Analysing Unlinkability and Anonymity Using the Applied Pi Calculus

Myrto Arapinis, Tom Chothia, Eike Ritter and Mark Ryan

School of Computer Science, University of Birmingham, UK
{m.d.arapinis, t.chothia, e.ritter, m.d.ryan}@cs.bham.ac.uk

Abstract—**An attacker that can identify messages as coming from the same source, can use this information to build up a picture of targets' behaviour, and so, threaten their privacy. In response to this danger, unlinkable protocols aim to make it impossible for a third party to identify two runs of a protocol as coming from the same device. We present a framework for analysing unlinkability and anonymity in the applied pi calculus. We show that unlinkability and anonymity are complementary properties; one does not imply the other. Using our framework we show that the French RFID e-passport preserves anonymity but it is linkable therefore anyone carrying a French e-passport can be physically traced.**

I. Introduction

The proliferation of portable computing devices has lead to a range of new computer security problems. Mobile phones, Bluetooth devices and RFID tags have all been shown to leak private information, and such security concerns are regularly reported by the media [9], [6], [17]. Protocols that keep the identity of their users secure may still allow an attacker to identify particular sessions as having involved the same principal. Such *linkability attacks* may, for instance, make it possible for a third party to trace the movements of someone carrying an RFID tag. Linkability attacks are not restricted to mobile devices; for example, they can also occur when using web-based services. When AOL published an anonymised list of search queries, the ability to link the different queries lead to serious breaches of privacy [6].

In this paper we give a framework for analysing unlinkability and anonymity in the applied pi-calculus. Using the applied pi-calculus [2] has the advantage of making our definitions precise, and lets us build on a large body of work on checking other security properties. Another advantage of the applied pi-calculus is that, in many cases, we can check protocols automatically using the ProVerif tool [7].

We give a "strong" and a "weak" definition of unlinkability. The strong definition is very much in the style of the applied pi-calculus: we say that a protocol is strongly unlinkable if an instance of the protocol made up of an arbitrary number of repeating processes is bisimilar to an instance in which no process runs more than once. While a protocol that has this property is unlinkable, the failure of this property does not guarantee a practical attack that would allow an observer to conclude that two particular runs come from the same process. Furthermore, bisimulation may fail due to the internal state of the processes, and these differences in the internal state may not be observable by an outside attacker. We define weak unlinkability based on the traces of a system. We consider the case where an attacker has observed a system and decides that two particular messages might be from different sessions being performed by the same agent. The system is weakly unlinkable if, for all such cases, there exists another trace of the system, which looks the same to the attacker, and in this other trace the two messages came from different agents. A failure of weak unlinkability directly implies an attack.

We show that our definition of strong unlinkability implies the weaker version. As strong unlinkability can sometimes be checked automatically using the ProVerif tool, this means that when checking a protocol it is useful to check the strong definition first. If it fails, one may go on to use the weaker version to look for a practical attack. We also show that unlinkability does not imply anonymity, contradictory to what has been suggested by other authors [13]. An example of a device that would be unlinkable but not anonymous is an RFID tag with an identity-revealing kill function. Suppose the protocol is such that the past sessions are not linkable with the identity, and now the kill function is executed. It permanently disables the tag but also reveals its identity. Anonymity has been broken, but unlinkability of past sessions remains intact.

While looking for case studies to illustrate our work, we found a new linkability attack against the French version of the e-Passport. The Basic Access Control (BAC) protocol on e-Passports is designed to make the passports unlinkable [21]. We show that in the case of the French e-Passport, a replayed message can be used to identify a passport. This attack makes it possible to detect when a particular passport comes into range of a reader, which could for instance, be placed by a doorway, in order to monitor when a target enters or leaves a particular building. We also show that other nations, like UK, Germany, Ireland, and Russia have avoided this linkability attack in their implementation of the BAC protocol.

In the next section we briefly describe the applied pi-calculus. Our main definitions of strong and weak unlinkability are defined in Section III. The following two sections give our case study. Finally we conclude and discuss further work in Section V.

Previous and Related Work: We have previously announced the existence of the attack on the French passport [11], [17], this paper is the first to present a formal analysis of the attack we found. We suggested using the applied pi calculus to model untraceability in a previous paper [3]. The exact forms of our definitions and all of the theorems presented in this paper are new.

