag. However, the ability to query a tag several times, knowing that it is the same tag that replies each time, is very common to RFID systems. If the attacker obtains proximity to a tag, he can perform multiple queries within a very short time and be almost certain that he is communicating with the same tag. Adding this ability to

their model would make it impossible to satisfy this definition, as it is trivial to distinguish a tag that can execute multiple sessions from a tag that can only execute a single session. Weak untraceability, the second definition of [2], requires instead a system where a particular tag executes two sessions to be equivalent to a system where one of these two sessions is replaced by one belonging to a different tag. Our definition instead requires a system with a tag that executes an infinite number of sessions to be equivalent to a system where these sessions are run by two different tags. In general, [2] provides interesting alternatives to our definitions. However, our work provides several results that are outside its scope: for example the definition of forward and backward privacy, showing their relationship with unlinkability. Moreover, in Chapter 5 we study a class of protocols and analyse identification protocols from the literature.

Deursen et al. [38] also define untraceability in a symbolic setting. This work differs from ours because their model and definitions are based on traces, which are a set of actions performed by the system. While our work provides a specific framework to model RFID systems, this approach does not take into account any RFID feature, defining untraceability in terms of abstract traces that could have been generated by any identification system. More importantly, the definition of Deursen et al. is inherently weaker than ours as it corresponds to the definition of weak unlinkability given in Section 3.2. It forbids the attacker from knowing that two sessions are linked, i.e. executed by the same tag, but it does not forbid the attacker from knowing about the *existence of linked sessions*. Moreover, it does not prevent the attacker from knowing when two sessions *are not linked*. This, of course, might not be required depending on the application.

Also the works of [28] and [6] use the applied pi calculus and ProVerif to prove properties close in nature to unlinkability, such as untraceability and anonymity, for anonymous credential protocols. As in our work, they provide formal definitions of security properties and show their effectiveness analysing existing protocols. However, both the context and the properties studied in these works are substantially different from ours: they aim at providing anonymity using zero-knowledge proofs while our work focuses on privacy for RFID protocols.

Finally, some RFID protocols [22, 24] have been designed focusing on the goal of backward privacy. [22] introduces a new protocol for delegation and ownership transfer of tags without compromising the privacy of future or past owners. Their definition of backward privacy is rather informal, but it is similar in nature to ours. [24] describes a new protocol that guarantees

backward privacy based on the OSK protocol. It also provides a definition of backward privacy in terms of a game. As for the unlinkability and forward privacy games, there is a first phase where the attacker can interact with tags and readers. He chooses two tags and he is given one of them after a synchronisation phase. Then, he wins the game if he is able to guess which tag he was given. This work differs from ours because, while we let the attacker break a tag and get its secret data, [24] allows the attacker to discover the information of all the tags.

4.6 Conclusions

In this chapter we define a model in the applied pi calculus to study privacy properties like unlinkability, forward and backward privacy. Existing definitions are compared in Chapter 3, where we show that they offer different privacy guarantees. Here we define more concrete versions of the strong forms of those privacy definitions. In fact, the definition of unlinkability in the applied pi calculus is inspired by the game-based definition of Section 3.2.4, which we demonstrate to be equivalent to the notion of strong unlinkability. Also, the definitions of forward and backward privacy correspond to the strongest definitions of Chapter 3.

Finally, we show the relationship between all these properties, formally proving that the notions of forward and backward privacy both imply unlinkability.

In the next chapter we study a class of protocols and demonstrate that verifying privacy definitions for this class is simpler than in the general case. In fact we identify some necessary and sufficient conditions that are easier to prove and can be verified by automatic tools such as ProVerif for finite traces.

5

SINGLE-STEP PROTOCOLS

In previous chapters we studied the privacy properties of unlinkability, forward and backward privacy. In this chapter we study how the privacy definitions given in Chapters 3 and 4 apply to a class of RFID protocols which we call *single-step protocols*. More concretely, we describe this class in the model presented in [3], which we instantiate in our abstract model, and we study its properties.

This chapter is structured as follows. First we describe single-step protocols and formalise them in the applied pi calculus. Then, in Section 5.2 we instantiate our abstract trace-based model defined in Chapter 3 using the model in the applied pi calculus of Arapinis et al. [3]. We show that all the conditions described in Section 3.3 for the abstract model are satisfied in this instantiation, thus the privacy properties presented in Chapter 3 always coincide. Section 5.3 introduces necessary and sufficient conditions for single-step protocols under which the privacy properties described in Chapter 4 for our RFID model hold. We also study some protocols from the literature [33, 41, 29] and variations thereof, and prove that they satisfy some of the conditions described in Session 5.3. Section 5.4 provides conclusions of this chapter.

5.1 Single-step protocols in the applied pi calculus

Single-step protocols are RFID protocols that consist of a single message from a tag to a reader. The reader only activates the tag, so the tag can send its identification message. This message should suffice to allow the backend system to identify the tag. We study this class in full generality giving necessary and

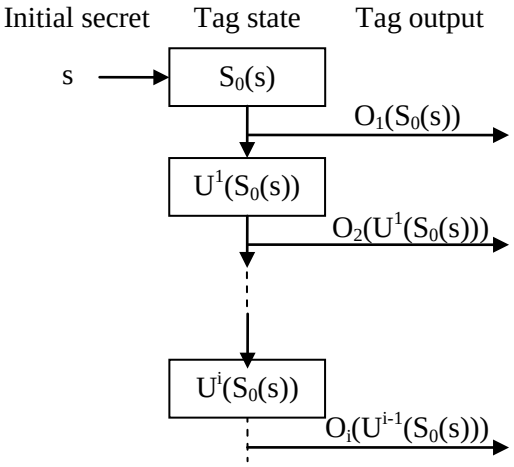

FIGURE 5.1: A single-step protocol

sufficient conditions for unlinkability, forward and backward privacy. In particular, we define the class of single-step protocols in the applied pi calculus in this section, and we investigate its properties in the remainder of this chapter.

Each protocol in this class starts with a reader activating a tag. However the reader does not send any information. Then, the tag reads its state, constructs a message (possibly containing fresh nonces) and sends it to the reader. Depending on the system the reader may forward it to the backend or directly identify the tag. Finally, the tag updates its state and the session ends.

Although this class of protocols is quite restricted, it is still of great interest for several reasons. First, the simplicity of single-step protocols helps us to understand the properties that are required to ensure privacy, making it easier to understand more complex protocols. Second, as we will see in the Section 5.3.3, some published protocols fall into this class. Moreover, it should be possible to build on these results, extending them to a wider class of protocols, e.g. protocols that allow a tag to receive inputs from the reader. Finally, having general results for a class of protocols allows us to experiment with protocol design. For example, our results could allow a designer to try a single-step protocol and quickly check whether it satisfies privacy.

A single step protocol is defined based on three terms:

- $S_0(y)$: the initial state of a tag with tag secret y;
- $U(x)$: the next state resulting from running the protocol in state x;

- $O(x)$: the output produced by a tag in state x.

The output and state update may use names that are specific for that run of the protocol, e.g. a random nonce. This is achieved by having a sequence of names $\tilde{\rho}$ that are restricted in the protocol session process. Each run of the protocol will generate new copies of the names $\tilde{\rho}$. We use $\tilde{\rho}_i$ to denote the sequence of names generated in the first i runs of the protocol. We use $O_i(x)$, $U_i(x)$ to denote the output and update terms using the names generated in the i-th protocol run, i.e. with the names in $\tilde{\rho}$ renamed to the corresponding ones in $\tilde{\rho}_i$. Finally, we define $U^i(M)$ as $U_i(U_{i-1}(\ldots U_1(M)))$. Thus, the output of the i-th session is $\nu s, \widetilde{\rho}_i.O_i(U^{i-1}(S_0))$.

Example 11. Consider a protocol that generates a new random number n, outputs it together with the hash of its current state added to n and updates its state to the sum of its state, n and m, with m a publicly known (fixed) value. This can be modelled by taking

- $O(x) = (n, h(x + n))$;
- $U(x) = (x + n + m)$;
- $\tilde{\rho} = n$.

Here $\tilde{\rho}_i$ is $n_1, \ldots, n_i$, $O_i(x)$ is $(n_i, h(x + n_i))$, $U_i(x)$ is $(x + n_i + m)$.

A generic single-step protocol is shown in Figure 5.1. Each tag starts with an initial state $S_0(s)$, where s is the tag initial secret.

We model single-step protocols in the applied pi calculus as described in Section 4.1. To properly define a $Tag(c)$ process, we need to instantiate its $P(w, c)$ and $InitSt(w, s)$ sub-processes.

DEFINITION 5.1.1. *The class of single-step protocols consists of all protocols of the form*

$$P(w, c) \triangleq c(\_).t(\_).w(x).\nu\widetilde{\rho}.\overline{c}\langle O(x)\rangle.(\overline{t}\langle\_\rangle \mid St(w, U(x)))$$
$$InitSt(w, s) \triangleq St(w, S_0(s))$$

for some terms $O(x), U(x), S_0(s)$ and names $\widetilde{\rho}$ s.t. $s \notin fn(O(x)) \cup fn(U(x))$.

$InitSt(w, s)$ simply initialises the state with $S_0(s)$. $P(w, c)$ starts with an input on c, which simply triggers the beginning of the session. Then the tag consumes the synchronisation token t, reads its state in x, generates new

names in $\widetilde{\rho}$, and outputs $O(x)$ on the channel c. Finally, it releases the token and updates the state with $U(x)$.

A complete tag is first initialised with a state $S_0(s)$ and can execute an unbounded number of sessions.

$$Tag \triangleq \nu s.\nu w.(InitSt(w,s) \mid !P(w,c))$$

Note that from now on we use S_0 for $S_0(s)$, since the secret s is clear from the context.

For this class of protocols, the readers are completely passive, they only trigger the tag without sending any data to it. Since c is a public channel, the tag can be triggered by any process in parallel to it, thus we can completely avoid specifying the reader. To complete the processes instantiation of an RFID system (Section 4.1), we set $Reader = Backend = 0$ and $\widetilde{n} = \varepsilon$. Thus, a complete system P consists of an unbounded number of tags and can be defined as

$$P \triangleq !Tag$$

5.2 Instantiating our abstract model

In Section 3.3 we showed that, under some reasonable conditions, all the unlinkability and inseparability definitions coincide. In this section, we show that these conditions are indeed satisfied by the class of single-step protocols. To this end, we use our definition of single-step protocol and the model of [3], which provides a concrete set of traces and an equivalence relation between them.

We start by describing how we obtain a trace from an RFID system P (the one introduced in the previous section) in the model of [3]. A 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

$$tr = P \stackrel{\alpha_1}{\Longrightarrow} P_1 \stackrel{\alpha_2}{\Longrightarrow} P_2 \stackrel{\alpha_3}{\Longrightarrow} \ldots \stackrel{\alpha_n}{\Longrightarrow} P_n.$$

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. Formally:

DEFINITION 5.2.1. *Let*

$$tr_A = A_0 \stackrel{\alpha_1}{\Longrightarrow} A_1 \stackrel{\alpha_2}{\Longrightarrow} \ldots \stackrel{\alpha_n}{\Longrightarrow} A_n$$

and

$$tr_B = B_0 \stackrel{\alpha_1}{\Longrightarrow} B_1 \stackrel{\alpha_2}{\Longrightarrow} \ldots \stackrel{\alpha_n}{\Longrightarrow} B_n.$$

The traces tr_A and tr_B are equivalent denoted $tr_A \sim_{tr} tr_B$ if $A_i \approx_s B_i$ for all i.

We recall that our abstract model (see Definition 3.1.1) requires

- a set of agents A;
- a set of strategies Σ such that a strategy σ together with a mapping π from a set of mappings Π form an abstract trace $\tau = (\pi, \sigma)$;
- an equivalence relation $\sim$ between abstract traces.

The agents $A = \{a_i \mid i \in \mathbb{N}\}$ 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 he will run, and how many protocol executions he will trigger in each transaction. The number of executions in a transaction may depend, for example, on the output obtained in the previous transaction. For simplicity and without loss of generality, we assume that the strategies are not adaptive. Thus, a strategy σ is a sequence $\sigma = (\sigma_1, \ldots, \sigma_k)$ such that k is the transaction index, and σ_i the number of executions that the attacker triggers in the i-th transaction. A mapping π determines which tag will participate in each transaction, for example $\pi(2) = a_3$ means that a_3 is the tag involved in the second transaction.

Given an abstract trace, i.e. a strategy σ and a mapping π, we can define a unique concrete trace $tr(\pi, \sigma)$ starting from P. In this trace, the tag $\pi(1)$ is first spawned and runs σ_1 executions. Then, the tag $\pi(2)$ is spawned (unless $\pi(1) = \pi(2)$, meaning that it had already been spawned for the first transaction) and runs σ_2 executions, and so on. Finally, we define trace equivalence as follows:

$$(\pi, \sigma) \sim (\pi', \sigma) \quad \text{if and only if} \quad tr(\pi, \sigma) \sim_{tr} tr(\pi', \sigma)$$

Now that we have a concrete definition of the set of agents, the set of traces and the trace equivalence, which together form a concrete trace model, we can show that single-step protocols satisfy all the conditions of Section 3.3.2.

THEOREM 5.2.1. *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.

The proof can be found in Appendix A.4.

5.3 Privacy properties for single-step protocols

We now provide conditions for single-step protocols, that are both necessary and sufficient to satisfy the privacy properties of Chapter 4 (unlinkability, forward and backward privacy). The conditions are based on the notion of frame independence that we develop in Section 5.3.1 before giving the conditions in Section 5.3.2. Finally, Section 5.3.3 provides some examples of single-step protocols from the literature. For each example we use the conditions to prove whether they guarantee the privacy definitions of unlinkability, forward and backward privacy.

5.3.1 Frame independence

In this section we discuss a notion that we call *frame independence*. As shown in Section 5.3.2, this concept can be used to give necessary and sufficient conditions for unlinkability, forward and backward privacy for a generic family of protocols. However, the notion itself is generic and we develop it on its own. We also discuss some results concerning frame independence of frames containing hash functions.

Consider two frames φ_1, φ_2, each containing some free names. Both frames, where some names $\widetilde{s}$ are restricted, are given to the attacker. The attacker's goal is to detect whether the names in $\widetilde{s}$ are assigned the same value in both frames. If the attacker is able to distinguish these two cases, we say that φ_1, φ_2 are *dependent* with respect to $\widetilde{s}$, otherwise they are *independent*. Intuitively, two frames being dependent means that the attacker can link them to the same owner due to the use of the same restricted names $\widetilde{s}$.

Example 12. Consider the frames

$$\{^{\mathbf{h}(s)}/_{x}\} \qquad\qquad \{^{\mathbf{h}(\mathbf{h}(s))}/_{y}\}$$

where $\mathbf{h}$ is a hash function. Even if s is restricted, we can still decide whether s contains the same value in both frames. Namely, we can apply $\mathbf{h}$ to x and then test whether $\mathbf{h}(x) =_{\mathrm{E}} y$. However, a shared secret does not always imply a dependence. Consider for example the frames

$$\{^{\mathbf{h}(s)}/_{x}\} \qquad\qquad \nu r.\{^{\mathbf{h}(s,r)}/_{y}\}$$

Given that r is restricted in the second frame, if s is restricted, it is impossible to tell whether $\mathbf{h}(s)$ and $\mathbf{h}(s, r)$ contain the same secret s.

We formalise this idea in the following definition.

DEFINITION 5.3.1. *Let φ_1 and φ_2 be closed frames with $\mathrm{dom}(\varphi_1) \cap \mathrm{dom}(\varphi_2) = \emptyset$. We say that φ_1 is independent of φ_2 with respect to the names $\widetilde{s}$, written $\varphi_1 \perp_{\widetilde{s}} \varphi_2$, if and only if*

$$\nu\widetilde{s}.(\varphi_1 \mid \varphi_2) \approx_s \nu\widetilde{s}.\varphi_1 \mid \nu\widetilde{s}.\varphi_2.$$

Intuitively, this definition states that φ_1, φ_2 are independent with respect to secrets $\widetilde{s}$ if and only if their composition under the same restricted names $\widetilde{s}$ is statically equivalent to their parallel execution, where each frame has its own restricted names. In other words, assigning the same value to names in $\widetilde{s}$ in both frames looks the same to the attacker as assigning different values in each frame. Clearly, when the sets of free names in φ_1 and φ_2 are disjoint, the frames are trivially independent with respect to those names. This definition is vaguely reminiscent of the independence of probability events, $p(A \wedge B) = p(A)p(B)$, which requires that the joint distribution (in our case composition of frames with shared names) is obtained by simply multiplying the marginal distributions (in our case putting in parallel the two frames).

Going back to our previous examples, we have that

$$\begin{aligned}\nu s.\{^{\mathbf{h}(s)}/_{x}, {}^{\mathbf{h}(\mathbf{h}(s))}/_{y}\} &\not\approx_{\mathrm{s}} \nu s.\nu s'.\{^{\mathbf{h}(s)}/_{x}, {}^{\mathbf{h}(\mathbf{h}(s'))}/_{y}\} \\ &\equiv \nu s.\{^{\mathbf{h}(s)}/_{x}\} \mid \nu s.\{^{\mathbf{h}(\mathbf{h}(s))}/_{y}\}\end{aligned}$$

thus

$$\{^{\mathbf{h}(s)}/_{x}\} \not\perp_s \{^{\mathbf{h}(\mathbf{h}(s))}/_{y}\}.$$

On the other hand

$$\nu s.\nu r.\{^{\mathbf{h}(s)}/_{x}, ^{\mathbf{h}(s,r)}/_{y}\} \approx_{\mathrm{s}} \nu s.\nu s'.\nu r.\{^{\mathbf{h}(s)}/_{x}, ^{\mathbf{h}(s',r)}/_{y}\}$$

thus

$$\{^{\mathbf{h}(s)}/_{x}\} \perp_{s} \nu r.\{^{\mathbf{h}(s,r)}/_{y}\}$$

as expected.

We now state some basic properties of frame independence.

PROPOSITION 5.3.1. *Let $\varphi_1, \varphi_2, \psi$ be closed frames such that $\varphi_1 \perp_{\widetilde{s}} \psi$. If one of the following holds*

1. $\varphi_2 \approx_s \varphi_1$
2. $\varphi_2 \equiv \varphi_1 \mid \varphi_1'$ *for some* φ_1' *with* $\{\widetilde{s}\} \cap \mathbf{fn}(\varphi_1') = \emptyset$ *and* $\mathrm{dom}(\varphi_1') \cap \mathrm{dom}(\psi) = \emptyset$
3. $\varphi_2 \equiv \nu u.\varphi_1$ *for some* $u \notin \mathbf{fv}(\psi) \cup \mathbf{fn}(\psi)$

then $\varphi_2 \perp_{\widetilde{s}} \psi$.

PROOF. From $\varphi_1 \perp_{\widetilde{s}} \psi$ by definition we get $\nu\widetilde{s}.(\varphi_1 \mid \psi) \approx_{\mathrm{s}} \nu\widetilde{s}.\varphi_1 \mid \nu\widetilde{s}.\psi$.

1. Since $\approx_{\mathrm{s}}$ is closed under application of closing evaluation contexts, we have

   $$\nu\widetilde{s}.(\varphi_2 \mid \psi) \approx_{\mathrm{s}} \nu\widetilde{s}.(\varphi_1 \mid \psi) \approx_{\mathrm{s}} \nu\widetilde{s}.\varphi_1 \mid \nu\widetilde{s}.\psi \approx_{\mathrm{s}} \nu\widetilde{s}.\varphi_2 \mid \nu\widetilde{s}.\psi$$

   thus $\varphi_2 \perp_{\widetilde{s}} \psi$.

2. Note that $\varphi_2 = \varphi_1 \mid \varphi_1'$ is closed, so by congruence we get $\nu\widetilde{s}.(\varphi_1 \mid \psi) \mid \varphi_1' \approx_{\mathrm{s}} \nu\widetilde{s}.\varphi_1 \mid \nu\widetilde{s}.\psi \mid \varphi_1'$. Since $\{s\} \cap \mathbf{fn}(\varphi_1') = \emptyset$ and $\approx_{\mathrm{s}}$ is closed under $\equiv$, we get $\nu\widetilde{s}.(\varphi_1 \mid \varphi_1' \mid \psi) \approx_{\mathrm{s}} \nu\widetilde{s}.(\varphi_1 \mid \varphi_1') \mid \nu\widetilde{s}.\psi$ and thus $\nu\widetilde{s}.(\varphi_2 \mid \psi) \approx_{\mathrm{s}} \nu\widetilde{s}.\varphi_2 \mid \nu\widetilde{s}.\psi$ which implies $\varphi_2 \perp_{\widetilde{s}} \psi$

3. Again by simple application of the congruence property.

Thus all the conditions imply $\varphi_2 \perp_{\widetilde{s}} \psi$. □

The second part of the above proposition says that we can extend a frame φ_1 while preserving independence. An extended frame φ_2 adds new terms to the ones exported by φ_1, but these terms can be constructed from φ_1. The new terms can contain restricted names of φ_1, but only if they are contained in some variable already present in φ_1. For example,

$$\varphi_2 = \nu n.\{^{\mathbf{f}(n)}/_{x}, {}^{\mathbf{g}(\mathbf{f}(n))}/_{y}\}$$

is an extension of $\varphi_1 = \nu n.\{^{\mathbf{f}(n)}/_{x}\}$ since

$$\varphi_2 \equiv \varphi_1 \mid \{^{\mathbf{g}(x)}/_{y}\}.$$

Reciprocally, the third part says that we can restrict φ_1 to a subset of the exported terms, while preserving independence. Moreover, we can restrict some free names of φ_1, provided that they are not free in ψ, and still preserve independence.