While security, authenticity and anonymity have been widely studied, unlinkability has received less attention. A number of papers discuss the privacy problems raised by RFID technologies (see for example [16], [18], [24]) but very few precisely define what they mean by unlinkable[1]. Avoine *et al.* in [5] were the first to give a formal definition of unlinkability. Some other attempts to formalise unlinkability then followed [4], [8], [19], [23]. All this work is carried out in the computational model, which is poorly supported by automatic tools. The advantage of our work is that it is carried out in the symbolic setting, which (as already mentioned) is supported by the ProVerif tool. As these two settings are very different, it is difficult to compare our work directly.

In the symbolic world, Deursen *et al.* propose a formal definition of untraceability in a particular trace model [13] and in an algebraic model [14]. Their definition is similar to our definition of weak unlinkability, however our model is more general and allows us to also consider stronger definitions of unlinkability and anonymity, while the equational theory of the applied pi calculus lets us add arbitrary cryptographic primitives.

Symbolic work on anonymity is more common than work on unlinkability. For instance, Schneider and Sidiropoulos use CSP to analyse anonymity [22] and Garcia et al. [15] develop their own framework for proving anonymity based on epistemic logic. In previous work, we used a definition of anonymity in the pi-calculus to find a flaw in an anonymous file-sharing system [10], the definition of anonymity that we present in this paper is much more general. We have also looked at the anonymity of voting systems in the applied pi calculus [12].

II. The applied pi calculus and p-party protocols

The applied pi calculus [2] is a language for modelling distributed systems and their interactions. It extends the pi calculus with an equational theory, which is particularly useful for modelling cryptographic protocols. The following subsections describe the syntax and semantics of the calculus, equality relations for processes and the way in which we will describe protocols.

A. Syntax

The calculus assumes an infinite set of names $\mathcal{N} = \{a, b, c, \ldots\}$, an infinite set of variables $\mathcal{V} = \{x, y, z, \ldots\}$ and a finite signature Σ, that is, a finite set of function symbols each with an associated arity. We use meta-variables u, v, w to range over both names and variables. Terms $M, N, T, \ldots$ are built by applying function symbols to names, variables and other terms. Tuples $M_1, \ldots, M_l$ are occasionally abbreviated $\tilde{M}$. We write $\{^{M_1}/_{u_1}, \ldots, ^{M_l}/_{u_l}\}$ for substitutions that replace $u_1, \ldots, u_l$ with $M_1, \ldots, M_l$. The applied pi calculus relies on a simple type system. Terms can be of sort $\mathsf{Channel}$ for channel names or Base for the payload sent out on these channels. Function symbols can only be applied to, and return, terms of sort Base. A term is ground when it does not contain variables. The grammar for processes is shown in Figure 1 where u is either a name or variable of channel sort.

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

Fig. 1. Applied pi calculus grammar

Plain processes are standard. Extended processes introduce *active substitutions* which generalise the classical let construct: the process $\nu\, x.(\{^{M}/_{x}\} \mid P)$ corresponds exactly to the process let $x = M$ in P. As usual names and variables have scopes which are delimited by restrictions and by inputs. All substitutions are assumed to be cycle-free.

The sets of free and bound names, respectively variables, in process A are denoted by $\text{fn}(A)$, $\text{bn}(A)$, $\text{fv}(A)$, $\text{bv}(A)$. We also write $\text{fn}(M)$, $\text{fv}(M)$ for the names, respectively variables, in term M. An extended process A is *closed* if it has no free variables. A *context* $C[\_]$ is an extended process with a hole. We obtain $C[A]$ as the result of filling $C[\_]$'s hole with A. An *evaluation context* is a context whose hole is not under a replication, a conditional, an input, or an output.

The signature Σ is equipped with an equational theory E, that is a finite set of equations of the form $M = N$. We define $=_E$ as the smallest equivalence relation on terms, that contains E and is closed under application of function symbols, substitution of terms for variables and bijective renaming of names.

B. Semantics

We now define the operational semantics of the applied pi calculus by the means of two relations: structural equivalence and internal reductions. *Structural equivalence* ($\equiv$) is the smallest equivalence relation closed under α-conversion of both bound names and variables and application of evaluation contexts such that:

$$\begin{array}{rcl@{\qquad}rcl}
A \mid 0 & \equiv & A & \nu n.0 & \equiv & 0 \\
A \mid (B \mid C) & \equiv & (A \mid B) \mid C & \nu u.\nu w.A & \equiv & \nu w.\nu u.A \\
A \mid B & \equiv & B \mid A & A \mid \nu u.B & \equiv & \nu u.(A \mid B) \\
!P & \equiv & P \mid !P & \multicolumn{3}{c}{\text{if } u \notin \text{fn}(A) \cup \text{fv}(A)} \\
 & & & & & \\
\nu x.\{^{M}/_{x}\} & \equiv & 0 & \{^{M}/_{x}\} & \equiv & \{^{N}/_{x}\} \\
\{^{M}/_{x}\} \mid A & \equiv & \{^{M}/_{x}\} \mid A\{^{M}/_{x}\} & \multicolumn{3}{c}{\text{if } M =_E N}
\end{array}$$

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

[1] The term "untraceability" is often used in other papers. While the terms are sometimes interchangeable, we use "unlinkability" because we do not want to imply that the targets are necessarily being physically tracked.

$$\begin{array}{ll}\textsc{Comm} & \overline{c}\langle x\rangle.P \mid c(x).Q \rightarrow P \mid Q \\ \textsc{Then} & \text{if } N = N \text{ then } P \text{ else } Q \rightarrow P \\ \textsc{Else} & \text{if } L = M \text{ then } P \text{ else } Q \rightarrow Q \\ & \quad \text{for ground terms } L, M \text{ where } L \neq_E M\end{array}$$

We write $\Rightarrow$ for an arbitrary (possibly zero) number of internal reductions.

Labelled reduction ($\xrightarrow{\alpha}$) extends the internal reduction and enables the environment to interact with the processes. The label α is either an input, or the output of a channel name or a variable of base type.

$$a(x).P \xrightarrow{a(M)} P\{^{M}/_{x}\} \qquad \overline{a}\langle u\rangle.P \xrightarrow{\overline{a}\langle u\rangle} P$$

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

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

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

$$\frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad A' \equiv B'}{A \xrightarrow{\alpha} A'}$$

We write $\overset{\alpha}{\Rightarrow}$ for an arbitrary number of internal reductions followed by a labelled reduction and then an arbitrary number of internal reductions, i.e., $\overset{\alpha}{\Rightarrow}$ equals $\Rightarrow\xrightarrow{\alpha}\Rightarrow$.

Example 1: Assume the empty signature, and consider the process

$$P \;\triangleq\; !U \qquad\qquad U \;\triangleq\; \nu id.\ !(\nu s.\ \overline{c}\langle hello\rangle)$$

This process P models a system with an unbounded number of agents ($!U$). Each agent has a distinct identity as indicated by νid (although the ID is not used by the agent). Each agent executes an unbounded number of runs of the protocol (as indicated by the "!" in the U process) and each session simply involves outputting the message "hello" on the public channel c. Every session is associated with a distinct session identifier (νs), which again is not used directly by the agent. The use of identifiers for particular agents help us keep track of who performs which actions, as we will see in Section III.

A possible trace of the process P is:

$$\begin{array}{rl} P \;\equiv & \nu id_1.\ \nu id_2.\ \nu s_1.\ \nu s_2. \\ & \left(\begin{array}{l} !U \mid !(\nu s.\ \overline{c}\langle hello\rangle) \mid !(\nu s.\ \overline{c}\langle hello\rangle) \mid \\ \overline{c}\langle hello\rangle \mid \overline{c}\langle hello\rangle \end{array}\right) \\ \xrightarrow{\nu x.\ \overline{c}\langle x\rangle} & \nu id_1.\ \nu id_2.\ \nu s_1.\ \nu s_2. \\ & \left(\begin{array}{l} !U \mid !(\nu s.\ \overline{c}\langle hello\rangle) \mid !(\nu s.\ \overline{c}\langle hello\rangle) \mid \\ \overline{c}\langle hello\rangle \mid \{hello/x\} \end{array}\right) \\ \xrightarrow{\nu y.\ \overline{c}\langle y\rangle} & \nu id_1.\ \nu id_2.\ \nu s_1.\ \nu s_2. \\ & (\ !U \mid !(\nu s.\ \overline{c}\langle hello\rangle) \mid !(\nu s.\ \overline{c}\langle hello\rangle) \mid \{hello/y\}\ ) \end{array}$$

In this trace, there are two agents with the identities id_1 and id_2 and two sessions of the protocol are executed. We cannot tell if both sessions where executed by the same agent, or if each agent executed a single run. This is because, in the first step, the $\overline{c}\langle hello\rangle$ processes could have been unwound from either agent.