Reasoning about hash functions

One-way hash functions are commonly used in RFID protocols. Indeed, the protocols from the literature analysed later on in Section 5.3.3 use solely hash functions as cryptographic primitives. In this section we give some results concerning the independence of frames using hash functions.

In the applied pi calculus, a hash function is typically modelled by a unary function symbol, e.g. $\mathbf{h}$ with no equational axioms. Still, hash functions can be combined with other cryptographic primitives with their own axioms. So we might want to use an equational theory with an arbitrary set of axioms, the only condition being that they should not contain $\mathbf{h}$. To provide generic results about hash functions, we should find properties that hold under any such theory. Seeking even more generality, we can pose the question of what it means for the function symbol $\mathbf{h}$ to be a hash function in an arbitrary equational theory $=_{\mathrm{E}}$, independently from how $=_{\mathrm{E}}$ is generated. We begin by giving a definition of hash function that we use later on in proofs.

We fix an equational theory $=_{\mathrm{E}}$ containing a unary function $\mathbf{h}$. We also assume the typical function symbols $\mathbf{pair}, \mathbf{fst}, \mathbf{snd}$ for creating and decomposing pairs, with the usual axioms $\mathbf{fst}(\mathbf{pair}(x, y)) =_{\mathrm{E}} x$, $\mathbf{snd}(\mathbf{pair}(x, y)) =_{\mathrm{E}} y$. Since pairs are used extensively, we simplify the notation by writing (x, y) instead of $\mathbf{pair}(x, y)$, and $\mathbf{h}(x, y)$ instead of $\mathbf{h}(\mathbf{pair}(x, y))$.

Let M, K, L be terms. We define $M[^{L}/_{\mathbf{h}(=K)}]$ as the term obtained from M by substituting L for all subterms of the form $\mathbf{h}(K')$ with $K =_{\mathrm{E}} K'$. More

precisely, $M[^{L}/_{\mathbf{h}(=K)}]$ is recursively defined as

$$\begin{aligned} x[^{L}/_{\mathbf{h}(=K)}] &= x \\ n[^{L}/_{\mathbf{h}(=K)}] &= n \\ \mathbf{f}(\widetilde{M})[^{L}/_{\mathbf{h}(=K)}] &= \mathbf{f}(\widetilde{M}[^{L}/_{\mathbf{h}(=K)}]), \mathbf{f} \neq \mathbf{h} \\ \mathbf{h}(M)[^{L}/_{\mathbf{h}(=K)}] &= \begin{cases} L & \text{if } M =_{\mathrm{E}} K \\ \mathbf{h}(M[^{L}/_{\mathbf{h}(=K)}]) & \text{if } M \neq_{\mathrm{E}} K \end{cases} \end{aligned}$$

Note that this is different from $M[^{L}/_{\mathbf{h}(K)}]$ which replaces only exact occurrences of $\mathbf{h}(K)$.

DEFINITION 5.3.2. *We say that a unary function* $\mathbf{h}$ *is a* one-way hash function *with respect to* $=_{\mathrm{E}}$ *if and only if*

$$K =_{\mathrm{E}} L \quad \Rightarrow \quad K[^{x}/_{\mathbf{h}(=M)}] =_{\mathrm{E}} L[^{x}/_{\mathbf{h}(=M)}]$$

for all terms K, L, M *and variables* x.

The idea behind this definition is that $\mathbf{h}(M)$ can appear in an equation $K =_{\mathrm{E}} L$ only as a "generic term". The equation should not depend on the fact that $\mathbf{h}(M)$ is a hash value. Thus, we require that replacing all occurrences of $\mathbf{h}(M)$ by a variable x gives us an equation that is still valid in $=_{\mathrm{E}}$. For example, assuming that $\mathbf{h}$ is a hash function, the equation

$$\mathbf{dec}(\mathbf{enc}(\mathbf{h}(n), k), k) = \mathbf{h}(n)$$

is allowed, provided that

$$\mathbf{dec}(\mathbf{enc}(x, k), k) =_{\mathrm{E}} x$$

holds in general. On the other hand, the equation

$$\mathbf{g}(\mathbf{h}(m)) = m$$

violates Definition 5.3.2 for $\mathbf{g}$ unless it is the constant function $\mathbf{g}(x) =_{\mathrm{E}} m$.

The following lemma shows that our definition of hash function behaves as expected.

LEMMA 5.3.2. *Let* $\mathbf{h}$ *be a hash function (Definition 5.3.2) and assume that* $=_{\mathrm{E}}$ *does not equate all terms. Then*

1. $\mathbf{h}$ *is* collision-free*, that is* $\mathbf{h}(M) =_{\mathrm{E}} \mathbf{h}(N) \Rightarrow M =_{\mathrm{E}} N$.

2. *If* $\mathbf{h}(M) =_{\mathrm{E}} N$ *then there exists* $\mathbf{h}(N') \trianglelefteq N$ *s.t.* $N' =_{\mathrm{E}} M$.

3. *There is no function that inverts* $\mathbf{h}$*, i.e. there is no function* $\mathbf{invh}$ *such that* $\mathbf{invh}(\mathbf{h}(x)) =_{\mathrm{E}} x$.

4. *There is no function that checks whether a value is a hash value, i.e.* $\mathbf{checkh}(M) =_{\mathrm{E}} \mathbf{ok}$ *if and only if* $M =_{\mathrm{E}} \mathbf{h}(M')$.

PROOF.

1. Assume $\mathbf{h}(M) =_{\mathrm{E}} \mathbf{h}(N)$ and $M \neq_{\mathrm{E}} N$. Then by Definition 5.3.2 we get $\mathbf{h}(M)[^{x}/_{\mathbf{h}(=M)}] =_{\mathrm{E}} \mathbf{h}(N)[^{x}/_{\mathbf{h}(=M)}]$ thus $x =_{\mathrm{E}} \mathbf{h}(N')$ where $N' =_{\mathrm{E}} N[^{x}/_{\mathbf{h}(=M)}]$. Finally by substituting $\mathbf{h}(N')$ we get $x =_{\mathrm{E}} y$ which is a contradiction.

2. If no such term exists then by substituting $\mathbf{h}(M)$ we get $x =_{\mathrm{E}} N$ which implies $x =_{\mathrm{E}} y$.

3. Such equation would imply that $\mathbf{invh}(z) =_{\mathrm{E}} x$ which again implies $x =_{\mathrm{E}} y$.

4. Such equation would imply that $\mathbf{checkh}(y) =_{\mathrm{E}} \mathbf{ok}$ for all y, not just those of the form $\mathbf{h}(x)$, therefore it would not correspond to a test for hash functions.

□

Definition 5.3.2 can be seen as a generalisation of the traditional definition of hash functions (a function symbol with no axioms) to arbitrary equational theories. It should be noted that Definition 5.3.2 is too strong, forbidding some equations that do hold for real hash functions. For example, if $\mathbf{g}$ is a hash function, then $\mathbf{h}(x) = \mathbf{g}(\mathbf{g}(x))$ is also a hash function, yet the above equation violates Definition 5.3.2. This problem, however, is also true for the traditional definition of hash functions, and can be circumvented by dropping $\mathbf{h}$ completely and always using $\mathbf{g}(\mathbf{g}(M))$ in place of $\mathbf{h}(M)$.

We now introduce the concept of *derivability* of a term, which is used in the following theorem and propositions.

DEFINITION 5.3.3. *Let* φ *be a frame in canonical form. The term* M *is* derivable *from* φ *if and only if* $M =_{\mathrm{E}} M'\varphi$ *for some* M' *such that* $\mathbf{fn}(M') \cap \mathbf{bn}(\varphi) = \emptyset$.

This definition says that a term M is derivable from a frame when the application of the frame to some term M', which does not contain free names that are bound in the frame, gives a term that is equivalent to M. In other words, M is derivable from a frame when the latter contains the knowledge needed to build M from another term.

We are now ready to state results concerning the derivability of terms containing hash functions.

THEOREM 5.3.3. *Let $\nu\widetilde{n}.\varphi$ be a frame. If $\mathbf{h}$ is a hash function, $\mathbf{h}(M)$ is not derivable from $\nu\widetilde{n}.\varphi$, and $\mathbf{h}(M) \not\preceq \varphi(y)$ for all $y \in \mathrm{dom}(\varphi)$, then*

$$\nu\widetilde{n}.(\varphi \mid \{^{\mathbf{h}(M)}/_{x}\}) \approx_s \nu\widetilde{n}.\varphi \mid \nu r.\{^{r}/_{x}\}$$

Intuitively, if a frame does not allow to derive some hash value, then it will look as a random value when put in parallel to such frame, i.e. an attacker sees no difference between a random value and the hash $\{^{\mathbf{h}(M)}/_{x}\}$.

PROOF. Let

$$\psi_1 = \{^{\mathbf{h}(M)}/_{x}\} \qquad \varphi_1 = \nu\widetilde{n}.(\varphi \mid \psi_1)$$
$$\psi_2 = \nu r.\{^{r}/_{x}\} \qquad \varphi_2 = \nu\widetilde{n}.(\varphi \mid \psi_2)$$

To prove the theorem, we first need to show that $K\varphi_1 =_{\mathrm{E}} L\varphi_1 \Leftrightarrow K\varphi =_{\mathrm{E}} L\varphi$ (1.) and $K\varphi_2 =_{\mathrm{E}} L\varphi_2 \Leftrightarrow K\varphi =_{\mathrm{E}} L\varphi$ (2.).

(1. $\Leftarrow$) and (2. $\Leftarrow$) are trivial. For (1. $\Leftarrow$) we have that $K\varphi_1 =_{\mathrm{E}} L\varphi_1$ is $K\varphi =_{\mathrm{E}} L\varphi$ after a specific variable is bound (similarly for (2. $\Leftarrow$)). The restrictions $\nu\widetilde{n}$ may be added using structural equivalence. To prove (1. $\Rightarrow$) and (2. $\Rightarrow$) we first demonstrate two properties. The first is

$$K\varphi\psi_1[^{x}/_{\mathbf{h}(=M)}] = K\varphi \qquad \forall K \text{ with } \{\widetilde{n}\} \cap \mathbf{fn}(K) = \emptyset. \tag{5.1}$$

In other words, the only terms with form $\mathbf{h}(= M)$ in $K\varphi\psi_1$ are those substituted for x by ψ_1. We assume that M has no subterm $\mathbf{h}(M')$ s.t. $M' =_{\mathrm{E}} M$, that is

$$M[^{x}/_{\mathbf{h}(=M)}] = M. \tag{5.2}$$

Otherwise we can replace the term $\mathbf{h}(M)$ in the Theorem by the smallest such subterm $\mathbf{h}(M')$, i.e. a term that is the hash of some term M' which does not contain any subterm $\mathbf{h}(= M')$. The proof is by structural induction on K.

- Case $K = x$: we have $K\varphi = x$, $K\varphi\psi_1 = \mathbf{h}(M)$.

- Case $K = y \in \mathrm{dom}(\varphi)$: we have $K\varphi = \varphi(y)$, and (5.1) follows from the fact that $\mathbf{h}(M) \not\sqsubseteq \varphi(y)$ (hypothesis).

- Cases $K = z \notin \mathrm{dom}(\varphi)$, $K = n$, $K = \mathbf{f}(\widetilde{K'}), \mathbf{f} \neq \mathbf{h}$: trivial.

- Case $K = \mathbf{h}(K')$ with $K'\varphi\psi_1 \neq_{\mathrm{E}} M$: we have that $K\varphi\psi_1[^x/_{\mathbf{h}(=M)}] = \mathbf{h}(K'\varphi\psi_1[^x/_{\mathbf{h}(=M)}])$ and the result follows from the induction hypothesis.

- Case $K = \mathbf{h}(K')$ with $K'\varphi\psi_1 =_{\mathrm{E}} M$: we show that this case is impossible. We have that $K'\varphi\psi_1[^x/_{\mathbf{h}(=M)}] =_{\mathrm{E}} M[^x/_{\mathbf{h}(=M)}]$ and from the induction hypothesis and (5.2) we get $K'\varphi =_{\mathrm{E}} M$ and hence $K\varphi =_{\mathrm{E}} \mathbf{h}(M)$. Since $\mathbf{fn}(K) \cap \{\widetilde{n}\} = \emptyset$ this means that $\mathbf{h}(M)$ is derivable from φ which is a contradiction.

Let K, L be terms such that $\{\widetilde{n}, r\} \cap (\mathbf{fn}(K) \cup \mathbf{fn}(L)) = \emptyset$. Assuming $K\varphi_1 =_{\mathrm{E}} L\varphi_1$, we can use the definition of hash function (Definition 5.3.2) to obtain $K\varphi_1[^x/_{\mathbf{h}(=M)}] =_{\mathrm{E}} L\varphi_1[^x/_{\mathbf{h}(=M)}]$. By (5.1) and (1. $\Leftarrow$) we can conclude that $K\varphi_1 =_{\mathrm{E}} L\varphi_1 \Leftrightarrow K\varphi =_{\mathrm{E}} L\varphi$, which is (1.). The second property is

$$K\varphi\psi_2[^x/_r] = K\varphi \qquad \forall K \text{ with } \{r\} \cap \mathbf{fn}(K) = \emptyset\text{"}'. \tag{5.3}$$

It follows from the assumption that $r \notin \mathbf{fn}(\varphi)$ and the fact that $=_{\mathrm{E}}$ is closed under substitution of variables for names. It states that the application of a substitution $[^x/_r]$ to $K\varphi\psi_2[^x/_r]$ does not affect $K\varphi$. By (5.3) and (2. $\Leftarrow$) we can conclude that $K\varphi_2 =_{\mathrm{E}} L\varphi_2 \Leftrightarrow K\varphi =_{\mathrm{E}} L\varphi$, which is (2.). From (1.) and (2.) we obtain

$$K\varphi_1 =_{\mathrm{E}} L\varphi_1 \quad \Leftrightarrow \quad K\varphi =_{\mathrm{E}} L\varphi \quad \Leftrightarrow \quad K\varphi_2 =_{\mathrm{E}} L\varphi_2.$$

Hence, we have that $K\varphi_1 =_{\mathrm{E}} L\varphi_1 \Leftrightarrow K\varphi_2 =_{\mathrm{E}} L\varphi_2$. By definition of static equivalence we can conclude that

$$\nu\widetilde{n}.(\varphi \mid \{^{\mathbf{h}(M)}/_x\}) \approx_{\mathrm{s}} \nu\widetilde{n}.\varphi \mid \nu r.\{^r/_x\}.$$

□

Using this theorem we can show that, under certain assumptions, the independence of two frames is preserved when we extend them by adding a substitution containing a hash, as stated by the following proposition.

PROPOSITION 5.3.4. *Let φ_1, φ_2 be frames such that $\varphi_1 \perp_{\widetilde{s}} \varphi_2$. Assume that $\mathbf{h}$ is a hash function, $\mathbf{h}(M)$ is not derivable from $\nu\widetilde{s}.(\varphi_1 \mid \varphi_2)$, and $\mathbf{h}(M) \not\preceq \varphi_1(x_1)$, $\mathbf{h}(M) \not\preceq \varphi_2(x_2)$ for all $x_1 \in \mathrm{dom}(\varphi_1), x_2 \in \mathrm{dom}(\varphi_2)$. Then*

$$\varphi_1 \perp_{\widetilde{s}} \varphi_2 \mid \{^{\mathbf{h}(M)}/_{x}\} \qquad \textit{and} \qquad \varphi_1 \mid \{^{\mathbf{h}(M)}/_{x}\} \perp_{\widetilde{s}} \varphi_2$$

PROOF. Since $\mathbf{h}(M)$ is not derivable from $\nu\widetilde{s}.(\varphi_1 \mid \varphi_2)$ it is easy to see that it is not derivable from $\nu\widetilde{s}.\varphi_2$ either. By applying Theorem 5.3.3 once for $\nu\widetilde{s}.(\varphi_1 \mid \varphi_2)$ and once for $\nu\widetilde{s}.\varphi_2$ we get

$$\begin{aligned}
\nu\widetilde{s}.(\varphi_1 \mid \varphi_2 \mid \{^{\mathbf{h}(M)}/_{x}\}) &\approx_{\mathrm{s}} \nu\widetilde{s}.(\varphi_1 \mid \varphi_2) \mid \nu r.\{^{r}/_{x}\} && \text{Theorem 5.3.3}\\
&\approx_{\mathrm{s}} \nu\widetilde{s}.\varphi_1 \mid \nu\widetilde{s}.\varphi_2 \mid \nu r.\{^{r}/_{x}\} && \varphi_1 \perp_{\widetilde{s}} \varphi_2\\
&\approx_{\mathrm{s}} \nu\widetilde{s}.\varphi_1 \mid \nu\widetilde{s}.(\varphi_2 \mid \{^{\mathbf{h}(M)}/_{x}\}) && \text{Theorem 5.3.3}
\end{aligned}$$

which means that $\varphi_1 \perp_{\widetilde{s}} \varphi_2 \mid \{^{\mathbf{h}(M)}/_{x}\}$.
The fact that $\varphi_1 \mid \{^{\mathbf{h}(M)}/_{x}\} \perp_{\widetilde{s}} \varphi_2$ follows directly by symmetry of $\perp$. □

The next proposition states that a hashed term is derivable from a frame only when the term itself is.

PROPOSITION 5.3.5. *Let φ be a frame and $\mathbf{h}$ a hash function. Assume that $\mathbf{h}(M) \not\preceq \varphi(y)$ for all $y \in \mathrm{dom}(\varphi)$. Then $\mathbf{h}(M)$ is derivable from φ if and only if M is derivable from φ.*

PROOF. The if direction is trivial. For the only if direction, assume that $K\varphi =_{\mathrm{E}} \mathbf{h}(M)$ for some K such that $\mathbf{fn}(K) \cap \mathbf{bn}(\varphi) = \emptyset$. From Lemma 5.3.2, we get that there exists $\mathbf{h}(N) \preceq K\varphi$ such that $N =_{\mathrm{E}} M$. From the hypothesis we have that $\mathbf{h}(N) \not\preceq \varphi(y)$ for all $y \in \mathrm{dom}(\varphi)$, so the $\mathbf{h}$ of $\mathbf{h}(N)$ must come from K. That is, there must exist some $\mathbf{h}(K') \preceq K$ such that $K'\varphi =_{\mathrm{E}} N$. Since $K' \lhd K$ we have $\mathbf{fn}(K') \cap \mathbf{bn}(\varphi) = \emptyset$, and since $K'\varphi =_{\mathrm{E}} N =_{\mathrm{E}} M$ we conclude that M is derivable from φ. □

We conclude this section by introducing the next proposition. The intuition behind it is that the knowledge of a hash $\mathbf{h}(M)$ does not contribute to derive a term N, unless it contains the hash.

PROPOSITION 5.3.6. *Let $\varphi = \nu\widetilde{n}.(\varphi' \mid \{^{\mathbf{h}(M)}/_{x}\})$ be a frame, let $\mathbf{h}$ be a hash function and N a closed term. Assume that $\mathbf{h}(M) \not\preceq \varphi(y)$ for all $y \in \mathrm{dom}(\varphi)$, and $\mathbf{h}(M) \not\preceq N$. Then N is derivable from φ if and only if it is derivable from $\nu\widetilde{n}.\varphi'$.*

PROOF. The if direction is trivial. For the only if direction, assume that $K\varphi =_{\mathrm{E}} N$ for some K such that $\mathbf{fn}(K) \cap \mathbf{bn}(\varphi) = \emptyset$.

It is easy to see that if $\mathbf{h}(M)$ itself is derivable from $\nu\widetilde{n}.\varphi'$ then, intuitively, we can first derive $\mathbf{h}(M)$ from $\nu\widetilde{n}.\varphi'$ alone, and then use it to derive N. More concretely, assume that $L\varphi' =_{\mathrm{E}} \mathbf{h}(M)$ for some L such that $\mathbf{fn}(L) \cap \mathbf{bn}(\varphi') = \emptyset$. Then, by setting $K' = K[{}^{L}/_{x}]$, we have

$$K'\varphi' = K[{}^{L}/_{x}]\varphi' = K\varphi'[{}^{L\varphi'}/_{x}] =_{\mathrm{E}} K\varphi'[{}^{\mathbf{h}(M)}/_{x}] = K\varphi =_{\mathrm{E}} N$$

which means that N is derivable from $\nu\widetilde{n}.\varphi'$.