C. Equivalence Relations for Processes

In this subsection, we define our notions of equivalence for processes. First, we say when active substitutions are the same, then we go on to define two complementary notions of equivalence: labelled bisimulation and trace equivalence.

A *frame*, denoted φ or ψ, is an extended process built from 0 and active substitutions $\{^{M}/_{x}\}$, which are composed by parallel composition and restriction. The frame $\varphi(A)$ represents the static knowledge output by a process to its environment. The *domain* dom(φ) of a frame φ is the set of variables that φ exports; that is, the set of variables x for which φ contains an active substitution $\{^{M}/_{x}\}$ such that x is not under restriction. Every extended process A can be mapped to a frame $\varphi(A)$ by replacing every plain process in A with 0.

Two frames are considered different when there exists a pair of variables that are equal under the substitutions of one frame but not equal under the substitutions of the other. When it is impossible to tell two frames apart in this way, we say that they are statically equivalent:

Definition 1 (Static equivalence): Two closed frames $\phi \equiv \nu\,\tilde{m}.\sigma$ and $\theta \equiv \nu\,\tilde{n}.\tau$ are statically equivalent, denoted $\varphi \approx_s \theta$, if dom(φ) $=$ dom(θ) and for all terms M, N such that $(\tilde{m} \cup \tilde{n}) \cap (\text{fn}(M) \cup \text{fn}(N)) = \emptyset$, we have $M\sigma =_E N\sigma$ holds if and only if $M\tau =_E N\tau$ holds. Two closed extended processes A, B are statically equivalent, written $A \approx_s B$, if $\varphi(A) \approx_s \varphi(B)$.

We now present two notions of equivalence for the applied pi calculus. The first, trace equivalence, says that two processes are the same when they process the same sequences of inputs and outputs:

Definition 2 (Trace equivalence): Let

$$tr_A \;=\; A_0 \overset{\alpha_1}{\Rightarrow} A_1 \overset{\alpha_2}{\Rightarrow} \ldots \overset{\alpha_{n-1}}{\Rightarrow} A_{n-1} \overset{\alpha_n}{\Rightarrow} A_n$$

and

$$tr_B \;=\; B_0 \overset{\alpha_1}{\Rightarrow} B_1 \overset{\alpha_2}{\Rightarrow} \ldots \overset{\alpha_{n-1}}{\Rightarrow} B_{n-1} \overset{\alpha_n}{\Rightarrow} B_n$$

The traces tr_A and tr_B are *equivalent* denoted $tr_A \sim tr_B$ if $A_i \approx_s B_i$ for all i.

Two processes are trace equivalent if for every trace from each process there is an equivalent trace from the other.

Our second notion of equivalence, labelled bisimilarity, requires that the states of the reductions can be matched, as well as their actions. It is a stronger relation than trace equivalence; labelled bisimulation implies trace equivalence. However, if labelled bisimulation fails due to miss-matched states, then it still may be the case that an outside observer cannot tell the processes apart, as there may be no way to detect the difference in state from outside the process. Labelled bisimulation is however more efficient to check, in terms of time, because a failure can be detected sooner.

Definition 3 (Labelled bisimilarity): Labelled bisimilarity ($\approx_l$) is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A \mathrel{\mathcal{R}} B$ implies:

1) $A \approx_s B$;
2) if $A \to A'$ then $B \Rightarrow B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B';
3) if $A \xrightarrow{\alpha} A'$ and $\text{fv}(\alpha) \subseteq \text{dom}(A)$ and $\text{bn}(\alpha) \cap \text{fn}(B) = \emptyset$; then $B \overset{\alpha}{\Rightarrow} B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B'.

D. The Class of p-Party Protocols

In order to simplify our definitions we define standard forms of processes and protocols.

Definition 4 (Canonical form): A process A is said to be in canonical form if *(i)* names and variables never appear both bound and free, *(ii)* each name and variable is bounded at most once in A:

(i) $\text{bn}(A) \cap \text{fn}(A) = \emptyset$ and $\text{bv}(A) \cap \text{fv}(A) = \emptyset$
(ii) $\forall \mathcal{C}[\_], \mathcal{D}[\_],\ \forall B, C,\ \forall u$
$A = \mathcal{C}[\nu u.B] \ \wedge\ A = \mathcal{D}[\nu u.C] \ \Rightarrow\ \mathcal{C}[\_] = \mathcal{D}[\_] \ \wedge\ B = C$

Note that by α-conversion we can always transform a process in a structurally equivalent one which is in canonical form.

Definition 5 (Well-formed Protocol): A p-party protocol is said to be *well-formed* if it is a closed plain process P of the form:

$$\begin{array}{rcl} P & = & \nu\tilde{n}.\ (!R_1 \mid \ldots \mid !R_p) \\ \forall i \in \{1, \ldots, p\}\ R_i & = & \nu id.\ \nu\tilde{m}.\ init_i.\ !(\nu s.\ main_i) \end{array}$$

where *(i)* all channels are ground, *(ii)* private channels are never sent on public channels, *(iii)* P is in canonical form, *(iv)* $init_i$ and $main_i$ are any two processes (possibly empty) such that P is a closed plain process, and *(v)* because we consider a Dolev-Yao attacker, *i.e.* controlling the network, public input channels and public output channels of P are disjoint.

For all i in $\{1, \ldots, p\}$, R_i models the i^{th} role of the protocol and $main_i$ models a session of role R_i. A user is an instance of a role, and we consider an unbounded number of users (replication in front of R_is). Each user is dynamically associated to a distinct identity (νid). A user can execute an unbounded number of sessions of the role he takes (replication in front of $main_i$s). Each such session will dynamically be associated to a distinct session identifier (νs).

It is important to note that this representation of protocols does not limit the kinds of processes we may consider. In the above definition either the $main$ or the $init$ process may be null and the processes do not have to explicitly make use of their identities. The roles may include a number of sub-roles, so each user may play more than one part in the protocol. This means that any protocol can be written as a well-formed protocol.

Example 2: Consider an RFID system with two roles: readers (R) and tags (T). All readers share the same private key k corresponding to the public key $\mathsf{pbk}(k)$. The key-servers (K) publish the readers' public key $\mathsf{pbk}(k)$. Tags identify themselves to readers by sending their identity id encrypted with the readers' public key $\mathsf{pbk}(k)$. For this example we consider the asymmetric encryption algorithm to be deterministic. This protocol can be modelled in the Alice and Bob notation by

$$T\ (id, \mathsf{pbk}(k)) \quad \xrightarrow{\{id\}_{\mathsf{pbk}(k)}} \quad R\ (k, \mathsf{pbk}(k))$$

and by the following closed plain process over $\Sigma = \{\mathsf{aenc}, \mathsf{adec}, \mathsf{pbk}\}$ with the equational theory $E = \{\mathsf{adec}(\mathsf{aenc}(x, y)) = pdk(x)\}$

$$\begin{array}{rcl} P & \triangleq & \nu k.\ (!R \mid !T \mid !K) \\ R & \triangleq & \nu id.\ !(\nu s.\ in(x)) \\ T & \triangleq & \nu id.\ !(\nu s.\ \overline{out_T}\langle \mathsf{aenc}(id, \mathsf{pbk}(k))\rangle) \\ K & \triangleq & \nu id.\ \overline{out_K}\langle \mathsf{pbk}(k)\rangle \end{array}$$

Example 3: We consider three traces of P of Figure 2. It is easy to see that $A_2 \not\approx_s A_4$. Indeed, if we consider the terms y_1 and y_2, then

$$y_1\phi(A_2) = \mathsf{aenc}(id_1, \mathsf{pbk}(k)) =_E \mathsf{aenc}(id_1, \mathsf{pbk}(k)) = y_2\phi(A_2)$$

but

$$y_1\phi(A_4) = \mathsf{aenc}(id_1, \mathsf{pbk}(k)) \neq_E \mathsf{aenc}(id_2, \mathsf{pbk}(k)) = y_2\phi(A_4)$$

We can thus conclude that $tr_1 \not\sim tr_2$. On the other hand, $\phi(P) = \phi(A_5)$, $\phi(A_1) = \phi(A_6)$, and $\phi(A_2) = \phi(A_7)$ imply that $tr_1 \sim tr_3$.

III. Formalising privacy

The formal definitions of unlinkability and anonymity require us to define two complementary operations, namely the *augment* and the *erase* operations. The need for having the augment operation comes from the fact that the applied pi calculus does not provide us with information on the initiator of a transition. To express unlinkability we need to distinguish transitions initiated by the same users and transitions initiated by different users. However, we cannot do so in the trace exhibited in Example 1, for instance, because this trace corresponds to both the case where *hello* is outputted by the same user and the case when it is outputted by two different users.