If $\mathbf{h}(M)$ is not derivable from $\nu\widetilde{n}.\varphi'$, then we can apply Theorem 5.3.3 for $\nu\widetilde{n}.\varphi'$ and $\mathbf{h}(M)$, from which we get that $\varphi \approx_{\mathrm{s}} \nu\widetilde{n}.\varphi' \mid \nu r.\{{}^{r}/_{x}\}$. Note that $N = N\varphi$ since N is closed. Picking an r not occurring in K, N, from $K\varphi =_{\mathrm{E}} N\varphi$ and the definition of $\approx_{\mathrm{s}}$ we get that $K[{}^{r}/_{x}]\varphi' =_{\mathrm{E}} N[{}^{r}/_{x}]\varphi' = N$, which means that N is derivable from $\nu\widetilde{n}.\varphi'$ (using $K' = K[{}^{r}/_{x}]$). □

We use these results in Section 5.3.3 to prove privacy properties for single-step protocols.

5.3.2 Conditions

Not all the single-step protocols satisfy unlinkability, forward and backward privacy. To understand which conditions need to be satisfied in order to guarantee these properties, we identify the possible causes of privacy violations in a single-step protocol. We start with the notion of *unlinkability*. This property is clearly violated when the attacker can distinguish between the i-th and j-th sessions of a tag, thus infer some information about a tag session number. Consider, for instance, the extreme case where the output of a tag i-th session is $O_i(U^{i-1}(S_0)) = i$. Suppose that S_0 is a number, $U(x) = x + 1, O(x) = x$. Hence, a tag always outputs the sum of its state S_0 and its session number. The attacker can simply run some sessions on c_1 and c_2 to violate our definition of unlinkability. If the interfaces correspond to the same tag, a session run c_1 affects the outputs on c_2 and vice versa. For instance, if the attacker observes the output sequence $(1, 2, 3)$, after querying c_1, c_2 and c_1, respectively, he can infer that the interfaces are linked, otherwise the output sequence would have been $(1, 2, 2)$.

To improve the readability we define $x : M \triangleq \{{}^{M}/_{x}\}$. Then, we define the property $\mathcal{P}_{SI}$ (session indistinguishability property) as

DEFINITION 5.3.4. *A single-step protocol satisfies $\mathcal{P}_{SI}$ if and only if*

$$\nu s, \widetilde{\rho}_n.x : O_n(U^{n-1}(S_0)) \approx_s \nu s, \widetilde{\rho}_m.x : O_m(U^{m-1}(S_0)) \quad \forall n, m \in \mathbb{N}^+.$$

$\mathcal{P}_{SI}$ prevents the simple attack discussed above by forcing outputs coming from different sessions to look identical. However, it does not imply unlinkability. For instance, consider another extreme case where the tag output contains only the tag secret, i.e. $O_i(U^{i-1}(S_0)) = s$, e.g. with $S_0(y) = y, O(x) = U(x) = x$. This protocol satisfies $\mathcal{P}_{SI}$ since the output does not depend on the session number i. However, unlinkability is clearly violated, because the tag secret is sent in cleartext. The attacker can easily violate our definition of unlinkability by querying the interfaces c_1 and c_2, obtaining two messages and testing them for equivalence. If the interfaces are linked, the messages are identical, since each tag secret is unique. Protecting the secret with a hash, i.e. $O_i(U^{i-1}(S_0)) = \mathbf{h}(s)$ does not help either. Running two sessions on c_1, c_2 will give two different hashes in the case of independent interfaces and the same hash twice if the interfaces are linked. Thus, the two cases are again distinguishable. Unlinkability is violated in these examples because the output of every session remains the same. However, imposing a variable output would not suffice to imply unlinkability. Consider a single-step protocol sending a message $O_i(U^{i-1}(S_0)) = \mathbf{h}^i(s)$. Running two sessions on c_1 and c_2 will give $(x_1 = \mathbf{h}(s_1), x_2 = \mathbf{h}(s_2))$ in the case of two independent interfaces and $(x_1 = \mathbf{h}(s), x_2 = \mathbf{h}^2(s))$ in the case of linked interfaces. Although in both cases the outputs are different, by checking whether $\mathbf{h}(x_1) = x_2$ holds, the attacker can once again distinguish them.

The common problem behind these attacks is that the output of two different sessions can be linked through the use of the common name s. The solution lies precisely in the notion of *frame independence*, which brings us to the definition of the property $\mathcal{P}_{OI}$ (output independence property).

DEFINITION 5.3.5. *A single-step protocol satisfies $\mathcal{P}_{OI}$ if and only if*

$$\prod_{i=1}^{n-1} x_i : O_i(U^{i-1}(S_0)) \perp_{s,\widetilde{\rho}_n} x_n : O_n(U^{n-1}(S_0)) \quad \forall n \in \mathbb{N}^+.$$

Intuitively, $\mathcal{P}_{OI}$ requires the tag output of the first $n-1$ sessions to be independent from the output of the n-th session, with respect to the tag secret s and the names $\widetilde{\rho}_n$. Hence, it requires n outputs of a tag to be indistinguishable from $n-1$ outputs of a tag and an output of a different tag.

Note that $\mathcal{P}_{SI}$ and $\mathcal{P}_{OI}$ are incomparable. In the first extreme case, $O_i(U^{i-1}(S_0)) = i$, the attacker exploits a dependency on i and the protocol satisfies $\mathcal{P}_{OI}$ but not $\mathcal{P}_{SI}$. Instead, in the second extreme case, $O_i(U^{i-1}(S_0)) = s$, the attacker exploits a dependency on s and the protocol satisfies $\mathcal{P}_{SI}$ but not $\mathcal{P}_{OI}$. To obtain unlinkability for single-step protocols we need both the properties $\mathcal{P}_{SI}$ and $\mathcal{P}_{OI}$ to hold.

THEOREM 5.3.7. *A single-step protocol satisfies unlinkability if and only if it satisfies both $\mathcal{P}_{SI}$ and $\mathcal{P}_{OI}$.*

The proof of Theorem 5.3.7 can be found in Appendix A.5.

For the stronger properties of forward and backward privacy we have to take into account a stronger attacker. In the definition of *forward privacy* we assume that the attacker can reveal the state of a tag. His goal is to link a tag state to its previous outputs. The properties $\mathcal{P}_{SI}$ and $\mathcal{P}_{OI}$ do not suffice to guarantee forward privacy. A new property $\mathcal{P}_{FI}$ (forward output independence property) is needed to ensure that a tag state (and not just its output) is independent from all its previous sessions.

DEFINITION 5.3.6. *A single-step protocol satisfies $\mathcal{P}_{FI}$ if and only if*

$$\prod_{i=1}^{n-1} x_i : O_i(U^{i-1}(S_0)) \perp_{s,\widetilde{\rho}_n} x_n : U^{n-1}(S_0) \quad \forall n \in \mathbb{N}^+.$$

The property $\mathcal{P}_{FI}$ is similar to $\mathcal{P}_{OI}$, but requires the first $n-1$ outputs to be independent from the n-th *tag state* rather than the n-th output. $\mathcal{P}_{FI}$ is strictly stronger than $\mathcal{P}_{OI}$.

PROPOSITION 5.3.8. *For all single-step protocols, property $\mathcal{P}_{FI}$ implies $\mathcal{P}_{OI}$.*

PROOF. $\mathcal{P}_{FI}$ requires the first $n-1$ outputs to be independent from the n-th state content, while $\mathcal{P}_{OI}$ requires their independence from the n-th output. Intuitively, when the attacker knows the n-th state of a tag, he can derive its next output by applying O_n to its state $U^{n-1}(S_0)$. Since the only occurrences of the secret s in the n-th output $O_n(U^{n-1}(S_0))$ are in the tag state $U^{n-1}(S_0)$, it cannot introduce a dependency on s, because the tag state is already independent from the first $n-1$ outputs by hypothesis.

Formally, when $\mathcal{P}_{FI}$ holds, we have that

$$\textstyle\prod_{i=1}^{n-1} x_i : O_i(U^{i-1}(S_0)) \perp_{s,\widetilde{\rho}_n} y : U^{n-1}(S_0) \quad \forall n \in \mathbb{N}^+.$$

We extend the right-hand side with $x_n : O_n(y)$ (note that $s \notin O_n(y)$) and we restrict y to obtain

$$\textstyle\prod_{i=1}^{n-1} x_i : O_i(U^{i-1}(S_0)) \perp_{s,\widetilde{\rho}_n} \nu y.\big(y : U^{n-1}(S_0) \mid x_n : O_n(y)\big) \quad \forall n \in \mathbb{N}^+$$

which due to structural equivalence and by Proposition 5.3.1 is equivalent to

$$\textstyle\prod_{i=1}^{n-1} x_i : O_i(U^{i-1}(S_0)) \perp_{s,\widetilde{\rho}_n} x_n : O_n(U^{n-1}(S_0)) \quad \forall n \in \mathbb{N}^+$$

which is $\mathcal{P}_{OI}$. □

$\mathcal{P}_{SI}$ and $\mathcal{P}_{FI}$ together are necessary and sufficient conditions for forward privacy, as stated by the theorem below.

THEOREM 5.3.9 (Forward privacy for single-step protocols). *A single-step protocol satisfies forward privacy (Definition 4.3.1) if and only if it satisfies $\mathcal{P}_{SI}$ and $\mathcal{P}_{FI}$.*

The proof of Theorem 5.3.9 can be found in Appendix A.6 and it is similar to the one for unlinkability. Note that Theorem 5.3.9 and Proposition 5.3.8 imply that forward privacy is stronger than unlinkability for single-step protocols, as already stated in Proposition 4.3.1.

Symmetrically to forward privacy, *backward privacy* is guaranteed by a protocol when an attacker cannot link the internal state of a tag to its future sessions. We need to assume that, after the attacker has tampered with a tag, the tag executes a legitimate run that the attacker cannot eavesdropped. If this run suffices to successfully update the internal state in a way that it cannot be inferred by the attacker, then the protocol guarantees backward privacy. Hence, we need to express the fact that the internal state and all the previous sessions are independent from the protocol sessions executed *after* the restore session. This leads us to define property $\mathcal{P}_{BI}$ (backward output independence property).

DEFINITION 5.3.7. *A single-step protocol satisfies $\mathcal{P}_{BI}$ if and only if*

$$\prod_{i=1}^{n-2} x_i : O_i(U^{i-1}(S_0)) \mid y : U^{d-1}(S_0) \perp_{s,\widetilde{\rho}_m} \prod_{i=n}^{m} x_i : O_i(U^{i-1}(S_0)) \qquad \forall d, n, m \in \mathbb{N}^+ : d+1 < n \leq m.$$

$\mathcal{P}_{BI}$ requires the internal state of a tag (i.e. the state at the d-th run, contained in variable y) and all its output before the restore session (the $n-1$-st session of the tag) to be independent from its outputs after the restore session. It implies $\mathcal{P}_{OI}$, confirming that backward privacy is stronger than unlinkability, as expected by Proposition 4.4.1.

PROPOSITION 5.3.10. *For all single-step protocols, property $\mathcal{P}_{BI}$ implies $\mathcal{P}_{OI}$.*

PROOF. $\mathcal{P}_{BI}$ requires the outputs of a tag after $n-1$ runs to be independent from its previous outputs and state content. Instead, $\mathcal{P}_{OI}$ requires the independence of the first $n-1$-st outputs from the n-th output. Intuitively,

the left-hand side of the independence formula of $\mathcal{P}_{BI}$ differs from the one of $\mathcal{P}_{OI}$ for the variable y, which is bound to a tag state in $\mathcal{P}_{BI}$ and does not appear in $\mathcal{P}_{OI}$, where instead there is the $n-1$-st output of such tag. Also, the right-hand side of $\mathcal{P}_{OI}$ contains one output only (the n-th), while in $\mathcal{P}_{BI}$ we can have an arbitrary number of tag outputs (from the n-th to the m-th).

Formally, assuming $\mathcal{P}_{BI}$ we have

$$\prod_{i=1}^{n-2} x_i{:}O_i(U^{i-1}(S_0)) \mid y{:}U^{d-1}(S_0) \perp_{s,\widetilde{\rho}_m} \prod_{i=n}^{m} x_i{:}O_i(U^{i-1}(S_0)) \qquad \forall d, n, m \in \mathbb{N}^+ : d+1 < n \leq m.$$

We extend the left hand side with the output $x_{n-1} : O_{n-1}(U^{n-d-1}(y))$ and restrict the variable y. We also restrict the variables $x_{n+1}, \ldots, x_m$ on the right-hand side. By Proposition 5.3.1 and structural equivalence we obtain that

$$\prod_{i=1}^{n-2} x_i{:}O_i(U^{i-1}(S_0)) \mid \nu y.\big(y{:}U^{d-1}(S_0) \mid x_{n-1}{:}O_{n-1}(U^{n-d-1}(y))\big) \perp_{s,\widetilde{\rho}_m} \nu x_{n+1}, \ldots, x_m.\prod_{i=n}^{m} x_i{:}O_i(U^{i-1}(S_0)) \quad \forall d, n, m \in \mathbb{N}^+ : d+1 < n \leq m$$

that corresponds to

$$\prod_{i=1}^{n-1} x_i{:}O_i(U^{i-1}(S_0)) \perp_{s,\widetilde{\rho}_n} x_n{:}O_n(U^{n-1}(S_0)) \quad \forall n \in \mathbb{N}^+$$

which is $\mathcal{P}_{OI}$. □

The properties $\mathcal{P}_{SI}$ and $\mathcal{P}_{BI}$ are necessary and sufficient conditions for backward privacy, as stated by the theorem below.

THEOREM 5.3.11. *A single-step protocol satisfies backward privacy (Definition 4.4.1) if and only if it satisfies $\mathcal{P}_{SI}$ and $\mathcal{P}_{BI}$.*

The proof of Theorem 5.3.11 can be found in Appendix A.7.

The advantage of having simple conditions to prove properties with respect to the general definitions of Section 4.1 is that they are easier to verify. These conditions involve only static equivalences between messages and not the full dynamics of the protocol. In the next section we use these conditions to verify some protocols from the literature, manually in the general case and automatically for a bounded number of sessions. This task is much more challenging using the general definitions of Section 4.1.

5.3.3 Examples

In this section we apply the results of Section 5.3.2 to protocols from the literature. We study the OSK protocol [33] in more detail. We also discuss some variations of the protocol, where we modify some of its aspects to examine how privacy is affected. Then we analyse a basic hash protocol of [41], which falls in the same class even though it is quite different in nature from the OSK protocol. Finally, we study a simplified version of a protocol proposed in [29].

The OSK protocol

Tags running the OSK protocol [33] (see Figure 4.1 in Section 4.1) can compute two distinct hash functions $\mathbf{g}$ and $\mathbf{h}$. The state of each tag is initialised with a randomly generated secret, which is also known to the backend. During each run, the tag computes the hash $\mathbf{g}$ of its current state and sends it to the reader. Then it computes the hash $\mathbf{h}$ of its current state and updates the state with the result. The output of the i-th run of a tag is $\mathbf{g}(\mathbf{h}^{i-1}(s))$, where s is the initial secret. The backend knows the secret of all the tags. It can thus compute $\mathbf{g}(\mathbf{h}^{i-1}(s))$ for all the secrets to identify the tag. For efficiency, the backend can precompute the expected output for the next run of all the tags and perform a fast search. Once the tag is identified, its expected output can be updated.

The OSK protocol can be modelled as a single-step protocol (Definition 5.1.1) with

$$\begin{aligned} S_0(y) &= y & O(x) &= \mathbf{g}(x) \\ \widetilde{\rho} &= \varepsilon & U(x) &= \mathbf{h}(x) \end{aligned}$$

This protocol guarantees forward privacy [33]. To prove this, we only need to show that the OSK protocol satisfies the properties $\mathcal{P}_{SI}$ and $\mathcal{P}_{FI}$.

PROPOSITION 5.3.12. *The OSK protocol satisfies properties $\mathcal{P}_{SI}, \mathcal{P}_{FI}$, namely*

$$\begin{aligned} &\mathcal{P}_{SI} && \nu s.x : \mathbf{g}(\mathbf{h}^{n-1}(s)) \approx_s \nu s.x : \mathbf{g}(\mathbf{h}^{m-1}(s)) \quad \forall n, m \in \mathbb{N}^+ \\ &\mathcal{P}_{FI} && \textstyle\prod_{i=1}^{n-1} x_i : \mathbf{g}(\mathbf{h}^{i-1}(s)) \perp_s x_n : \mathbf{h}^{n-1}(s) \quad \forall n \in \mathbb{N}^+ \end{aligned}$$

PROOF. The frames in $\mathcal{P}_{SI}$ contain hashes of the secret. From Theorem 5.3.3 we know that both these frames corresponds exactly to $\nu r.x : r$. Therefore, they are statically equivalent, i.e. an observer who does not know the secret s sees two values that look random.

The challenging part of the proof is showing that $\mathcal{P}_{FI}$ holds. We have to show the independence of the current state of a tag from the previous outputs. We use Proposition 5.3.4 to prove this condition. We choose $\varphi_1 = \prod_{i=1}^{n-1} x_i : \mathbf{g}(\mathbf{h}^{i-1}(s))$ and $\varphi_2 = \emptyset$, which are trivially independent with respect to the secret s. We want to extend the frame φ_2 with the hash $\mathbf{h}^{n-1}(s)$ while preserving its independence from φ_1. The functions $\mathbf{h}$ and $\mathbf{g}$ are assumed to be one-way hash functions. The hash $\mathbf{h}^{n-1}(s)$ is not derivable from $\nu s.(\varphi_1 \mid \varphi_2)$ because the hash functions are not reversible. Also, it is easy to see that for $i < n-1$, $\mathbf{h}^{n-1}(s)$ is not a subterm of $\varphi_2(x)$ (its domain is empty) or of $\varphi_1(x) = \mathbf{g}(\mathbf{h}^{i-1}(s))$, which follows from the fact that a subterm of $\mathbf{g}(\mathbf{h}^{i-1}(s))$ contains at most $n-2$ occurrences of $\mathbf{h}$. Hence, we can directly apply Proposition 5.3.4 obtaining $\varphi_1 \perp_s \varphi_2 \mid \mathbf{h}^{n-1}(s)$, which is $\mathcal{P}_{FI}$. □

By Theorem 5.3.9 we can conclude that the OSK protocol satisfies forward privacy (and, as a consequence, also unlinkability).

Proof mechanisation. Static equivalence between applied pi calculus processes can be proven automatically in many cases using the tool ProVerif [8]. Proving $\mathcal{P}_{SI}$ and $\mathcal{P}_{FI}$ involves proving an infinite number of static equivalences. For $\mathcal{P}_{SI}$ we can overcome this problem by constructing a pair of dynamic processes that can produce all the pairs of messages, and then use ProVerif to prove all the static equivalences at once. The ProVerif code can be found in Appendix B.1. This is not possible for $\mathcal{P}_{OI}$, $\mathcal{P}_{FI}$ and $\mathcal{P}_{BI}$. Still, each one of their static equivalences can be proven automatically by ProVerif (see Appendix B.2). Proving these equivalences up to a fixed n corresponds to proving the privacy properties up to a fixed number of tag sessions. We used ProVerif successfully to prove these equivalences for up to 1000 sessions, which only took a few minutes of computation.

ProVerif is capable of automatically proving *observational* equivalence [9], but this capability is limited to very simple cases where the processes to test are almost identical. Equivalences like the ones of our general definitions of unlinkability, forward and backward privacy (Definitions 4.2.1, 4.3.1 and 4.4.1) are currently beyond the capabilities of the tool. Also our conditions for unlinkability, forward and backward privacy do not result in a complete mechanisation for the single-step protocols. Still, it can be argued that extending ProVerif (or building new techniques) to deal with conditions $\mathcal{P}_{OI}$, $\mathcal{P}_{FI}$ and $\mathcal{P}_{BI}$ should be easier than to deal with Definitions 4.2.1, 4.3.1 and 4.4.1. Moreover, ProVerif translates applied pi calculus processes into horn clauses which effectively causes them to be replicated. Thus, there is no direct

way of using ProVerif to prove our general definitions of unlinkability even for a bounded number of sessions (which we achieved for the single-step protocols). Even though no complete mechanisation has been achieved, the use of the conditions rather than the general definitions allowed us to manually prove forward privacy for OSK for an *unbounded number* of sessions and the proof was greatly simplified.

Weak OSK protocol

Here we examine the effects of relaxing the conditions on the hash functions $\mathbf{h}$ and $\mathbf{g}$ of the OSK protocol. First, we consider the case where $\mathbf{h}$ is not one-way, that is there exists a function $\mathbf{invh}$ and an equation $\mathbf{invh}(\mathbf{h}(x)) =_{\mathrm{E}} x$. This violates forward privacy since the attacker can compute s from $\mathbf{h}^n(s)$ (the state of the tag that the attacker tampered with), which can be then used to link past sessions to the tag. Indeed, Proposition 5.3.4 cannot be applied to $\mathbf{h}^n(s)$ since $\mathbf{h}$ is not a hash function as defined by Definition 5.3.2. Property $\mathcal{P}_{FI}$ is violated. On the other hand, even if $\mathbf{h}(x)$ is an invertible but non-repeating function ($\mathbf{h}^i(x) \neq \mathbf{h}^j(x)$ for $i \neq j$), e.g. $\mathbf{h}(x) = x + 1$, then property $\mathcal{P}_{OI}$ is still satisfied as

$$\textstyle\prod_{i=1}^{n-1} x_i : \mathbf{g}(\mathbf{h}^{i-1}(s)) \perp_s x_n : \mathbf{g}(\mathbf{h}^{n-1}(s)).$$

This follows again from Proposition 5.3.4. By Theorem 5.3.7 we can conclude that the protocol satisfies unlinkability.