A. Augmented processes

The purpose of the augment operation is to augment/annotate protocols in such a way that the initiator of particular actions is made explicit in the traces of the augmented/annotated protocol. Annotations are message outputs $\overline{f}\langle M\rangle$ where f is a channel not used in the original process. In labelled transitions, output labels for annotations use particular annotation variables. These annotation variables are not allowed to appear in input labels $u(M)$, so the

$$
\begin{array}{rl}
tr_1 & = P\\
\xrightarrow{\nu y_1.\ \overline{out_T}\langle y_1\rangle} & A_1 = \nu id_{T,1}.\ \nu s_{T,1}.\ \nu k.\\
& \left(\begin{array}{l} !R \mid !T \mid !K \mid \\ !(\nu s.\ \overline{out_T}\langle \mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))\rangle) \mid \\ \{\mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))/y_1\} \end{array}\right)\\
\xrightarrow{\nu y_2.\ \overline{out_T}\langle y_2\rangle} & A_2 = \nu id_{T,1}.\ \nu s_{T,1}.\ \nu s_{T,2}.\ \nu k.\\
& \left(\begin{array}{l} !R \mid !T \mid !K \mid \\ !(\nu s.\ \overline{out_T}\langle \mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))\rangle) \mid \\ \{\mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))/y_1\} \mid \\ \{\mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))/y_2\} \end{array}\right)\\
tr_2 & = P\\
\xrightarrow{\nu y_1.\ \overline{out_T}\langle y_1\rangle} & A_3 = \nu id_{T,1}.\ \nu s_{T,1}.\ \nu k.\\
& \left(\begin{array}{l} !R \mid !T \mid !K \mid \\ !(\nu s.\ \overline{out_T}\langle \mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))\rangle) \mid \\ \{\mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))/y_1\} \end{array}\right)\\
\xrightarrow{\nu y_2.\ \overline{out_T}\langle y_2\rangle} & A_4 = \nu id_{T,1}.\ \nu id_{T,2} \nu s_{T,1}.\ \nu s_{T,2}.\ \nu k.\\
& \left(\begin{array}{l} !R \mid !T \mid !K \mid \\ !(\nu s.\ \overline{out_T}\langle \mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))\rangle) \mid \\ !(\nu s.\ \overline{out_T}\langle \mathsf{aenc}(id_{T,2},\mathsf{pbk}(k))\rangle) \mid \\ \{\mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))/y_1\} \mid \\ \{\mathsf{aenc}(id_{T,2},\mathsf{pbk}(k))/y_2\} \end{array}\right)\\
tr_3 & = P\\
\xrightarrow{\qquad\qquad} & A_5 = \nu id_{T,1}.\ \nu id_{R,1}.\ \nu s_{T,1}.\ \nu s_{R,1}.\ \nu k.\\
& \left(\begin{array}{l} !R \mid !T \mid !K \mid \\ !(\nu s.\ \overline{out_T}\langle \mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))\rangle) \mid \\ !(\nu s.\ in(x)) \end{array}\right)\\
\xrightarrow{\nu y_1.\ \overline{out_T}\langle y_1\rangle} & A_6 = \nu id_{T,1}.\ \nu id_{R,1} .\nu s_{T,1}.\ \nu s_{T,2}.\ \nu s_{R,1} .\nu k.\\
& \left(\begin{array}{l} !R \mid !T \mid !K \mid \\ !(\nu s.\ \overline{out_T}\langle \mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))\rangle) \mid \\ !(\nu s.\ in(x)) \mid \\ \{\mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))/y_1\} \end{array}\right)\\
\xrightarrow{\nu y_2.\ \overline{out_T}\langle y_2\rangle} & A_7 = \nu id_{T,1}.\ \nu id_{R,1} .\nu s_{T,1}.\ \nu s_{T,2}.\ \nu s_{T,3}.\ \nu s_{R,1}.\ \nu k.\\
& \left(\begin{array}{l} !R \mid !T \mid !K \mid \\ !(\nu s.(\overline{out_T}\langle \mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))\rangle) \mid \\ !(\nu s.\ in(x)) \mid \\ \{\mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))/y_1\} \mid \\ \{\mathsf{aenc}(id_{T,1},\mathsf{pbk}(k))/y_2\} \end{array}\right)
\end{array}
$$

Fig. 2. Three valid traces of P

adversary cannot learn them. (This condition is important, since annotations are added just for checking conditions on the initiator of some actions; an output $\overline{f}\langle M\rangle$ doesn't reveal M to the adversary). Hence, the execution of the process P after inserting annotations $\overline{f}\langle M\rangle$ is the execution of P without annotations, plus the recording of annotations using labels $\overline{f}\langle\ell\rangle$ and active substitutions $\{M/\ell\}$.