On the other hand, if the inverse of $\mathbf{g}$ exists, then both unlinkability and forward privacy are violated. In this case, given two outputs $\mathbf{g}(\mathbf{h}^i(s))$ and $\mathbf{g}(\mathbf{h}^j(s))$ with $i < j$, the attacker can easily infer whether they belong to the same tag. He can extract $\mathbf{h}^i(s)$ and $\mathbf{h}^j(s)$ from the outputs. Then, since $\mathbf{h}$ is a public hash function, he can apply it $(j - i)$ times to the first value. If the resulting value coincides with the output $\mathbf{g}(\mathbf{h}^j(s))$, the attacker can conclude that the outputs belong to the same tag. Hence, both properties $\mathcal{P}_{OI}$ and $\mathcal{P}_{FI}$ are violated (even though $\mathcal{P}_{SI}$ is still satisfied).

BP protocol: a variation of the protocol in [29]

The OSK protocol does not guarantee backward privacy. Once the attacker obtains the secret of a tag he can calculate all the next outputs because they only depend on the secret. To achieve backward privacy this protocol should be strengthened so that a tag state is updated in a way that looks random to the attacker when he misses one session. Hence, we propose here the BP protocol

(Backward Privacy protocol) inspired by the protocol proposed in [29], with

$$S_0(y) = y \qquad O(x) = (\mathbf{g}(x), r)$$
$$\widetilde{\rho} = r \qquad U(x) = \mathbf{h}(x, r)$$

In the BP protocol the tag generates a random number r and sends it to the reader together with the hash ($\mathbf{g}$) of the current state. Both the tag and the reader update the secret with the hash ($\mathbf{h}$) of the secret and the random number. Intuitively, if the attacker misses one session, he also misses the random number sent during that session that is needed for calculating the next secret. He therefore cannot link the future sessions of a tag to its state. Proving backward privacy for the BP protocol is equivalent to proving the properties $\mathcal{P}_{SI}$ and $\mathcal{P}_{BI}$. We use r_i for the random number sent in the i-th session and we define $\mathbf{h}_r^{n+1}(s)$ as $\mathbf{h}(s, r_1)$ for $n = 0$ and $h(h_r^n(s), r_{n+1})$ for $n > 0$.

PROPOSITION 5.3.13. *The BP protocol satisfies properties $\mathcal{P}_{SI}, \mathcal{P}_{BI}$, namely*

$$\mathcal{P}_{SI} \quad \nu s, \tilde{r}.x : (\mathbf{g}(\mathbf{h}_r^{n-1}(s), r_{n-1})) \approx_s \nu s, \tilde{r}.x : (\mathbf{g}(\mathbf{h}_r^{m-1}(s), r_{m-1})) \qquad \forall n, m \in \mathbb{N}^+$$

$$\mathcal{P}_{BI} \quad \prod_{i=1}^{n-2} x_i : (\mathbf{g}(\mathbf{h}_r^{i-1}(s)), r_i) \mid y : \mathbf{h}_r^{d-1}(s) \perp_{s, r_1, \ldots, r_m} \prod_{i=n}^{m} x_i : (\mathbf{g}(\mathbf{h}_r^{i-1}(s)), r_i) \quad \forall d, n, m \in \mathbb{N}^+ : d + 1 < n \leq m$$

PROOF. By Theorem 5.3.3 we have that $\nu s, \tilde{r}.x : (\mathbf{g}(\mathbf{h}_r^{n-1}(s), r_{n-1}))$ and $\nu s, \tilde{r}.x : (\mathbf{g}(\mathbf{h}_r^{m-1}(s), r_{m-1}))$ are both equivalent to $\nu s, r^*.x : r^*$, therefore they are equivalent to each other by transitivity ($\mathcal{P}_{SI}$). For $\mathcal{P}_{BI}$ we repeatedly use Proposition 5.3.4 first with $\varphi_1 = \prod_{i=1}^{n-2} x_i : (\mathbf{g}(\mathbf{h}_r^{i-1}(s)), r_i) \mid y : \mathbf{h}_r^{d-1}(s)$ and $\varphi_2 = \prod_{i=n}^{m} y_i : r_i$. These frames are trivially independent with respect to the secret s and the random values r_i because φ_1 only contains the random values $r_1, \ldots, r_{n-2}$ and φ_2 only contains $r_n, \ldots, r_m$ and does not contain s. We take $\mathbf{g}$ as the hash function and $\mathbf{h}_r^n(s)$ as the term M. The term $\mathbf{g}(\mathbf{h}_r^{n-1}(s)) = \mathbf{g}(\mathbf{h}_h^{n-2}(s), r_{n-1})$ is not derivable from $\nu s, r_1, \ldots, r_m.(\varphi_1 \mid \varphi_2)$ because the latter does not contain r_{n-1}. Also, this term ($\mathbf{g}(\mathbf{h}_r^{n-1}(s))$) is not equivalent to a subterm of terms in φ_1 or φ_2. Therefore we can apply proposition 5.3.4 obtaining

$$\prod_{i=1}^{n-2} x_i : (\mathbf{g}(\mathbf{h}_r^{i-1}(s)), r_i) \mid y : \mathbf{h}_r^{d-1}(s) \perp_{s, r_1, \ldots, r_m} z_n : (\mathbf{g}(\mathbf{h}_r^n(s))) \mid \prod_{i=n}^{m} y_i : r_i \quad \forall d, n, m \in \mathbb{N}^+ : d + 1 < n \leq m.$$

Note that (as $\mathbf{g}$ is a hash function) having $\mathbf{g}(\mathbf{h}_r^i(s))$ does not help in deriving $\mathbf{g}(\mathbf{h}_r^j(s))$ ($j > i$). In fact, by Theorem 5.3.3 we know that $x : \mathbf{g}(\mathbf{h}_r^i(s))$ equates

to $x:\nu r.\{^r/_x\}$ (a nonce) which clearly gives no information to build $\mathbf{g}(\mathbf{h}_r^j(s))$. Therefore we can use the same reasoning to add substitutions to the right-hand side of the independence and obtain

$$\prod_{i=1}^{n-2} x_i:(\mathbf{g}(\mathbf{h}_r^{i-1}(s)), r_i) \mid y:\mathbf{h}_r^{d-1}(s) \perp_{s,r_1,\ldots,r_m} \prod_{i=n}^{m} z_i:(\mathbf{g}(\mathbf{h}_r^{i-1}(s))) \mid \prod_{i=n}^{m} y_i:r_i \quad \forall d,n,m \in \mathbb{N}^+ : d+1 < n \leq m$$

We now have $\mathbf{g}(\mathbf{h}_r^{i-1}(s))$ and r_i on the right-hand side, but we need $\mathbf{g}(\mathbf{h}_r^{i-1}(s), r_i)$. We extend the right-hand side with $\prod_{i=n+1}^{m} x_i:(z_i, y_i)$ as the conditions for Proposition 5.3.4 are trivially satisfied. Then, we can restrict all the y_i and z_i using structural equivalence and we get

$$\prod_{i=1}^{n-2} x_i:(\mathbf{g}(\mathbf{h}_r^{i-1}(s)), r_i) \mid y:\mathbf{h}_r^{d-1}(s) \perp_{s,r_1,\ldots,r_m} \prod_{i=n}^{m} x_i:(\mathbf{g}(\mathbf{h}_r^{i-1}(s)), r_i) \quad \forall d,n,m \in \mathbb{N}^+ : d+1 < n \leq m$$

which is $\mathcal{P}_{BI}$. □

By Theorem 5.3.11 we can conclude that the BP protocol satisfies backward privacy (and, as a consequence, also unlinkability by Proposition 5.3.10).

Basic hash protocol of [41]

The basic hash protocol of [41] uses a random number generator and a hash function $\mathbf{h}$. The state of each tag is initialised with a randomly generated secret, known to the backend, which is never updated. Instead, a tag generates a fresh nonce r and computes the hash $\mathbf{h}(s,r)$ of its secret together with r. Finally it outputs the pair $(r, \mathbf{h}(s,r))$. The backend computes $\mathbf{h}(s,r)$ for all the known tags and compares it with the given value to identify the tag.

This basic hash protocol differs from the other protocols described in this section in that it never updates the tag state. We can model it as a single-step protocol (Definition 5.1.1) with

$$S_0 = s \qquad O(x) = (r, \mathbf{h}(x,r))$$
$$\widetilde{\rho} = r \qquad U(x) = x$$

We use r_i for the random number sent during the i-th session. Therefore, we can prove unlinkability by proving the properties $\mathcal{P}_{SI}$ and $\mathcal{P}_{OI}$.

PROPOSITION 5.3.14. *The basic hash protocol satisfies properties $\mathcal{P}_{SI}, \mathcal{P}_{OI}$, namely*

$$\mathcal{P}_{SI} \qquad \nu s.\nu r.x:(r, \mathbf{h}(s,r)) \approx_s \nu s.\nu r.x:(r, \mathbf{h}(s,r))$$
$$\mathcal{P}_{OI} \qquad \prod_{i=1}^{n-1} x_i:(r_i, \mathbf{h}(s,r_i)) \perp_{s,r_1,\ldots,r_n} x_n:(r_i, \mathbf{h}(s,r_i)) \forall n \in \mathbb{N}^+$$

PROOF. $\mathcal{P}_{SI}$ follows trivially from reflexivity of $\approx_s$.

For $\mathcal{P}_{OI}$ we want to prove the independence of a tag output from its previous ones. Let $r_1, \ldots, r_n$ be distinct names, $\varphi_1 = \prod_{i=1}^{n-1} x_i : (r_i, \mathbf{h}(s, r_i))$ and $\varphi_2 = y_n : r_n$. Since $\mathbf{h}(s, r_n)$ cannot be derived from $\nu s, r_1, \ldots, r_n.(\varphi_1 \mid \varphi_2)$ (s is secret) and is not a subterm of terms contained in φ_1 or φ_2, we can directly apply Proposition 5.3.4 to obtain $\varphi_1 \perp_{s,r_1,\ldots,r_n} \varphi_2 \mid \mathbf{h}(s, r_n)$ that is

$$\prod_{i=1}^{n-1} x_i : (r_i, \mathbf{h}(s, r_i)) \perp_{s,r_1,\ldots,r_n} y_n : r_n \mid z_n : \mathbf{h}(s, r_n).$$

We now have $\mathbf{h}(s, r_n)$ and r_n in the right-hand side of the independence, but we need $(r_n, \mathbf{h}(s, r_n))$. Using Proposition 5.3.1 (2.) and (3.) we extend the right-hand side with $x_n : (y_n, z_n)$ and we restrict y_n and z_n. Thus we get

$$\prod_{i=1}^{n-1} x_i : (r_i, \mathbf{h}(s, r_i)) \perp_{s,r_1,\ldots,r_n} x_n : (r_n, \mathbf{h}(s, r_n))$$

which is $\mathcal{P}_{OI}$.

Thus, by Theorem 5.3.7, we conclude that the protocol satisfies unlinkability. □

Similarly to the OSK protocol, each one of the infinite equivalences of $\mathcal{P}_{OI}$ can be proven by ProVerif (see Appendix B.3). We can thus have an automated proof of unlinkability for any bounded number of sessions.

Forward and backward privacy are violated. If the attacker obtains s by tampering with a tag, he can link it to any previous and future session by hashing the secret with the random numbers sent by the tags. Indeed, $\mathcal{P}_{FI}$ and $\mathcal{P}_{BI}$ are not satisfied.

5.4 Conclusions

In this chapter we study the class of the single-step protocols. First, we show how our abstract model in the epistemic logic can be instantiated to a concrete model, in particular to the one in the applied pi calculus of Arapinis et al. [3]. Then, we demonstrate that all the privacy properties defined in Chapter 3 coincide for the class of single-step protocols. Also, we identify the conditions that must hold for a single-step protocol in order to guarantee unlinkability, forward or backward privacy in the RFID model that we presented in Chapter 4. Finally, we study some protocols from the literature and prove when they satisfy the privacy properties described in Chapter 4 and prove it by showing that they satisfy the conditions presented in this chapter.

6

CONCLUSIONS

In this thesis we study RFID systems, a technology that raised a number of concerns and introduced new challenges in the last few years. The mobile nature of RFID systems and the limited capabilities of tags are the main factors that make it difficult to protect user privacy. In fact, the wireless communication simplifies eavesdropping protocol sessions, while the impossibility to use full-fledged cryptographic algorithms makes it unfeasible to use most of the existing protocols. These features cause RFID protocols to be subject to a range of attacks that differ from privacy threats in other systems. These attacks do not aim at disclosing private information stored in the user devices, as tags do not usually store valuable data, but rather at linking sessions run by the same user in order to trace the movements of a specific tag and gain sensitive information about the person or the object carrying the tag.

The main goal of this thesis is to propose a formal framework in which we can define privacy properties, describe identification protocols and verify whether they guarantee these properties. Our main challenge is to formally demonstrate whether an RFID protocol guarantees privacy properties such as unlinkability, forward and backward privacy. Accordingly, the research questions of this thesis (see Section 1.3) are:

- *How can we define a unifying framework to model and compare different privacy definitions?*
- *How can we implement a model in which we can verify whether a protocol guarantees unlinkability and its related notions?*

A framework that responds to the first research question is required to be general and abstract, so that it can capture properties defined in other models, and also allow their formal comparison. To satisfy this requirement, we present a

model in the epistemic logic in Chapter 3. Our model is kept general by not instantiating the elements it is composed of. By abstracting away from the technical details of concrete models, we can capture the intuition behind the corresponding definitions and compare them in a unifying framework. We demonstrate that these definitions provide different privacy guarantees and study these differences. Hence, we study the conditions under which the definitions coincide. Such conditions, in our opinion, are satisfied by a large number of identification systems. By also capturing different notions of forward and backward privacy we provide a complete study of the most relevant privacy definitions in the context of RFID systems. Moreover, in Chapter 5 we show, as an example, how to instantiate our abstract model into a concrete one, in particular the one of Arapinis et al. [3]. To instantiate our model, we only need to select a concrete model and use its definitions of the attacker's abilities and the equivalence relation between traces as instances of our abstract definitions. By using the concrete model presented in [3], we can prove that all the notions of unlinkability discussed in Chapter 3 coincide for the class of single-step protocols in this specific setting.

To answer the second research question, we take the following actions:

(1) we fully clarify which verifiable conditions must hold for a protocol in order to satisfy the notions of unlinkability, forward and backward privacy;

(2) we define an intuitive model, tailored to RFID systems, that allows to:

   (a) specify protocol descriptions;

   (b) specify privacy properties;

   (c) formally (and automatically) verify whether a protocol guarantees the properties.

The answer to the first research question provides us with an overview of the existing definitions and corresponding privacy guarantees that they imply, satisfying (1). We express the strongest privacy properties of our abstract model in the syntax of our concrete RFID model. This model, described in Chapter 4, is based on the applied pi calculus, that provides mechanisms to describe protocols and specify cryptographic primitives through an equational theory. However, since one of our goals is to specifically model RFID systems, the description of a protocol must follow some rules. We modelled the system as an infinite amount of tags running in parallel with readers and a backend

system. Each tag process is composed of an initialisation phase and an unbounded number of protocol sessions (2a). The initialisation phase is needed to capture the fact that tags have a state, therefore we use it to store data in the tag memory including its initial unique secret, which we use in our privacy definitions. In fact, our RFID model defines unlinkability, forward and backward privacy (2b) by requiring that sessions executed by one tag (using the same initial secret) are indistinguishable from sessions executed by different tags (having therefore different initial tag secrets). These privacy definitions use the concept of observational equivalence, that cannot be verified by automatic tools because they are not satisfactorily able to handle it yet. Hence, in Chapter 5, we show that our definitions can be reduced to checking simpler conditions for a class of protocols. Each of these conditions can be verified by the tool ProVerif for a *bounded* number of sessions. We also formally prove by hand that such conditions are satisfied by some protocols for an *unbounded* number of sessions, as required by (2c).

Several works in the literature aim at comparing existing privacy definitions, thus sharing our goal of analysing definitions in an abstract framework as we do in Chapter 3. For example, [37, 26, 25, 17, 21] follow a logic approach to study and compare privacy properties. However, none of these works investigates unlinkability. Instead, they focus on other privacy properties, e.g. anonymity. Although the concept of unlinkability has been widely studied, its several definitions in the literature have never been formally compared. As far as we know, only [35] attempted to clarify the meaning of unlinkability (and other privacy notions), but only at an informal level. Hence, with respect to existing works, the benefits of our first contribution, namely our abstract model, are two-fold. First, by formalising existing definitions of unlinkability in a unifying framework, we clarify exactly which privacy guarantees are offered by such definitions. Second, we extend the model to study other privacy notions such as inseparability, forward and backward privacy.

The second contribution of this thesis is an RFID model where it is possible to describe protocols and to define and verify privacy properties. The literature on this topic includes several works, both in a computational setting [18, 27, 33, 4, 34, 40] and in a symbolic setting [38, 39, 2, 3]. On the one hand, the definitions given in a computational setting correspond to games played between an attacker and a challenger. In these games, unlinkability holds if the attacker cannot win, e.g. by telling whether two sessions are run by the same tag or different ones, with a probability higher than guessing. On the other hand, all the models in a symbolic setting give definitions based on the comparison of traces. We do not use a game-based approach because we

want our RFID model to allow automatic verification of protocols. Therefore we choose a model in a symbolic setting using the applied pi calculus. The approach we use to define our framework is based on observational equivalence and provides a stronger notion of unlinkability than [38]. Moreover, it also captures forward and backward privacy, and it formally proves that these properties are stronger than unlinkability. As far as we know, ours is the first attempt to define forward and backward privacy in a symbolic setting, since the existing definitions are usually given in a computational setting [33, 22, 24]. Hence, the main benefit of this second contribution is that the privacy definitions expressed in our model offer the precision deriving by the use of a symbolic setting and the strongest privacy guarantees.

Finally, the definition of our formal models led us to the study of a class of protocols. In Chapter 5 we study their properties in both our abstract and concrete models. Since our models are given in a symbolic setting, we formally demonstrate that all the privacy properties that we studied in Chapter 3 coincide for single-step protocols. Also, we prove that the privacy properties defined in Chapter 4 correspond to simpler conditions in this case. These conditions consist of equivalences that can be tested in ProVerif, therefore we could automatically verify some RFID protocols.

6.1 Limitations of the models

Our models achieve their main objectives, but have some limitations.

Our abstract model captures and compares definitions of unlinkability expressed in the epistemic logic. Our ddefinitions are inspired by definitions from the literature given in both a symbolic and a computational setting. On the one hand, this framework helps to understand the differences between the intuition behind each definition, but, on the other hand, translating definitions in our setting is a difficult task that may introduce imprecisions. These imprecisions may negatively affect the possibility of comparing definitions given in different models.

In Chapter 4 we present an RFID model based on the applied pi calculus. By using a formal language, we can provide precise definitions and formally prove our results. Moreover, we can model protocols from the literature and demonstrate (manually in the general case and automatically for a bounded number of sessions) whether they guarantee the privacy definitions given in our model. The drawback of using our symbolic analysis is that it does not allow to capture all the concrete properties that cryptographic primitives may

have. This drawback is common to any symbolic approach, but it is worth mentioning. The applied pi calculus allows to describe such primitives by means of an equational theory. However, such a theory cannot always fully express their functioning. For example, when we model the functions for the encryption and decryption of messages using a symmetric key, as shown in Section 2.2.1, we are implicitly assuming that there is no way to reverse the encryption function without knowing the symmetric key. However, there may be some weakness in the function implementation that leak information about the encrypted message. Such weaknesses are not always expressible in the applied pi calculus, therefore the protocol analysis might miss attacks that aim at using them to disclose secret information. As mentioned in the introduction of this thesis, RFID protocols often adopt ad-hoc cryptographic solutions, which have sometimes been proven insecure. If the cryptographic primitives in use leak private information and this is not modelled in the equational theory, a protocol may not guarantee unlinkability, forward or backward privacy, while satisfying the conditions of our definitions. Therefore, when a protocol satisfies our privacy definitions, it is important to ensure that the assumptions in the equational theory actually match those on the security level offered by the cryptographic primitives in use.

6.2 Directions for future work

In this section we discuss possible extensions to the work presented in this thesis.

The set of properties that we analyse in Chapter 3 includes several definitions that are translated into our *possibilistic* epistemic model. However, there exist also some *probabilistic* approaches to define privacy notions that our model does not capture. In order to extend it to include probabilistic definitions, it is possible to use the technique that [25] introduced for the definition of probabilistic variants of anonymity. In the case of anonymity, [25] argues that a possibilistic definition may seem to have strong requirements while giving no concrete guarantee. This may happen also for unlinkability and its related notions of inseparability, forward and backward privacy. Consider for example a protocol satisfying our epistemic definition of strong unlinkability, meaning that an attacker cannot infer the existence of a link in any given trace. The fact that it is *possible* that each transaction of a trace is performed by a different agent (as required by our definition) does not imply that this is *probable*. It may actually happen that a protocol guarantees the

epistemic definition of strong unlinkability, but, when the attacker applies his strategy and observes some particular outcome, he knows that there is a high chance that this outcome was produced by a specific sequence of agents. In this example, although strong unlinkability holds, it is easy for the attacker to guess links. Hence, a possible direction for future work is to extend both our abstract and concrete RFID models and investigate probabilistic descriptions of unlinkability and its related notions.

One of the objectives of this thesis was to develop solutions to automatically verify privacy protocols. We achieved this goal for the class of single-step protocols by providing privacy definitions that can be verified using ProVerif [8] for a bounded number of sessions. A second direction for future work is to develop a framework able to verify the privacy properties described in this thesis for a wider class of protocols. One way to do so is to extend the RFID model by modifying the conditions of Chapter 5 to include protocols that consist of multiple messages, allowing an agent to receive inputs (in single-step protocols the agents are the tags of an RFID system and they can only output messages). This extension should consider that a protocol session may generate more than one output at a time and that the number of messages may even change from a session to another if the protocol has more functionalities (e.g. a protocol may have a mechanism to change a tag secret when needed). Hence, the analysis has to deal with much more information than in single-step protocols. Also, the attacker may overlap different sessions by sending, for example, the first tag output of a session as an input to another tag running a different session. Our analysis of single-step protocols does not capture these attacks, since these protocols allows no input in a tag process. Another way to generalise the analysis is to start from our abstract model described in Chapter 3 and follow the approach of [21]. [21] develops a more concrete model that bridges the gap between operational semantics and epistemic logic. It does so by offering a combined framework where it is possible to easily model a protocol in a process language with an operational semantics and to reason about properties expressed in a rich epistemic temporal logic.

A
PROOFS

A.1 Theorem 3.2.2 (Game-based unlinkability)

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.

PROOF. In Chapter 3 we introduced the definitions of two-agents game unlinkability (corresponding to equation 3.3) and three-agents game unlinkability (corresponding to equation 3.4). Here, we demonstrate that these two notions coincide to a third simpler definition of unlinkability, which we called game-based unlinkability (Definition 3.2.3).

First, we prove that both two-agents game unlinkability and three-agents game unlinkability imply game-based unlinkability. To show this, we have to prove that, when two or three-agents unlinkability holds, all the traces with the same length are equivalent. In particular, we show their equivalence to the trace executed entirely by a single agent for any given length n.

- *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 two-agents game unlinkability, 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 this procedure on τ' until the number of agents of the resulting trace is 1, we have that τ is equivalent

to the trace executed by a single agent, in particular its first agent, by transitivity. Using this method, we can prove that all traces are equivalent to a trace run by their first agent. To prove that it is also equivalent to any other trace, we need to show that the traces run by a single agent are equivalent to each other. By choosing π_x such that it is a sequence of x only, we obtain that all the traces executed by a single agent are equivalent. Therefore we can conclude that all the traces in the system are equivalent 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 $i = 2$. We replace all the occurrences of a_2 by x, obtaining $\pi_x = (a_1, x, a_3, x)$. Using equation (3.3), 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 using equation (3.3) 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 $\pi_\tau = (a_1, a_2, a_3, a_2) \sim (a_1, a_1, a_1, a_1) = \pi'_{a_1}$. Using two-agents game unlinkability, we choose $\pi_x = (x, x, x, x)$ and obtain that π'_{a_1} is equivalent to any trace run by a single agent.

- *Three-agents game unlinkability*. Three-agents game unlinkability is exactly two-agents game unlinkability when partial mappings do not contain y. Since two-agents game unlinkability implies game-based unlinkability, we can conclude that three-agents game unlinkability also implies game-based unlinkability.

We can conclude that when two-agents game unlinkability (3.3) or three-agents game unlinkability (3.4) holds, then all traces are equivalent, therefore also game-based unlinkability (see Definition 3.2.3) holds.

Now we need to prove that game-based unlinkability implies two-agents game unlinkability and three-agents game unlinkability. This implication trivially holds, since two and three-agents game unlinkability both require *specific* pairs of mappings to be equivalent while Definition 3.2.3 already implies the equivalence of *any* pair of mappings. □

A.2 Theorem 3.3.1 (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.

PROOF. In Chapter 3 we introduced three definitions of unlinkability: weak unlinkability, strong unlinkability and game-based unlinkability. We want to prove that these definitions coincide under the conditions listed in the theorem. We start by showing that weak unlinkability coincides to game-based unlinkability. By Theorem 3.2.3 we know that game-based unlinkability always implies weak unlinkability, therefore, in order to demonstrate their equivalence under the conditions, we only need to prove that weak unlinkability implies game-based unlinkability.

Game-based unlinkability holds when all the traces in T are equivalent under any attacker strategy $\sigma \in \Sigma$. Hence, we need to show that, under the conditions Unbounded number of agents, Renaming, Swapping, Extension I and II, weak unlinkability implies that

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

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), hence there is no trace equivalence 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, the mapping π must be equivalent to a mapping where the link is broken, and this can only correspond to π'. Thus all the mappings with length 2 are equivalent.

Inductive case. We have to show that all the mappings with length $n > 2$ are equivalent. The induction hypothesis gives that all the 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 and 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 to last and DE_2 at the second to last and third to 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 only have three sets and equivalence is transitive, this is sufficient to conclude that all the 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). All these mappings are equivalent:

  - The condition Extension I applied to the mappings with length 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 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 and condition Extension II.

FR Mappings ending in a transaction with a fresh agent (π^{-1}, y), $y \notin \pi^{-1}$. They are equivalent to each other according to the induction assumption and 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 to last transaction to position $n-3$ and twice more to move the last transaction to place $n-2$, 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) with $k + l = n - 1$. They 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 it is also equivalent to $\pi \sim (a_1^{k-1}, a_2, a_1^{n-k})$ by (ST), so $sw_k(\pi) \sim ({a_1}^k, a_2, a_1^{n-k-1})$ by the condition Swapping. But then $sw_k(\pi) \sim ({a_1}^n)$ by (ST).

These sets contain all the 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.

The second part of the proof consists in showing that weak and strong unlinkability coincide. By Theorem 3.2.1 we know that strong unlinkability

always implies weak unlinkability, hence we only need 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 of a given protocol are equivalent under any attacker strategy. The existence of an unbounded number of agents implies that there always exists a mapping where all the transactions are not linked, which is the requirement for strong unlinkability to hold. □

A.3 Theorem 3.3.2 (Unification of inseparability)

If a protocol guarantees the following conditions:

- *Renaming*
- *Extension II*

then it satisfies weak inseparability if and only if it satisfies strong inseparability.

PROOF. As for unlinkability, we expressed the notion of inseparability in a weak and a strong form. Here we show that under the conditions Renaming and Extension II they coincide. By Theorem 3.2.4 we already know that strong inseparability always implies weak inseparability (no condition needed). Hence, we only have to show that under the two aforementioned conditions, weak inseparability implies strong inseparability. In our abstract model this corresponds to showing that all the traces with the same length given by a protocol are equivalent. The proof is by induction on the number of transactions (n).

Base case. For $n = 2$ 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. We have to prove that all the mappings with length $n > 2$ are equivalent. The inductive hypothesis gives that all the traces $\tau \in \mathrm{T}$ such that $|\pi_\tau| < n$ 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, these traces are equivalent to the remaining traces $\tau \in \mathrm{T}$ such

that $|\pi_\tau| = n$ and $\pi_\tau(p_{n-1}) \neq \pi_\tau(p_n)$. By transitivity of $\sim$, all the traces with length n are equivalent. Therefore we can conclude that under the conditions Renaming and Extension II weak and strong inseparability coincide. □

A.4 Theorem 5.2.1 (Single-step protocols)

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. In Chapter 3 we presented several definitions of unlinkability and inseparability. We proved that, under the conditions listed above, all these properties coincide. Now we want to prove that the same conditions are satisfied by the class of single-step protocols in the model of Arapinis et al. [3]. Hence, a protocol in this class either guarantees all the properties or none of them.

A single-step protocol in the concrete model described in [3] provides traces of the form

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

where

- the indexes (i, j) are used to indicate the j-th query of the i-th transaction;
- $A_{i,j}$ corresponds to the process obtained from the process modelling the system after a tag output in the session (i, j);

- $x_{i,j}$ is the variable that contains the term sent over the network in the session (i, j).

We have to prove that single-step protocols as modelled in Section 5.1 guarantee all the conditions of Section 3.3.2.

First we define some notation and two lemmas used later on to prove the conditions.

A.4.1 Notation

We use $\Phi_{\pi,\sigma}(k)$ to denote the frame produced by a trace (π, σ) at the k-th transaction

$$\Phi_{\pi,\sigma}(k) = \nu\tilde{\rho}\prod_{i=1}^{\sigma_k} x_{k,i} : O_{I(k,i)}(U^{I(k,i)-1}(s_{\pi_k}))$$

where $I(k,i) = i + \sum_{\pi_j^{j<k}=\pi_k} \sigma_j$ is the number of sessions by the agent π_k in a trace up to the i-th session of the k-th transaction.
We use π^l and σ^l as short 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 agent and the $(l+1)$-st strategy, respectively.

A.4.2 Lemmas

The first lemma (Lemma A.4.1) simplifies the concept of trace equivalence between processes, so proving that two traces are equivalent reduces to proving that they produced the same inputs and outputs and that their final processes are statically equivalent while the original definition requires to prove the static equivalence of *each* intermediate process.

LEMMA A.4.1. *Consider two traces τ_A and τ_B ($|\tau_A| = |\tau_B|$) generated by two processes A and B in canonical form modelling an RFID system running a single-step protocol. If τ_A and τ_B process the same sequences of inputs and outputs and their last pair of processes are statically equivalent, then τ_A and τ_B are equivalent.*

PROOF. Consider two processes A_0 and B_0 in canonical form modelling a complete RFID system as defined in Section 5.1. Note that *any* process can be transformed in an equivalent process in canonical form. Their form corresponds to the one used in the definition of trace equivalence of Arapinis

et al. [3]. We use $\tilde{s}$ for the sequence of all the names restricted by the subprocesses modelling the tags in the system. Consider two traces τ_A and τ_B, generated by the processes A_0 and B_0:

$$\tau_A = A_0 \stackrel{\alpha_1}{\Longrightarrow} A_1 \stackrel{\alpha_2}{\Longrightarrow} \ldots \stackrel{\alpha_{n-1}}{\Longrightarrow} A_{n-1} \stackrel{\alpha_n}{\Longrightarrow} A_n$$

$$\tau_B = B_0 \stackrel{\alpha_1}{\Longrightarrow} B_1 \stackrel{\alpha_2}{\Longrightarrow} \ldots \stackrel{\alpha_{n-1}}{\Longrightarrow} B_{n-1} \stackrel{\alpha_n}{\Longrightarrow} B_n$$

We recall that the notion of trace equivalence given by Arapinis et al. [3] states that the traces τ_A and τ_B are equivalent ($\tau_A \sim_{tr} \tau_B$) if $A_i \approx_s B_i$ for all i. Each transition in the traces can only augment a frame, namely it can only add variables, but it cannot destroy them or change their content. $\varphi(A_i)$ always has all the variables in $\varphi(A_{i-1})$ and possibly more. Clearly, all the terms bounded to the variables in $\varphi(A_{i-1})$ do not contain any of the new variables present in the frame $\varphi(A_i)$. The same reasoning hold for the frames of the processes B_i. We want to prove that if the frames $\varphi(A_n)$ and $\varphi(B_n)$ are equivalent then the traces τ_A and τ_B are equivalent. In other words, if A_n is statically equivalent to B_n, then all the intermediate processes are statically equivalent:

$$A_n \approx_s B_n \Rightarrow \forall i : \phi(A_i) \approx_s \phi(B_i).$$

Each of the $\stackrel{\alpha}{\Longrightarrow}$-transitions $\alpha_1, \ldots, \alpha_n$ either produces a new substitution or does not change the frames of the corresponding processes. 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 simplicity, in the rest of the proof we only consider the transitions that produce a substitution. Consider the n-th frames

$$\varphi(A_n) = \nu\tilde{s}.\{\{^{T_1}/_{x_1}\}, \ldots, \{^{T_n}/_{x_n}\}\}$$

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

where T_i and T'_i indicate the terms produced by the i-th transitions of the traces τ_A and τ_B, respectively.

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 pairs of frames $(\varphi(A_i), \varphi(B_i))$ are statically equivalent. The inductive hypothesis, gives that

$$\varphi(A_i) \approx_s \varphi(B_i)$$

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 $\tilde{s}$ 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 equivalent frames

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

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

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

$\nu x_i.\{^{T_i}/_{x_i}\}$ and $\nu x_i.\{^{T'_i}/_{x_i}\}$ are structurally equivalent to the null process. We can also safely remove restrictions used by the i-th transaction only. Moreover, the parallel composition of a process P with the null process is structurally equivalent to the process P, therefore we obtain

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

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 A.4.2. *Let*

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

be frames in canonical form such that $\varphi_A \approx_s \varphi_B$, 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 $\varphi'_A = \varphi_A\Theta$ and $\varphi'_B = \varphi_B\Theta$ then $\varphi'_A \approx_s \varphi'_B$.

PROOF. Given two statically equivalent frames in the form of φ_A and φ_B, we want to prove that if we rename the variables $x_1, \ldots, x_m, x_{m+1}, \ldots, x_{m+n}$ into $x_{m+1}, \ldots, x_{m+n}, x_1, \ldots, x_m$ the resulting frames are still equivalent. By hypothesis we know that

$$\mathrm{dom}(\varphi_A) = \mathrm{dom}(\varphi_B) \wedge \forall M, N : (M = N)\varphi_A \text{ if and only if } (M = N)\varphi_B$$

and we want to prove that

$$\mathrm{dom}(\varphi'_A) = \mathrm{dom}(\varphi'_B) \wedge \forall M, N : (M = N)\varphi'_A \text{ if and only if } (M = N)\varphi'_B.$$

The domains of φ'_A and φ'_B coincide because Θ preserves the variable names, i.e. $\mathrm{dom}(\varphi_A) = \mathrm{dom}(\varphi'_A)$ and $\mathrm{dom}(\varphi_B) = \mathrm{dom}(\varphi'_B)$. From the hypothesis we know that

$$\forall M, N : M\varphi_A = N\varphi_A \text{ if and only if } M\varphi_B = N\varphi_B.$$

Since $M\Theta$ and $N\Theta$ are also terms and $\varphi_A \approx_s \varphi_B$, then we have that

$$\forall M, N : (M\Theta)\varphi_A = (N\Theta)\varphi_A \text{ if and only if } (M\Theta)\varphi_B = (N\Theta)\varphi_B.$$

But $\Theta\varphi_A$ is φ'_A and $\Theta\varphi_B$ is φ'_B therefore

$$\forall M, N : M\varphi'_A = N\varphi'_A \text{ if and only if } M\varphi'_B = N\varphi'_B.$$

Thus, we can conclude that $\varphi'_A \approx_s \varphi'_B$.

□

We are now ready to demonstrate the theorem.

A.4.3 Conditions

SSP: Condition Unbounded number of agents. *We say that a protocol has an unbounded number of agents if and only if*

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

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. *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 if and only if $\pi \sim \pi[a'/a]$ for all mappings π and agents $a' \notin A_\pi$.

We have to show that $\pi \sim \pi[s'/s]$ for all possible mappings π of traces in T and tags $s \in A_\pi$, $s' \notin A_\pi$. We use s as an agent in the RFID model is identified by its unique secret (typically denoted by s). In the applied pi calculus it is always possible to α-rename a name or a variable in a process without changing its semantics. In the model of [3], a trace $\tau = (\pi, \sigma) \in \mathrm{T}$ is a sequence of processes obtained by applying some sequence of $\stackrel{\alpha}{\Longrightarrow}$-transitions. α-renaming all the processes in τ to map the transactions executed by an agent s_i to an agent s_i' does not affect the $\stackrel{\alpha}{\Longrightarrow}$-transitions. Also, each α-renamed process is equivalent to the original one. Consider for example the frame of the last process produced by τ

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

If we α-rename all the occurrences of the secret s_i with $s_i' \notin A_\tau$ in the trace, then we obtain the following frame

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

By α-renaming, this frame is statically equivalent to the original one. We can apply the same reasoning to all the intermediate processes in τ and conclude that the resulting α-renamed trace is equivalent to τ. Since this hold for any attacker strategy σ, we can conclude that $\pi \sim \pi[{}^{s'}/_{s}]$.

SSP: Condition 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*

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

We say that a protocol satisfies the condition Swapping if and only if

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

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

Consider two equivalent mappings π_A and π_B. We have to prove that π_A^k is equivalent to π_B^k for any valid k. This means that we have to show that a single-step protocol produces equivalent traces (π_A^k, σ) and (π_B^k, σ) for all valid k and strategy σ. According to the definition of trace equivalence in [3] and by Lemma A.4.1, this consists in showing that (π_A^k, σ) and (π_B^k, σ) $(\forall k, \sigma)$ produce the same sequences of $\stackrel{\alpha}{\Longrightarrow}$-transitions and that their final processes are statically equivalent. Since (π_A^k, σ) and (π_B^k, σ) share the same attacker strategy σ (which corresponds to the number of queries executed for each transaction) and they are produced by a single-step protocol, they trivially lead to the same sequence of $\stackrel{\alpha}{\Longrightarrow}$-transactions, which correspond to as many message outputs on the public channel as queries in σ. We also need to show that all the pairs of swapped traces produce final processes that are statically equivalent, namely

$$\nu\tilde{s}.\nu\tilde{w}.\textstyle\prod_{l=1}^{n}\Phi_{\pi_A^k,\sigma}(l) \approx_{\mathrm{s}} \nu\tilde{s}.\nu\tilde{w}.\prod_{l=1}^{n}\Phi_{\pi_B^k,\sigma}(l)$$

for all valid k and σ. The frames $\Phi_{\pi_A^k,\sigma}(l)$ and $\Phi_{\pi_B^k,\sigma}(l)$ contain exactly the same terms produced by the traces (π_A, σ^k) and (π_B, σ^k), namely traces with the original mappings π_A and π_B and an attacker strategy σ^k that is obtained by swapping the attacker strategy $\sigma(k)$ with the attacker strategy $\sigma(k+1)$ in σ. Although the terms in (π_A^k, σ) and (π_A, σ^k) are the same, the ones involved in the swapping process are assigned to different variables, and similarly for (π_B^k, σ) and (π_B, σ^k). Therefore, we can equivalently express the final frames of the traces (π_A^k, σ) and (π_B^k, σ) in terms of the original mappings by applying a substitution function Θ that renames the variables $(x_{k,1}, \ldots, x_{k,\sigma_k}, x_{k+1,1}, \ldots, x_{k+1,\sigma_{k+1}})$ to $(x_{k+1,1}, \ldots, x_{k+1,\sigma_{k+1}}, x_{k,1}, \ldots, x_{k,\sigma_k})$. Clearly, the substitution only changes the variables produced by the k-th and the $k+1$-st transactions:

$$\Phi_{\pi_A,\sigma^k}(k+1)\Theta = \Phi_{\pi_A^k,\sigma}(k) \tag{A.1}$$

$$\Phi_{\pi_A,\sigma^k}(k)\Theta = \Phi_{\pi_A^k,\sigma}(k+1) \tag{A.2}$$

and

$$\Phi_{\pi_B,\sigma^k}(k+1)\Theta = \Phi_{\pi_B^k,\sigma}(k) \tag{A.3}$$

$$\Phi_{\pi_B,\sigma^k}(k)\Theta = \Phi_{\pi_B^k,\sigma}(k+1) \tag{A.4}$$

Furthermore, we have that

$$\Phi_{\pi_A,\sigma^k}(i) = \Phi_{\pi_A^k,\sigma}(i) \quad \forall i \in \{1, \ldots, k-1, k+2, \ldots, n\} \tag{A.5}$$

(and similarly for Φ_{π_B,σ^k} and $\Phi_{\pi_B^k,\sigma}$), i.e. the transactions that are not involved in the swapping process produce the same assignments in (π_A^k, σ) and (π_A, σ^k). Note that the final frames of (π_A, σ^k) and (π_B, σ^k) are in canonical form and the function Θ is in the form required by Lemma A.4.2. Therefore, we can use Lemma A.4.2 by applying Θ to the final frames of (π_A, σ^k) and (π_B, σ^k) to obtain the equivalence

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

Since Θ does not affect the first $k-1$ and the last $n-k+2$ transactions, we can move it as follows:

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

Using the equations (A.1)-(A.5), the above equivalence can be written as

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

This corresponds to

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

Therefore, all the traces obtained by swapping two equivalent mappings π_A and π_B produce final frames that are statically equivalent for any choice of k and σ. By Lemma A.4.1, we can conclude that all these pairs of traces are equivalent, and thus also the mappings π_A^k and π_B^k. Hence, single-step protocols always satisfy the condition Swapping.

SSP: Condition 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 and only if

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

for all mappings π, π'.

We have to show that two equivalent mappings π_A and π_B (mappings that generate equivalent traces under any attacker strategy for a given protocol) preserve the equivalence under any attacker strategy σ when we extend their domains by adding a new agent. Again, this implies proving that all the pairs of traces that can be constructed with the mappings $extn(\pi_A)$ and $extn(\pi_B)$ are trace equivalent. By lemma A.4.1, this reduces to show that these traces perform the same $\stackrel{\alpha}{\Longrightarrow}$-transitions and their final frames are equivalent. As explained before, two traces of a single-step protocol will produce the same sequence of outputs under any attacker strategy σ. Hence, to conclude the proof we have to demonstrate that

$$\forall \sigma : \nu\tilde{s}.\nu\tilde{w}.(\prod_{l=1}^{n+1}\Phi_{extn(\pi_A),\sigma}(l)) \approx_s \nu\tilde{s}.\nu\tilde{w}.(\prod_{l=1}^{n+1}\Phi_{extn(\pi_B),\sigma}(l)) \quad \text{(A.6)}$$

where $n = |\pi_A|$. The left hand side of the equivalence can be written as

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

because $\Phi_{\pi_A,\sigma}(i) = \Phi_{extn(\pi_A),\sigma}(i)\ \forall i \leq n$. We know that the last transaction, by hypothesis, belongs to a fresh new tag. This means that the names that are restricted in $\Phi_{extn(\pi_A),\sigma}(n+1)$ are not used in $\prod_{l=1}^{n}\Phi_{\pi_A,\sigma}(l)$ and vice versa. Therefore, we can separate the restrictions obtaining an equivalent frame

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

Similarly, the right hand side of the equivalence can be written as

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

We know by hypothesis that the frames of the final processes generated by traces with mappings π_A and π_B are equivalent. Formally:

$$\forall \sigma : \nu\tilde{s}.\nu\tilde{w}.(\prod_{l=1}^{n}\Phi_{\pi_A,\sigma}(l)) \approx_s \nu\tilde{s}.\nu\tilde{w}.(\prod_{l=1}^{n}\Phi_{\pi_B,\sigma}(l)) \quad \text{(A.7)}$$

Also, if we choose any attacker strategy for one transaction executed by an agent, it will produce the same knowledge that would have been produced by using the same strategy with another agent. In particular

$$\nu s.\nu w.\Phi_{extn(\pi_A),\sigma}(n+1) \approx_s \nu s.\nu w.\Phi_{extn(\pi_B),\sigma}(n+1). \quad \text{(A.8)}$$

By putting in parallel the processes in A.7 and A.8 we obtain the equivalence

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

that corresponds exactly to A.6. Therefore, $extn(\pi_A) \sim extn(\pi_B)$, thus single-step protocols satisfy condition Extension I.

SSP: Condition 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 and only if

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

for all mappings π, π'.

As in the previous proof, we have to show that extended mappings preserve trace equivalence. The difference is that the last transaction is not executed by a new tag, but by the same tag which executed the last transaction in the original traces. Formally, we want to prove that if $\pi_A \sim \pi_B$ then also $extl(\pi_A) \sim extl(\pi_B)$.
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

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

By hypothesis we know that $\pi_A \sim \pi_B$, therefore $(\pi_A, \sigma') \sim (\pi_B, \sigma')$. Since (π_A, σ') and (π_B, σ') produce the same $\stackrel{\alpha}{\Longrightarrow}$-transitions and final frames as $(extl(\pi_A), \sigma)$ and $(extl(\pi_B), \sigma)$, respectively, then $(extl(\pi_A), \sigma) \sim (extl(\pi_B), \sigma)$ for any attacker strategy σ. Hence, also condition Extension II is satisfied. □

We can conclude that all the forms of unlinkability and inseparability defined in Section 3.2 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 5.2.1 it also guarantees all the unlinkability and inseparability properties.

A.5 Theorem 5.3.7 (Unlinkability for SSP)

A single-step protocol satisfies unlinkability if and only if it satisfies both $\mathcal{P}_{SI}$ and $\mathcal{P}_{OI}$.

PROOF. In Chapter 5 we described the class of single-step protocols. Here, we prove that a single-step protocol satisfies the definition of unlinkability (Definition 4.2.1) given for our RFID model when it satisfies two conditions, $\mathcal{P}_{SI}$ and $\mathcal{P}_{OI}$, which are easier to prove. The proof consists of two parts: first, we prove that the conditions $\mathcal{P}_{SI}$ and $\mathcal{P}_{OI}$ imply unlinkability. Then, we demonstrate that when unlinkability holds also the conditions are satisfied.

Note that, since the reader and the backend database are not explicitly modelled, Definition 4.2.1 is greatly simplified. To prove that $\mathcal{P}_{SI}$ and $\mathcal{P}_{OI}$ imply Definition 4.2.1, it is sufficient to show that $C[Tag(c_1, c_2)] \approx C[Tag(c_1) \mid Tag(c_2)]$ with $C[\,] = \nu t.([\,] \mid \bar{t}\langle\_\rangle)$, as the subprocess $ReplTag$ and the restrictions $\widetilde{n}$ can always be added without affecting the equivalence of the processes. The resulting processes are simple and both can clearly perform the same transitions. The challenging part of the proof is to show that all pairs of frames produced by $C[Tag(c_1, c_2)]$ and $C[Tag(c_1) \mid Tag(c_2)]$ are statically equivalent.

Let $L = C[Tag(c_1, c_2)]$ and $R = C[Tag(c_1) \mid Tag(c_2)]$ be closed extended processes. Formally, we want to prove that $L \approx R$, that is

$$\nu t.(\nu w_0.\nu s_0.\big(InitSt(w_0, S_0) \mid !P(w_0, c_1) \mid !P(w_0, c_2)\big) \mid \bar{t}\langle\_\rangle)$$

is observationally equivalent ($\approx$) to

$$\begin{aligned}\nu t.(&\nu w_1.\nu s_1.\big(InitSt(w_1, S_1) \mid !P(w_1, c_1)\big) \mid \\ &\nu w_2.\nu s_2.\big(InitSt(w_2, S_2) \mid !P(w_2, c_2)\big) \mid \bar{t}\langle\_\rangle).\end{aligned}$$

Note that S_0, S_1 and S_2 are terms containing the secrets s_0, s_1 and s_2, respectively. They are used to model the internal state of three tags in the definition.

For the sake of simplicity, we use the equivalent notion of labelled bisimilarity ($\approx_l$) [1] instead of observational equivalence. Hence, we have to show that there exists a relation $\mathscr{R}$ between the processes L and R ($L \mathscr{R} R$) such that

1. $L \approx_s R$;
2. if $L \to L'$, then $\exists R'$ s.t. $R \to^* R'$ and $L' \mathscr{R} R'$ and vice versa;
3. if $L \xrightarrow{\alpha} L'$ and $\mathbf{fv}(\alpha) \subseteq \mathrm{dom}(L)$ and $\mathbf{bv}(\alpha) \cap \mathbf{fn}(R) = \emptyset$, then $\exists R'$ s.t. $R \to^* \xrightarrow{\alpha} \to^* R'$ and $L' \mathscr{R} R'$ and vice versa.

We use the following notation:

- We use $P_{i,j}$ as a short for $P(w_i, c_j)$:
  $P_{i,j} \triangleq c_j(\_).t(\_).w_i(x).\nu\widetilde{\rho}.\overline{c_j}\langle O(x)\rangle.(\overline{t}\langle\_\rangle \mid St(w_i, U(x)))$.

- We use $Q_{i,j}$ for the process $P_{i,j}$ that has been triggered on the public channel c_j:
  $Q_{i,j} \triangleq t(\_).w_i(x).\nu\widetilde{\rho}.\overline{c_j}\langle O(x)\rangle.(\overline{t}\langle\_\rangle \mid St(w_i, U(x)))$
  and $Q^k_{i,j}$ for k processes $Q_{i,j}$ in parallel.

- We use $\psi_L, \psi_{R1}, \psi_{R2}$ for frames with the following forms:

  - $\psi_L = \prod_{i=1}^{k+l} x_i : O_i(U^{i-1}(S_0))$
  - $\psi_{R1} = \prod_{i=1}^{k} x_{\alpha_i} : O_i(U^{i-1}(S_1))$
  - $\psi_{R2} = \prod_{j=1}^{l} x_{\beta_j} : O_j(U^{j-1}(S_2))$

  $\forall$ increasing sequences α_i, β_j s.t. $\{\alpha_i \mid 1 \leq i \leq k\} \cup \{\beta_j \mid 1 \leq j \leq l\} = [1, ..., k+l]$. The frame $\nu s_0, \widetilde{\rho}_{k+l}.\psi_L$ corresponds to the frame obtained after the execution of k sessions on the first interface and l sessions on the second (in any order). Similarly $\nu s_1, \widetilde{\rho}_k.\psi_{R1} \mid \nu s_2, \widetilde{\rho}_l.\psi_{R2}$ represents the corresponding frame for the independent interfaces.

- We define:

  - $LI(X) \triangleq \nu t, s_0, w_0, \widetilde{\rho}_{k+l}.(\psi_L \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid X)$ to describe the tag on two linked interfaces;
  - $II_1(X) \triangleq \nu s_1, w_1, \widetilde{\rho}_k.(\psi_{R1} \mid !P_{1,1} \mid Q^n_{1,1} \mid X)$ to describe the tag on the first of the two independent interfaces;
  - $II_2(X) \triangleq \nu s_2, w_2, \widetilde{\rho}_l.(\psi_{R2} \mid !P_{2,2} \mid Q^m_{2,2} \mid X)$ to describe the tag on the second of the two independent interfaces.

To describe the state of a system modelling a single-step protocol we need to model any possible pair of processes that may be obtained from L and R.

Using the notation above we can write, for instance, $LI(\overline{t}\langle\_\rangle \mid St(w_0, U^{k+l}(S_0)))$ to express a process modelling a single tag, which initiated n sessions on the first interface and m sessions on the second one, and completed k and l sessions, respectively, on these interfaces. Similarly, we can write $\nu t.(II_1(St(w_1, \sigma^k(x))) \mid II_2(St(w_2, \sigma^l(x))) \mid \overline{t}\langle\_\rangle)$ to describe the corresponding process with interfaces modelling two independent tags.

As other tags can be added by congruence, we only have to address the actions of the tags connected to the interfaces c_1 and c_2. In single-step protocols, tags do not input any information, hence their state is fully determined by the number of sessions they run. A tag state may depend on new names it introduced in previous sessions, so we keep track of those names in $\widetilde{\rho}$. Because of the token there can be only one session active at a time on either interface. However, the token does not prevent from triggering new sessions on $c_j(\_)$, thus we also need to account for the number of sessions triggered but not yet started. Finally, we need to model the knowledge that the attacker gains from the tag sessions (the output of each session, collected in ψ). Summarising, the state of a system is captured by (1) the attacker knowledge, (2) the number of sessions started on either interfaces (n and m, respectively), (3) the number of sessions that have completed on either interface (k and l, respectively), and (4) the state of the current session.

We are now ready to define the relation $\mathscr{R}$. Each pair of processes models a possible evolution of the process L as first element and the corresponding evolution of the process R as second element, i.e. the processes obtained after the execution of the same labelled transactions on L and R. For instance, the first element of the first pair (1.1) in $\mathscr{R}$ (see the next page) corresponds to a parallel composition of:

- the knowledge of the attacker (ψ_L);
- the replications of the protocol processes for the same tag with two different interfaces ($!P_{0,1}$ and $!P_{0,2}$);
- the processes spawned by the replications that have been triggered, but did not consume the token yet ($Q_{0,1}^n$ and $Q_{0,2}^m$); n and m denote the numbers of processes triggered on the first and second interface, respectively;
- the token ($\bar{t}\langle\_\rangle$);
- the content of the tag memory ($St(w_0, U^{k+l}(S_0))$).

$$
\begin{aligned}
&\mathscr{R} = \{ \\
&(LI(\overline{t}\langle\_\rangle \mid St(w_0, U^{k+l}(S_0))), && (1.1)\\
&\ \nu t.(II_1(St(w_1, U^k(S_1))) \mid II_2(St(w_2, U^l(S_2))) \mid \overline{t}\langle\_\rangle)), && (1.2)\\
&\\
&(LI(St(w_0, U^{k+l}(S_0)) \mid \\
&\quad w_0(x).\nu\widetilde{\rho}.\overline{c_1}\langle O(x)\rangle.(\overline{t}\langle\_\rangle \mid St(w_0, U(x)))), && (2.1)\\
&\ \nu t.(II_1(St(w_1, U^k(S_1)) \mid \\
&\quad w_1(x).\nu\widetilde{\rho}.\overline{c_1}\langle O(x)\rangle.(\overline{t}\langle\_\rangle \mid St(w_1, U(x)))) \mid \\
&\quad II_2(St(w_2, U^l(S_2))))), && (2.2)\\
&\\
&(LI(St(w_0, U^{k+l}(S_0)) \mid \\
&\quad w_0(x).\nu\widetilde{\rho}.\overline{c_2}\langle O(x)\rangle.(\overline{t}\langle\_\rangle \mid St(w_0, U(x)))), && (3.1)\\
&\ \nu t.(II_1(St(w_1, U^k(S_1))) \mid \\
&\quad II_2(St(w_2, U^l(S_2)) \mid \\
&\quad w_2(x).\nu\widetilde{\rho}.\overline{c_2}\langle O(x)\rangle.(\overline{t}\langle\_\rangle \mid St(w_2, U(x))))), && (3.2)\\
&\\
&(LI(\nu\widetilde{\rho}.\overline{c_1}\langle O(U^{k+l}(S_0))\rangle.(\overline{t}\langle\_\rangle \mid St(w_0, U(U^{k+l}(S_0))))), && (4.1)\\
&\ \nu t.(II_1(\nu\widetilde{\rho}.\overline{c_1}\langle O(U^k(S_1))\rangle.(\overline{t}\langle\_\rangle \mid St(w_1, U(U^k(S_1))))) \mid \\
&\quad II_2(St(w_2, U^l(S_2))))), && (4.2)\\
&\\
&(LI(\nu\widetilde{\rho}.\overline{c_2}\langle O(U^{k+l}(S_0))\rangle.(\overline{t}\langle\_\rangle \mid St(w_0, U(U^{k+l}(S_0))))), && (5.1)\\
&\ \nu t.(II_1(St(w_1, U^k(S_1))) \mid \\
&\quad II_2(\nu\widetilde{\rho}.\overline{c_2}\langle O(U^l(S_2))\rangle.(\overline{t}\langle\_\rangle \mid St(w_2, U(U^l(S_2)))))) && (5.2)\\
&\forall k, l \in \mathbb{N} \\
&\}
\end{aligned}
$$

The process L corresponds to (1.1) for $\psi_L = \emptyset, n = m = k = l = 0$, and reaches this form after every complete protocol execution on one of the linked interfaces c_1, c_2. (1.2) models the same situation, but for independent interfaces.

The processes of the second and third pairs model the processes resulting after the token has been consumed to run a session on c_1 and on c_2, respectively.

The last two pairs in the relation model processes where a tag (on the first and on the second interface, respectively) has read its internal memory and is about to output the identification message on the public channel.

To complete this part of the proof we have to show that the relation $\mathscr{R}$ is a bisimulation and all its pairs are statically equivalent.
$\mathscr{R}$ is a bisimulation if any reduction step that may be performed by one element of a pair in $\mathscr{R}$ can be matched by the corresponding other element, and the resulting processes are in $\mathscr{R}$.

The only possible reduction steps for the first pair in $\mathscr{R}$ (1.1) and (1.2) are:

1. The token t synchronises with $Q^n_{0,1}$ and $Q^n_{1,1}$ in (1.1) and (1.2), respectively;

2. The token t synchronises with $Q^m_{0,2}$ and $Q^m_{2,2}$ in (1.1) and (1.2), respectively.

After step (1.), the processes (1.1) and (1.2) reduce to processes with the forms of (2.1) and (2.2), respectively. Similarly, after step (2.), they reduce to the forms of (3.1) and (3.2).

The processes of the second pair (2.1) and (2.2) can only synchronise the input and output processes on w_0 and w_1, respectively, because the token prevents other processes from starting a new session. This reduction creates a new pair of processes of the forms of (4.1) and (4.2). Similarly, the third pair would reduce to (5.1) and (5.2) after the communication on w_0 and w_2.

The fourth pair consists of the processes (4.1) and (4.2), that can only perform an output on the public channel c_1. The restrictions in $\nu\widetilde{\rho}$ can be moved to the beginning of the resulting processes by structural equivalence. The substitution, which bounds the output (i.e. the tag message) to a new variable, becomes part of the frames ψ_L and ψ_{R1}, respectively. The resulting processes have the forms of (1.1) and (1.2). Similarly, the last case reduces to the first pair in $\mathscr{R}$.

Note that in all these cases we should also consider that the processes might spawn a copy of $Q_{0,1}$, $Q_{1,1}$, $Q_{0,2}$ or $Q_{2,2}$. All the resulting pairs of processes trivially belong to $\mathscr{R}$. In fact, they preserve their form, causing only the exponents of corresponding processes to increase by one.

To complete our proof we have to demonstrate that all pairs in $\mathscr{R}$ contain processes that are statically equivalent. To do so, we have to show that their frames $\nu s_0, \widetilde{\rho}_{k+l}.(\psi_L)$ and $\nu s_1, \widetilde{\rho}_k.(\psi_{R1}) \mid \nu s_2, \widetilde{\rho}_l.(\psi_{R2})$ are equivalent. In the proof we use a property of $\mathcal{P}_{OI}$, which allows us to turn a frame of

the form $\nu s, \widetilde{\rho}_n. \prod_{i=1}^{n} x_i : O_i(U^{i-1}(S_0))$ into a new frame $\prod_{i=1}^{n} \nu s, \widetilde{\rho}_i.x_i : O_i(U^{i-1}(S_0))$, because

$$
\begin{aligned}
&\nu s, \widetilde{\rho}_n.\textstyle\prod_{i=1}^{n} x_i : O_i(U^{i-1}(S_0)) \\
\approx_s &\nu s, \widetilde{\rho}_n.(\textstyle\prod_{i=1}^{n-1} x_i : O_i(U^{i-1}(S_0)) \mid x_n : O_n(U^{n-1}(S_0))) \\
\approx_s &\nu s, \widetilde{\rho}_{n-1}.\textstyle\prod_{i=1}^{n-1} x_i : O_i(U^{i-1}(S_0)) \mid \nu s, \widetilde{\rho}_n.x_n : O_n(U^{n-1}(S_0)) \\
\approx_s &\nu s, \widetilde{\rho}_{n-2}.\textstyle\prod_{i=1}^{n-2} x_i : O_i(U^{i-1}(S_0)) \mid \nu s, \widetilde{\rho}_{n-1}.x_{n-1} : O_{n-1}(U^{n-2}(S_0)) \mid \\
&\nu s, \widetilde{\rho}_n.x_n : O_n(U^{n-1}(S_0)) \\
\approx_s &\cdots \\
\approx_s &\textstyle\prod_{i=1}^{n} \nu s, \widetilde{\rho}_i.x_i : O_i(U^{i-1}(S_0))
\end{aligned}
$$

Using this property and $\mathcal{P}_{SI}$, we can conclude that

$$
\begin{aligned}
\nu s_0, \widetilde{\rho}_{k+l}.\psi_L = \quad & \nu s_0, \widetilde{\rho}_{k+l}.\textstyle\prod_{i=1}^{k+l} x_i : O_i(U^{i-1}(S_0)) \\
\approx_s [\mathcal{P}_{OI}] \quad & \textstyle\prod_{i=1}^{k+l} \nu s_0, \widetilde{\rho}_i.x_i : O_i(U^{i-1}(S_0)) \\
\approx_s [\mathcal{P}_{SI}] \quad & \textstyle\prod_{i=1}^{k} \nu s_0, \widetilde{\rho}_i.x_{\alpha_i} : O_i(U^{i-1}(S_0)) \mid \\
& \textstyle\prod_{j=1}^{l} \nu s_0, \widetilde{\rho}_i.x_{\beta_j} : O_j(U^{j-1}(S_0)) \\
\approx_s [\mathcal{P}_{OI}] \quad & \nu s_0, \widetilde{\rho}_k.\textstyle\prod_{i=1}^{k} x_{\alpha_i} : O_i(U^{i-1}(S_0)) \mid \\
& \nu s_0, \widetilde{\rho}_l.\textstyle\prod_{j=1}^{l} x_{\beta_j} : .O_j(U^{j-1}(S_0)) \\
\approx_s [\alpha\text{-ren.}] \quad & \nu s_1, \widetilde{\rho}_k.\textstyle\prod_{i=1}^{k} x_{\alpha_i} : O_i(U^{i-1}(S_1)) \mid \\
& \nu s_2, \widetilde{\rho}_l.\textstyle\prod_{j=1}^{l} x_{\beta_j} : O_j(U^{j-1}(S_2)) \\
= \quad & \nu s_1, \widetilde{\rho}_k.(\psi_{R1}) \mid \nu s_2, \widetilde{\rho}_l.(\psi_{R2})
\end{aligned}
$$

$\forall$ increasing sequences α_i, β_j s.t. $\{\alpha_i \mid 1 \leq i \leq k\} \cup \{\beta_j \mid 1 \leq j \leq l\} = [1, ..., k+l]$.
Therefore, each pair in $\mathscr{R}$ contains statically equivalent processes. This concludes the first part of the proof.

The second part of the proof consists in showing that if unlinkability holds for a single-step protocol then $\mathcal{P}_{SI}$ and $\mathcal{P}_{OI}$ also hold. We prove its contrapositive: when one of the properties is violated, unlinkability does not hold. There are two cases:

1. If $\mathcal{P}_{SI}$ does not hold, then there exists k and l ($k < l$) such that

   $$\nu s, \widetilde{\rho}_k.x : O_k(U^{k-1}(S_0)) \not\approx_s \nu s, \widetilde{\rho}_l.x : O_l(U^{l-1}(S_0)).$$

   If the attacker queries $k-1$ times the first interface, $l-k$ the second, and then again the first, he obtains a frame with

   $$\nu s_0, \widetilde{\rho}_l.x_l : O_l(U^{l-1}(S_0))$$

if the interfaces are independent and

$$\nu s_1, \widetilde{\rho}_k.x_l : O_k(U^{k-1}(S_1))$$

if the frames are linked. These frames are not equivalent by hypothesis, thus the processes in the definition of unlinkability are not observationally equivalent.

2. If $\mathcal{P}_{OI}$ does not hold, then there exists n such that

$$\prod_{i=1}^{n-1} x_i : O_i(U^{i-1}(S_0)) \not\approx_{s,\widetilde{\rho}_n} x_n : O_n(U^{n-1}(S_0)).$$

If $\mathcal{P}_{SI}$ does not hold then unlinkability is violated (see case 1). If $\mathcal{P}_{SI}$ holds then the attacker can query n times the first interface and once the second interface, obtaining the frame

$$\nu s_0, \widetilde{\rho}_n.(x_1 : O_1(S_0) \mid \ldots \mid x_{n-1} : O_{n-1}(U^{n-2}(S_0)) \mid x_n : O_n(U^{n-1}(S_0)))$$

if the interfaces are linked, and

$$\nu s_1, \widetilde{\rho}_{n-1}.(x_1 : O_1(S_1) \mid \ldots \mid x_{n-1} : O_{n-1}(U^{n-2}(S_1))) \mid \nu s_2, \widetilde{\rho}_n.x_n : O_n(S_2)$$

if the interfaces are independent. By $\mathcal{P}_{SI}$, the last frame is equivalent to

$$\nu s_1, \widetilde{\rho}_{n-1}.(x_1 : O_1(S_1) \mid \ldots \mid x_{n-1} : O_{n-1}(U^{n-2}(S_1))) \mid \nu s_2, \widetilde{\rho}_n.x_n : O_n(U^{n-1}(S_2)).$$

By hypothesis we know that these frames are not statically equivalent. Therefore the definition of unlinkability is violated.

Thus, we can conclude that for single-step protocols the properties $\mathcal{P}_{SI}$ and $\mathcal{P}_{OI}$ are necessary and sufficient conditions for unlinkability. □

A.6 Theorem 5.3.9 (Forward privacy for SSP)

A single-step protocol satisfies forward privacy (Definition 4.3.1) if and only if it satisfies $\mathcal{P}_{SI}$ and $\mathcal{P}_{FI}$.

The proof is similar to the one for unlinkability. Note that Theorem 5.3.9 and Proposition 5.3.8 imply that forward privacy is stronger than unlinkability for single-step protocols, as expected by Proposition 4.3.1.

PROOF. Again, the proof is divided in two parts. First, we prove that the conditions $\mathcal{P}_{SI}$ and $\mathcal{P}_{FI}$ imply forward privacy by demonstrating that $C[BrTag^f(c_1, c_2)]$ is equivalent to $C[BrTag^f(c_1) \mid Tag(c_2)]$ for $C[\,] = \nu t.([\,] \mid \overline{t}\langle\_\rangle)$. Later we prove that forward privacy implies the conditions.

In the proof we use the process $Break^f(w)$ with two meanings, namely $br(\_).t(\_).w(x).\overline{br}\langle x\rangle$ and $t(\_).w(x).\overline{br}\langle x\rangle$. These processes model the ability of the attacker to get the secret of the tag linked to the first interface, before and after sending an input on the channel br. We have to prove that when a single-step protocol satisfies $\mathcal{P}_{SI}$ and $\mathcal{P}_{FI}$ then

$$\nu t.(\nu w_0.\nu s_0.\big(InitSt(w_0, S_0) \mid !P(w_0, c_1) \mid !P(w_0, c_2) \mid Break^f(w)\big) \mid \overline{t}\langle\_\rangle)$$

is observationally equivalent ($\approx$) to

$$\begin{aligned}\nu t.(&\nu w_1.\nu s_1.\big(InitSt(w_1, S_1) \mid !P(w_1, c_1) \mid Break^f(w)\big) \mid \\ &\nu w_2.\nu s_2.\big(InitSt(w_2, S_2) \mid !P(w_2, c_2)\big) \mid \overline{t}\langle\_\rangle)\end{aligned}$$

For simplicity, we prove that the processes are labelled bisimilar, which implies that they are observationally equivalent. We define the relation $\mathscr{R}$ as

$\mathscr{R} = \{$

$$\begin{aligned}&(LI(\overline{t}\langle\_\rangle \mid St(w_0, U^{k+l}(S_0)) \mid Break^f(w_0)), && (1.1)\\ &\ \nu t.(II_1(St(w_1, U^k(S_1)) \mid Break^f(w_1)) \mid \\ &\quad II_2(St(w_2, U^l(S_2))) \mid \overline{t}\langle\_\rangle)), && (1.2)\end{aligned}$$

$$\begin{aligned}&(LI(St(w_0, U^{k+l}(S_0)) \mid w_0(x).\overline{br}\langle x\rangle), && (2.1)\\ &\ \nu t.(II_1(St(w_1, U^k(S_1)) \mid w_1(x).\overline{br}\langle x\rangle) \mid \\ &\quad II_2(St(w_2, U^l(S_2))))), && (2.2)\end{aligned}$$

$$\begin{aligned}&(LI(\overline{br}\langle U^{k+l}(S_0)\rangle), && (3.1)\\ &\ \nu t.(II_1(\overline{br}\langle U^k(S_1)\rangle) \mid II_2(St(w_2, U^l(S_2))))), && (3.2)\end{aligned}$$

$$(LI(\{^{U^{k+l}(S_0)}/_{x_{k+l}}\}), \tag{4.1}$$
$$\nu t.(II_1(\{^{U^{k}(S_1)}/_{x_{k+l}}\}) \mid II_2(St(w_2, U^l(S_2))))), \tag{4.2}$$

$$(LI(St(w_0, U^{k+l}(S_0)) \mid$$
$$w_0(x).\nu\widetilde{\rho}.\overline{c_1}\langle O(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_0, U(x))) \mid Break^f(w_0)), \tag{5.1}$$
$$\nu t.(II_1(St(w_1, U^k(S_1)) \mid$$
$$w_1(x).\nu\widetilde{\rho}.\overline{c_1}\langle O(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_1, U(x))) \mid Break^f(w_1)) \mid$$
$$II_2(St(w_2, U^l(S_2))))), \tag{5.2}$$

$$(LI(St(w_0, U^{k+l}(S_0)) \mid$$
$$w_0(x).\nu\widetilde{\rho}.\overline{c_2}\langle O(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_0, U(x)) \mid Break^f(w_0))), \tag{6.1}$$
$$\nu t.(II_1(St(w_1, U^k(S_1)) \mid Break^f(w_1)) \mid$$
$$II_2(St(w_2, U^l(S_2)) \mid$$
$$w_2(x).\nu\widetilde{\rho}.\overline{c_2}\langle O(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_2, U(x))))), \tag{6.2}$$

$$(LI(\nu\widetilde{\rho}.\overline{c_1}\langle O(U^{k+l}(S_0))\rangle.$$
$$(\bar{t}\langle\_\rangle \mid St(w_0, U(U^{k+l}(S_0)))) \mid Break^f(w_0)), \tag{7.1}$$
$$\nu t.(II_1(\nu\widetilde{\rho}.\overline{c_1}\langle O(U^{k}(S_1))\rangle.$$
$$(\bar{t}\langle\_\rangle \mid St(w_1, U(U^{k}(S_1)))) \mid Break^f(w_1)) \mid$$
$$II_2(St(w_2, U^l(S_2))))), \tag{7.2}$$

$$(LI(\nu\widetilde{\rho}.\overline{c_2}\langle O(U^{k+l}(S_0))\rangle.$$
$$(\bar{t}\langle\_\rangle \mid St(w_0, U(U^{k+l}(S_0)))) \mid Break^f(w_0)), \tag{8.1}$$
$$\nu t.(II_1(St(w_1, U^k(S_1)) \mid Break^f(w_1)) \mid$$
$$II_2(\nu\widetilde{\rho}.\overline{c_2}\langle O(U^l(S_2))\rangle.(\bar{t}\langle\_\rangle \mid St(w_2, U(U^l(S_2)))))))) \tag{8.2}$$

$\forall k, l \in \mathbb{N}$
}

We have to prove that any reduction step that may be performed by one element of a pair in $\mathscr{R}$ can be matched by its corresponding element, and the resulting processes must be in $\mathscr{R}$. Also, we have to show that all the pairs contain statically equivalent processes.

We start from the first pair $((1.1),(1.2))$ in the relation, which also corresponds to the initial state of the system. The processes can perform three different transitions, by synchronising the token with the processes modelling the tags on the interfaces c_1 and c_2, or with $Break^f$. These transitions lead to the forms of $((5.1),(5.2))$, $((6.1),(6.2))$ and $((2.1),(2.2))$, respectively, which are in $\mathscr{R}$.

In the processes $((2.1),(2.2))$ the token is not available, therefore the only possible transition is an input on the channel w_0, which leads to the processes $((3.1),(3.2))$. Such processes can only output the tag state on the public channel br. After this transition, we obtain two new processes of the form of $((4.1),(4.2))$. Their frames ψ_L and ψ_{R1} are extended with the substitutions $x_{k+l} : U^{k+l}(S_0)$ and $x_{k+l} : U^k(S_1)$, respectively. For the fourth pair no transition is possible, since the token is no longer available for any process.

Both processes of the fifth pair can only synchronise the input and output processes on w_0 and w_1, respectively. The resulting processes have the form of $((7.1),(7.2))$, respectively. The case of the pair $((6.1),(6.2))$ is similar.

The processes in $((7.1),(7.2))$ and $((8.1),(8.2))$ can only output a message on the channel c_1 and c_2, respectively. The resulting pairs have the form of $((1.1),(1.2))$. Note that the position of the token in the resulting processes differ from the one in $((1.1),(1.2))$, but the processes are structurally equivalent.

In the above cases we should also consider that the processes might spawn a copy of $Q_{0,1}$ and $Q_{1,1}$ or $Q_{0,2}$ and $Q_{2,2}$. They trivially belong to the relation, because they only increase the powers of the corresponding processes. Moreover, when $Break^f(w)$ is $br(\_).t(\_).w(x).\overline{br}\langle x\rangle$, the process can be triggered becoming $t(\_).w(x).\overline{br}\langle x\rangle$, but this transition, due to our notation, would lead to processes of the same form.

To complete the proof we have to show that the frames of the processes in each pair are equivalent. All the static equivalences to prove for the processes in $\mathscr{R}$, except for the ones in $((4.1),(4.2))$, have the form

$$\nu s_0, \widetilde{\rho}_{k+l}.(\psi_L) \approx_s \nu s_1, \widetilde{\rho}_k.(\psi_{R1}) \mid \nu s_2, \widetilde{\rho}_l.(\psi_{R2})$$

This equivalence has been proven to hold when $\mathcal{P}_{SI}$ and $\mathcal{P}_{OI}$ hold. Being $\mathcal{P}_{FI}$ stronger than $\mathcal{P}_{OI}$, we can conclude that these frames are statically equivalent. Thus, we only need to prove the static equivalence of the frames of the processes (4.1) and (4.2):

$$\nu s_0, \widetilde{\rho}_{k+l}.(\psi_L \mid x_{k+l} : U^{k+l}(S_0)) \approx_s \nu s_1, \widetilde{\rho}_k.(\psi_{R1} \mid x_{k+l} : U^k(S_1)) \mid \nu s_2, \widetilde{\rho}_l.\psi_{R2}$$

This can be proven using the properties $\mathcal{P}_{SI}$ and $\mathcal{P}_{FI}$ (and $\mathcal{P}_{OI}$ that is implied by $\mathcal{P}_{FI}$):

$$
\begin{array}{ll}
\nu s_0, \widetilde{\rho}_{k+l}.(\psi_L \mid x_{k+l}{:}U^{k+l}(S_0)) & \\
= & \nu s_0, \widetilde{\rho}_{k+l}.(\prod_{i=1}^{k+l} x_i{:}O_i(U^{i-1}(S_0)) \mid x_{k+l}{:}U^{k+l}(S_0)) \\
\approx_s[\mathcal{P}_{FI}] & \nu s_0, \widetilde{\rho}_{k+l}.(\prod_{i=1}^{k+l} x_i{:}O_i(U^{i-1}(S_0))) \mid \nu s_0, \widetilde{\rho}_{k+l}.x_{k+l}{:}U^{k+l}(S_0) \\
\approx_s[\mathcal{P}_{OI}] & \prod_{i=1}^{k+l} \nu s_0, \widetilde{\rho}_i.x_i{:}O_i(U^{i-1}(S_0)) \mid \nu s_0, \widetilde{\rho}_{k+l}.x_{k+l}{:}U^{k+l}(S_0) \\
\approx_s[\mathcal{P}_{SI}] & \prod_{i=1}^{k} \nu s_0, \widetilde{\rho}_i, x_{\alpha_i}{:}O_i(U^{i-1}(S_0)) \mid \\
 & \prod_{j=1}^{l} \nu s_0, \widetilde{\rho}_j.x_{\beta_j}{:}O_j(U^{j-1}(S_0)) \mid \nu s_0, \widetilde{\rho}_{k+l}.x_{k+l}{:}U^{k}(S_0) \\
\approx_s[\mathcal{P}_{OI}] & \nu s_0, \widetilde{\rho}_k.(\prod_{i=1}^{k} x_{\alpha_i}{:}O_i(U^{i-1}(S_0)) \mid x_{k+l}{:}U^{k}(S_0)) \mid \\
 & s_0, \widetilde{\rho}_l.\prod_{j=1}^{l} x_{\beta_j}{:}O_j(U^{j-1}(S_0)) \\
\approx_s[\alpha\text{-ren.}] & \nu s_1, \widetilde{\rho}_k.(\prod_{i=1}^{k} x_{\alpha_i}{:}O_i(U^{i-1}(S_1)) \mid x_{k+l}{:}U^{k+l}(S_1)) \mid \\
 & \nu s_2, \widetilde{\rho}_l.\prod_{j=1}^{l} x_{\beta_j}{:}O_j(U^{j-1}(S_2)) \\
= & \nu s_1, \widetilde{\rho}_k.(\psi_{R1} \mid x_{k+l}{:}U^{k}(S_1)) \mid \nu s_2, \widetilde{\rho}_l.\psi_{R2}
\end{array}
$$

$\forall$ increasing sequences α_i, β_j s.t. $\{\alpha_i \mid 1 \leq i \leq k\} \cup \{\beta_j \mid 1 \leq j \leq l\} = [1, ..., k+l]$. Therefore, all the frames in each pair of $\mathscr{R}$ are statically equivalent.

To conclude the proof we still need to prove that if forward privacy holds also $\mathcal{P}_{SI}$ and $\mathcal{P}_{FI}$ hold. We prove this by showing that $\neg\mathcal{P}_{SI} \vee \neg\mathcal{P}_{FI}$ implies

$$C[BrTag^f(c_1, c_2)] \not\approx C[BrTag^f(c_1) \mid Tag(c_2)].$$

There are two cases:

1. If $\mathcal{P}_{SI}$ does not hold, see proof of Theorem 5.3.7.

2. If $\mathcal{P}_{FI}$ does not hold then there exists n such that

$$\prod_{i=1}^{n-1} x_i{:}O_i(U^{i-1}(S_0)) \;\not\sim_{s,\widetilde{\rho}_n}\; x_n{:}U^{n-1}(S_0).$$

   If $\mathcal{P}_{SI}$ does not hold then forward privacy is violated (see case 1). If $\mathcal{P}_{SI}$ holds, the attacker can still break forward privacy by querying n times the second interface and then breaking the tag on the first interface. The resulting frames are

$$\nu s_0, \widetilde{\rho}_n.(x_1{:}O_1(S_0) \mid \ \ldots\ \mid x_{n-1}{:}O_{n-1}(U^{n-2}(S_0)) \mid x_n{:}U^{n-1}(S_0))$$

   and

$$\nu s_2, \widetilde{\rho}_{n-1}.(x_0{:}O_1(S_2) \mid \ \ldots\ \mid x_{n-1}{:}O_{n-1}(U^{n-2}(S_2))) \mid \nu s_1.x_n{:}U^0(S_1)$$

that, by $\mathcal{P}_{SI}$ corresponds to

$$\nu s_2, \widetilde{\rho}_{n-1}.(x_0 : O_1(S_2) \mid \ldots \mid x_{n-1} : O_{n-1}(U^{n-2}(S_2))) \mid \nu s_1.x_n : U^{n-1}(S_1).$$

By assumption we know that these frames are not statically equivalent, therefore the definition of forward privacy is violated.

□

A.7 Theorem 5.3.11 (Backward privacy for SSP)

A single-step protocol satisfies backward privacy (Definition 4.4.1) if and only if it satisfies $\mathcal{P}_{SI}$ and $\mathcal{P}_{BI}$.

PROOF. First we prove that $\mathcal{P}_{SI}$ and $\mathcal{P}_{BI}$ imply backward privacy, namely the equivalence

$$C[BrTag^b(c_1, c_2)] \approx C[BrTag^b(c_1) \mid Tag(c_2)]$$

with $C[\,] = \nu t, rc, c_2.([\,] \mid \bar{t}\langle\_\rangle)$. This corresponds to proving that

$$\nu rc, t, c_2.(\nu w_0.\nu s_0.\big(InitSt(w_0, S_0) \mid !P(w_0, c_1) \mid !P(w_0, c_2) \mid Break^b(w_0)\big) \mid \bar{t}\langle\_\rangle)$$

is observationally equivalent ($\approx$) to

$$\nu rc, t, c_2.(\nu w_1.\nu s_1.\big(InitSt(w_1, S_1) \mid !P(w_1, c_1) \mid Break^b(w_1)\big) \mid \nu w_2.\nu s_2.\big(InitSt(w_2, S_2) \mid !P(w_2, c_2)\big) \mid \bar{t}\langle\_\rangle).$$

This equivalence can be proven by demonstrating that these processes are labelled bisimilar. The relation $\mathscr{R}$ between the processes is similar to the one that we defined in the proofs of Theorems 5.3.7 and 5.3.9, hence we do not explicitly define it here. It consists of:

- Pairs of processes similar to those defined in the proof of Theorem 5.3.9 for forward privacy. The only difference is in the process $Break^b$ (that replaces $Break^f$). The pairs $((2.1), (2.2))$, $((3.1), (3.2))$ and $((4.1), (4.2))$ must be replaced by the processes modelling the evolution of the process $Break^b$, which lead to the disclosure of the secret on c_1 (and consequent restoring of the tag secret) or not, according to the attacker's choice. In both cases, the interface c_2 is enabled.

- Pairs of processes as defined in the proof of Theorem 5.3.7 for unlinkability, modelling the processes obtainable after the complete execution of $Break^b$. Note that all these processes must also contain the substitution $\{^{U^{d-1}(S_0)}/_y\}$.

All the frames (obtained by these pairs of processes) that model the system before the state disclosure, or when the attacker chooses not to break the tag, are proven equivalent in the proof of Theorem 5.3.7. The frames corresponding to the processes that have just output the state of the tag on c_1 are proven equivalent in the proof of Theorem 5.3.9. We only need to prove the equivalence of the frames corresponding to processes that have sent some outputs after the tag state disclosure ($k + l \geq n$):

$$\nu s_0, \widetilde{\rho}_{k+l}.(\psi_L \mid y{:}U^{d-1}(S_0)) \approx_s \nu s_1, \widetilde{\rho}_k.(\psi_{R1} \mid y{:}U^{d-1}(S_1)) \mid \nu s_2, \widetilde{\rho}_l.\psi_{R2}$$

This can be proven using the conditions $\mathcal{P}_{SI}$ and $\mathcal{P}_{BI}$ (and $\mathcal{P}_{OI}$ that is implied by $\mathcal{P}_{BI}$):

$$\begin{array}{ll}
\nu s_0, \widetilde{\rho}_{k+l}.(\psi_L \mid y{:}U^{d-1}(S_0)) & \\
= & \nu s_0, \widetilde{\rho}_{k+l}.(\prod_{i=1}^{n-2} x_i{:}O_i(U^{i-1}(S_0)) \mid y{:}U^{d-1}(S_0) \\
& \mid \prod_{i=n}^{k+l} x_i{:}O_i(U^{i-1}(S_0))) \\
\approx_s[\mathcal{P}_{OI}\ \mathcal{P}_{BI}] & \prod_{i=1}^{n-2} \nu s_0, \widetilde{\rho}_i.x_i{:}O_i(U^{i-1}(S_0)) \mid \nu s_0, \widetilde{\rho}_d.y{:}U^{d-1}(S_0) \mid \\
& \prod_{i=n}^{k+l} \nu s_0, \widetilde{\rho}_i.x_i{:}O_i(U^{i-1}(S_0)) \\
\approx_s[\mathcal{P}_{SI}] & \prod_{i=1}^{k-1} \nu s_0, \widetilde{\rho}_i.x_{\alpha_i}{:}O_i(U^{i-1}(S_0)) \mid \\
& \prod_{j=1}^{l} \nu s_0, \widetilde{\rho}_j.x_{\beta_j}{:}O_j(U^{j-1}(S_0)) \mid \nu s_0, \widetilde{\rho}_d.y{:}U^{d-1}(S_0) \\
\approx_s[\mathcal{P}_{BI}] & \nu s_0, \widetilde{\rho}_k.(\prod_{i=1}^{k-1} x_{\alpha_i}{:}O_i(U^{i-1}(S_0)) \mid y{:}U^{d-1}(S_0)) \\
& \mid s_0, \widetilde{\rho}_l.\prod_{j=1}^{l} x_{\beta_j}{:}O_j(U^{j-1}(S_0)) \\
\approx_s[\alpha\text{-ren.}] & \nu s_1, \widetilde{\rho}_k.(\prod_{i=1}^{k-1} x_{\alpha_i}{:}O_i(U^{i-1}(S_1)) \mid y{:}U^{d-1}(S_1)) \\
& \mid \nu s_2, \widetilde{\rho}_l.\prod_{j=1}^{l} x_{\beta_j}{:}O_j(U^{j-1}(S_2)) \\
= & \nu s_1, \widetilde{\rho}_k.(\psi_{R1} \mid y{:}U^{d-1}(S_1)) \mid \nu s_2, \widetilde{\rho}_l.\psi_{R2}
\end{array}$$

$\forall$ increasing sequences α_i, β_j s.t. $\{\alpha_i \mid 1 \leq i \leq k\} \cup \{\beta_j \mid 1 \leq j \leq l\} = [1, ..., k + l]$.

To conclude the proof we still need to prove that if backward privacy holds also the conditions hold. We show that $\neg\mathcal{P}_{SI} \vee \neg\mathcal{P}_{BI}$ implies

$$C[BrTag^b(c_1, c_2)] \not\approx C[BrTag^b(c_1) \mid Tag(c_2)]$$

There are two cases:

1. If $\mathcal{P}_{SI}$ does not hold, see proof of Theorem 5.3.7.

2. If $\mathcal{P}_{SI}$ holds but $\mathcal{P}_{BI}$ does not, there exists n such that

$$\prod_{i=1}^{n-2} x_i : O_i(U^{i-1}(S_0)) \mid y : U^{d-1}(S_0) \not\sim_{s,\widetilde{\rho}_m} \prod_{i=n}^{m} x_i : O_i(U^{i-1}(S_0)).$$

   If $\mathcal{P}_{SI}$ holds, the attacker can still break backward privacy by querying $n-2$ times the first interface, breaking it after $d-1$ protocol executions, and querying the second interface $m - n$ times. The frames that the attacker obtains in the case of independent and linked interfaces are not statically equivalent by assumption, therefore we can conclude that backward privacy is violated.

Thus, $\mathcal{P}_{SI}$ and $\mathcal{P}_{BI}$ are necessary and sufficient conditions for backward privacy.

□

B

PROVERIF CODE

To run ProVerif:

```
analyzer -in pi <file>
```

$\mathcal{P}_{SI}$ for OSK can be verified for infinite number of sessions:

```
analyzer -in pi model-osk-p1-infinite
```

For $\mathcal{P}_{FI}$ OSK a model can be generated for any fixed number of sessions:

```
perl generate-osk-p3.pl 10 > model-osk-p3-10
analyzer -in pi model-osk-p3-10
```

Same for the basic hash protocol and $\mathcal{P}_{OI}$:

```
perl generate-basichash-p2.pl 10 > model-basichash-p2-10
analyzer -in pi model-basichash-p2-10
```

B.1 $\mathcal{P}_{SI}$ for OSK protocol

```
(*
   This model checks P_SI for an infinite number of
   sessions

   The bi-processes starts with x=s on one side and
   x=h(s) on the other (x is the state, communicated
   on channel a).
   At each step we ouput g(x), and set x = h(x).
```

```
   Proverif proves that the processes are
   observationally equivalent hence the frames at
   each step are statically equivalent. The
   equivalences we get at each step:
     step 1: g(s) = g(h(s))
     step 2: g(h(s)) = g(h^2(s))
     step 3: g(h^2(s)) = g(h^3(s))
     ...
     step n: g(h^n(s)) = g(h^n+1(s))

   This holds for all n, so by transitivity we get
   property P_SI:
     g(h^n(s)) = g(h^m(s)) for all n,m
*)

  fun h/1.
  fun g/1.

  free c.

  process
    new a;
    new s;
    (
      !
      in(a, x);
      out(c, g(x));
      out(a, h(x))
    ) | (
      out(a, choice[s, h(s)])
    )

(*
   This is the same but using "phase" so that there
   is a single output
     g(h^n(s)) = g(h^n+1(s))
   for some deterministically chosen n.

  process
    new a;
    new s;
```

```
    (
      !
      in(a, x);
      out(a, h(x))
    ) | (
      in(a, x);
      phase 1;
      out(c, g(x))
    ) | (
      out(a, choice[s, h(s)])
    )
*)
```

B.2 $\mathcal{P}_{FI}$ for OSK protocol

B.2.1 Generate code

```
#!/usr/bin/perl -w
use strict;

# Generates a ProVerif model for checking P_FI for
# OSK for N sessions.

my $n = $ARGV[0]
or die "\nusage: generate-osk-p3.pl
  <number of sessions> [--p2]\n\n";
my $p2 = $ARGV[1] && $ARGV[1] eq '--p2';

# static part
print q{
param traceDisplay = none.
param verboseExplainClauses = false.
param explainDerivation = false.
param reconstructTrace = false.
param traceBacktracking = false.

fun h/1.
fun g/1.

(* sanity check: uncommenting the part below adds an
   invert function for g hence the property should no
```

```
   longer work *)

(*
  fun gi/1.
  equation gi(g(x)) = x.
*)

process
  new s;
  new s';
};

# print output common for both processes

my $i;
for($i = 0; $i < $n; $i++) {
  my $s = "g(" . ("h(" x $i) . "s" . (")" x $i) . ")";
  print "out(c, $s);\n";
}

# print "choice" part

my $s1 = ("h(" x $i) . "s"  . (")" x $i);
my $s2 = ("h(" x $i) . "s'" . (")" x $i);

if($p2) {
  $s1 = "g($s1)";
  $s2 = "g($s2)";
}

print "out(c, choice[$s1, $s2]);\n";
print "0\n";
```

B.2.2 Model for the OSK protocol

```
param traceDisplay = none.
param verboseExplainClauses = false.
param explainDerivation = false.
param reconstructTrace = false.
param traceBacktracking = false.
```

```
fun h/1.
fun g/1.

(* sanity check: including the part below adds an
   invert function for g hence the property should
   no longer work *)

fun gi/1.
equation gi(g(x)) = x.

process
  new s;
  new s';
  out(c, g(s));
  out(c, g(h(s)));
  out(c, g(h(h(s))));
  out(c, g(h(h(h(s)))));
  out(c, g(h(h(h(h(s))))));
  out(c, g(h(h(h(h(h(s)))))));
  out(c, g(h(h(h(h(h(h(s))))))));
  out(c, g(h(h(h(h(h(h(h(s)))))))));
  out(c, g(h(h(h(h(h(h(h(h(s))))))))));
  out(c, g(h(h(h(h(h(h(h(h(h(s)))))))))));
  out(c, choice[h(h(h(h(h(h(h(h(h(h(s))))))))))),
    h(h(h(h(h(h(h(h(h(h(s'))))))))))]);
  0
```

B.3 $\mathcal{P}_{FI}$ for basic hash protocol

B.3.1 Generate code

```
#!/usr/bin/perl -w
use strict;

# Generates a proverif model for checking P_OI
# for the basic hash protocol for N sessions.

my $n = $ARGV[0]
  or die "\nusage: generate-basichash-p2.pl
    <number of sessions>\n\n";
```

```
# static part
print q{
param traceDisplay = none.
param verboseExplainClauses = false.
param explainDerivation = false.
param reconstructTrace = false.
param traceBacktracking = false.

fun h/1.

(* sanity check: uncommenting the part below adds an
   invert function for h hence the property should no
   longer work *)

(*
  fun hi/1.
  equation hi(h(x)) = x.
*)

process
  new s;
  new s';
};

# print output common for both processes
#
my $i;
for($i = 0; $i < $n; $i++) {
  print qq {
    new r$i;
    out(c, (r$i,h((s, r$i))));
  };
}

# print "choice" part

print qq{
  new r$i;
};
my $s1 = "(r$i, h((s , r$i)))";
my $s2 = "(r$i, h((s', r$i)))";
```

```
print qq {
  out(c, choice[$s1, $s2]);
};
print "0\n";
```

B.3.2 Model for the basic-hash protocol

```
param traceDisplay = none.
param verboseExplainClauses = false.
param explainDerivation = false.
param reconstructTrace = false.
param traceBacktracking = false.

fun h/1.

(* sanity check: uncommenting the part below adds an
   invert function for h hence the property should no
   longer work *)

(*
  fun hi/1.
  equation hi(h(x)) = x.
*)

process
  new s;
  new s';

  new r0;
  out(c, (r0,h((s, r0))));

  new r1;
  out(c, (r1,h((s, r1))));

  new r2;
  out(c, (r2,h((s, r2))));

  new r3;
  out(c, (r3,h((s, r3))));
```

```
new r4;
out(c, (r4,h((s, r4))));

new r5;
out(c, (r5,h((s, r5))));

new r6;
out(c, (r6,h((s, r6))));

new r7;
out(c, (r7,h((s, r7))));

new r8;
out(c, (r8,h((s, r8))));

new r9;
out(c, (r9,h((s, r9))));

new r10;

out(c, choice[(r10, h((s , r10))), (r10, h((s', r10)))]);
0
```

BIBLIOGRAPHY

[1] ABADI, M. AND FOURNET, C. (2001). Mobile Values, New Names, and Secure Communication. In *Proc. of POPL*. ACM Press. pp. 104–115.

[2] ARAPINIS, M., CHOTHIA, T., RITTER, E. AND RYAN, M. (2009). Untraceability in the applied pi-calculus. In *Proc. of ICITST*. IEEE. pp. 1–6.

[3] ARAPINIS, M., CHOTHIA, T., RITTER, E. AND RYAN, M. (2010). Analysing Unlinkability and Anonymity Using the Applied Pi Calculus. In *Proc. of CSF*. IEEE Computer Society Press. pp. 107–121.

[4] AVOINE, G. (2005). Adversary Model for Radio Frequency Identification. Technical Report LASEC-REPORT-2005-001. Swiss Federal Institute of Technology Lausanne, Switzerland.

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

[6] BACKES, M., MAFFEI, M. AND UNRUH, D. (2008). Zero-Knowledge in the Applied Pi-calculus and Automated Verification of the Direct Anonymous Attestation Protocol. In *SP '08: Proc. of the 2008 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, Washington, DC, USA. pp. 202–215.

[7] BERMAN, R., FIAT, A. AND TA-SHMA, A. (2004). Provable Unlinkability Against Traffic Analysis. In *Proc. of Financial Cryptography*. LNCS. Springer. pp. 266–280.

[8] BLANCHET, B. (2001). An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *CSFW*. IEEE Computer Society Press. pp. 82–96.

[9] BLANCHET, B. (2004). Automatic Proof of Strong Secrecy for Security Protocols. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society Press. pp. 86–.

[10] BRUSÓ, M., CHATZIKOKOLAKIS, K. AND DEN HARTOG, J. Formalising Privacy Properties for RFID Systems. Submitted.

[11] BRUSÓ, M., CHATZIKOKOLAKIS, K. AND DEN HARTOG, J. (2010). Formal Verification of Privacy for RFID Systems. In *Proc. of CSF*. IEEE Computer Society Press. pp. 75–88.

[12] BRUSÓ, M., CHATZIKOKOLAKIS, K., ETALLE, S. AND DEN HARTOG, J. Dissecting Unlinkability. Submitted.

[13] BRUSÓ, M., CHATZIKOKOLAKIS, K., ETALLE, S. AND DEN HARTOG, J. (2013). Linking Unlinkability. In *Proc. of Trustworthy Global Computing*. Eds. C. Palamidessi and M. D. Ryan. LNCS. Springer.

[14] BURMESTER, M., LE, T. V. AND DE MEDEIROS, B. (2006). Provably Secure Ubiquitous Systems: Universally Composable RFID Authentication Protocols. In *Proc. of Securecomm*. IEEE. pp. 1–9.

[15] CAMENISCH, J. AND GROÃ§, T. (2008). Efficient Attributes for Anonymous Credentials. In *ACM Conference on Computer and Communications Security*. Eds. P. Ning, P. F. Syverson, and S. Jha. ACM Press. pp. 345–356.

[16] CAMENISCH, J. AND LYSYANSKAYA, A. (2001). An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *Proc. of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*. Ed. B. Pfitzmann. EUROCRYPT '01. Springer, London, UK, UK. pp. 93–118.

[17] CHADHA, R., DELAUNE, S. AND KREMER, S. (2009). Epistemic Logic for the Applied Pi Calculus. In *Proc. of IFIP*. Vol. 5522 of *LNCS*. Springer, Lisbon, Portugal. pp. 182–197.

[18] CHATMON, C., VAN LE, T. AND BURMESTER, M. (2006). Secure Anonymous RFID Authentication Protocols. Technical Report TR-060112. Florida State University Tallahassee, Florida, USA.

[19] CHAUM, D. (1985). Showing Credentials Without Identification. Signatures Transferred Between Unconditionally Unlinkable Pseudonyms. In *Proc. of EUROCRYPT'85*. Ed. F. Pichler. Vol. 219 of *LNCS*. Springer. pp. 241–244.

[20] CHOTHIA, T. AND SMIRNOV, V. (2010). A Traceability Attack against e-Passports. In *Financial Cryptography*. Ed. R. Sion. Vol. 6052 of *LNCS*. Springer. pp. 20–34.

[21] DECHESNE, F., MOUSAVI, M. R. AND ORZAN, S. (2007). Operational and Epistemic Approaches to Protocol Analysis: Bridging the Gap. In *Proc. of LPAR*. Vol. 4790 of *LNCS*. Springer. pp. 226–241.

[22] DIMITRIOU, T. (2008). RFID-DOT: RFID Delegation and Ownership Transfer Made Simple. In *Proc. of 4th International Conference on Security and Privacy in Communication Networks*. ACM Press. pp. 1–8.

[23] FRANZ, M., MEYER, B. AND PASHALIDIS, A. (2007). Attacking Unlinkability: The Importance of Context. In *Proc. of Privacy Enhancing Technologies*. Vol. 4776 of *LNCS*. Springer. pp. 1–16.

[24] GARCIA, F. D. AND VAN ROSSUM, P. (2010). Modeling Privacy for Off-Line RFID Systems. In *Proc. of CARDIS*. Vol. 6035 of *LNCS*. Springer. pp. 194–208.

[25] HALPERN, J. Y. AND O'NEILL, K. R. (2003). Anonymity and Information Hiding in Multiagent Systems. In *Proc. of CSFW*. IEEE Computer Society Press. pp. 75–88.

[26] HUGHES, D. AND SHMATIKOV, V. (2004). Information Hiding, Anonymity and Privacy: A Modular Approach. Vol. 12. IOS Press. pp. 3–36.

[27] JUELS, A. AND WEIS, S. A. (2007). Defining Strong Privacy for RFID. In *Proc. of PerCom Workshops*. IEEE Computer Society Press. pp. 342–347.

[28] LI, X., ZHANG, Y. AND DENG, Y. (2009). Verifying Anonymous Credential Systems in Applied Pi Calculus. In *CANS '09: Proc. of the 8th International Conference on Cryptology and Network Security*. Springer, Berlin, Heidelberg. pp. 209–225.

[29] LIM, C. H. AND KWON, T. (2006). Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer. In *Conference on Information and Communications Security – ICICS'06*. LNCS. Springer, Raleigh, North Carolina, USA.

[30] MEYER, J.-J. C. AND HOEK, W. V. D. (2004). *Epistemic Logic for AI and Computer Science.* Cambridge University Press, New York, NY, USA.

[31] MILNER, R., PARROW, J. AND WALKER, D. (1989). A Calculus of Mobile Processes, parts I and II. Vol. 100. pp. 1–77.

[32] NOHL, K. AND EVANS, D. (2009). Privacy Through Noise: a Design Space For Private Identification. In *Annual Computer Security Applications Conference (ACSAC 2009)*. IEEE Computer Society Press. pp. 518–527.

[33] OHKUBO, M., SUZUKI, K. AND KINOSHITA, S. (2003). Cryptographic Approach to "Privacy-Friendly" Tags. In *Proc. of RFID Privacy Workshop*.

[34] OUAFI, K. AND PHAN, R. C.-W. (2008). Privacy of Recent RFID Authentication Protocols. In *Proc. of ISPEC*. Vol. 4991 of *LNCS*. Springer. pp. 263–277.

[35] PFITZMANN, A. AND KOHNTOPP, M. (2001). Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology. Springer.

[36] STEINBRECHER, S. AND KOPSELL, S. (2003). Modelling Unlinkability. In *Proc. of Privacy Enhancing Technologies*. Vol. 2760 of *LNCS*. Springer. pp. 32–47.

[37] SYVERSON, P. F. AND STUBBLEBINE, S. G. (1999). Group Principals and the Formalization of Anonymity. In *Proc. of World Congress on Formal Methods*. Vol. 1708 of *LNCS*. Springer. pp. 814–833.

[38] VAN DEURSEN, T., MAUW, S. AND RADOMIROVIC, S. (2008). Untraceability of RFID Protocols. In *Proc. of WISTP*. Vol. 5019 of *LNCS*. Springer. pp. 1–15.

[39] VAN DEURSEN, T. AND RADOMIROVIC, S. (2009). Algebraic Attacks on RFID Protocols. In *Proc. of WISTP*. Vol. 5746 of *LNCS*. Springer. pp. 38–51.

[40] VAUDENAY, S. (2007). On Privacy Models for RFID. In *Proc. of ASIACRYPT*. Vol. 4833 of *LNCS*. Springer. pp. 68–87.

[41] WEIS, S. A., SARMA, S. E., RIVEST, R. L. AND ENGELS, D. W. (2003). Security and Privacy Aspects of Low-Cost RFID Systems. In *Proc. of SPC*. Vol. 2802 of *LNCS*. Springer. pp. 201–212.

SUMMARY

Radio-frequency identification (RFID) systems are a wireless technology for automatic identification of objects or persons. This technology has raised a number of privacy concerns due to its mobile nature and to the limited capabilities of RFID tags.

The privacy issues in RFID systems differ from those affecting other systems. In identification systems an attack typically aims at disclosing private information stored in the user devices, while in RFID systems an attack aims at tracing the movements of a person (or an object) carrying a specific tag, as tags do not usually store personal information (or object description).

In this thesis we study the notion of unlinkability, a privacy property that holds in an RFID system when it is impossible to trace one of its tags. As the current literature comprises a number of different interpretations and definitions of this property and other related notions, we start by formally comparing these interpretations in a unifying framework. Then we develop a formal framework tailored to RFID systems where it is possible to describe and analyse RFID protocols.

More specifically, in Chapter 1 we briefly introduce the concept of unlinkability in the context of RFID systems. Chapter 2 provides the reader with the main ingredients that we use to define the unifying framework presented in Chapter 3 and the RFID model in Chapter 4. We use our unifying framework to capture several definitions of unlinkability from the literature and to formalise other related privacy properties. The differences between all these privacy definitions are explored, formally proving their relationships and showing that, under a set of conditions, the properties coincide. The RFID model in Chapter 4 allows us to describe RFID protocols using the applied pi calculus and to define unlinkability and other privacy notions taking into account the particular features of RFID systems. In Chapter 5 we describe and study a class of protocols and analyse some protocols from the literature that fall in this class. Finally, we present our conclusion and future work in Chapter 6.

CURRICULUM VITAE

Mayla Brusó was born on 19/11/1984 in Venice, Italy.

She studied Computer Science at Universitá Ca' Foscari in Venice, Italy, where she graduated cum laude in 2008. The main subject of her research was formal analysis of security protocols. Her master thesis "'Non-Repudiation Analysis With LySa"' was supervised by Prof. Dr. Agostino Cortesi. From 2009 she started a Ph.D. project at TU/e in Eindhoven, The Netherlands, of which the results are presented in this dissertation.