Definition 6 (Annotated traces): Let A be an annotated process using the particular channel f for annotations. The set $\mathcal{T}_A$ is the set of traces tr such that

$$tr \ = \ A \Rightarrow\xrightarrow{\alpha_1}\xrightarrow{\beta_1}\Rightarrow A_1 \ldots A_{n-1} \Rightarrow\xrightarrow{\alpha_n}\xrightarrow{\beta_n}\Rightarrow A_n$$

where $\forall i \in \{1, \ldots, n\}$

1) $\alpha_i = \nu\ell.\ \overline{f}\langle\ell\rangle$. for some variable ℓ;
2) $(\mathrm{fn}(\beta_i) \cup \mathrm{bn}(\beta_i)) \cap \{f\} = \emptyset$;
3) $(\mathrm{fv}(\beta_i) \cup \mathrm{bv}(\beta_i)) \cap V = \emptyset$ where V is the set of variables occurring in the αs, *i.e.* $V = \underset{j\in\{1,\ldots,n\}}{\cup} \mathrm{bv}(\alpha_j)$.

Intuitively, $\mathcal{T}_A$ restricts the non-silent actions of traces to alternate between an action on the channel f and an action on some other channel. It also requires that annotation variables do not occur in input actions, which corresponds to the fact that the intruder cannot access those variables.

Definition 7 (Augment operation): The augment operation annotates all actions on channels in the set $\mathcal{C}$ with a term L (not already used in the process), using the particular channel $f \notin Chan(A)$ as depicted in Figure 3. For an extended process A, the augmented process is $A' = aug(A, \mathcal{C}, f, L)$.

We note that we do not allow channel names to be passed, therefore a broadcast of the label L on the channel f will be added before every use of the channels in the set $\mathcal{C}$. We also note that the coupling between an action and the broadcast of the label for that action is quite loose. Only when we augment all public channels and restrict the non-silent actions of traces to alternate between an action on the channel f and an action on some other channel do we get a direct correspondence between an action and its label.

Assume we have a p-Party protocol A and we augment this process so that it broadcasts the label L on the channel f before every use of a public channel, giving us A'. Then for all annotated traces of A', as formalised in Definition 6, the erasure of that trace is equivalent to some trace of A. In the other direction, for every trace of A there exists an equivalent erased trace of A'. We note that this only holds because we restrict ourselves to annotated traces, as defined in Definition 6, and we do not allow processes to output and input on the same public name.

We illustrate the intuition behind the augment operation in this example:

Example 4: Returning to our running example (Example 2), we can use the augment operation to indicate the initiator of each action. Our purpose is to accompany the first transition of tr_1 with a message that informs us that the initiator of this transition is a tag with identity $id_{T,1}$ and executing session $s_{T,1}$. In the same way we want to accompany the second transition of this trace with a message indicating that the initiator of this transition is the same tag $id_{T,1}$ but executing session $s_{T,2}$ this time. To do so, we consider a new ternary function symbol from and a constant T with no equational theory associated with it, and we accompany each action initiated by a tag with identity id executing session s with the message $\mathsf{from}(\mathsf{T}, id, s)$. If we use the channel c_{fr} to augment the process T of our running example, we obtain the following process:

$$
\begin{aligned}
T' &= aug(T, \{in, out_T, out_K\}, c_{fr}, \mathsf{from}(\mathsf{T}, id, s))\\
&= \nu id.\ !(\nu s.\ \overline{c_{fr}}\langle \mathsf{from}(\mathsf{T}, id, s)\rangle.\ \overline{out_T}\langle \mathsf{aenc}(id, \mathsf{pbk}(k))\rangle)
\end{aligned}
$$

In doing so, we enforce that, before each transition initiated by a tag t, t outputs a message indicating its identity and its current session. By restricting the set of considered traces for

$$
\begin{array}{lcl}
aug(0,\mathcal{C},f,L) & \triangleq & 0 \\
aug(!P,\mathcal{C},f,L) & \triangleq & !aug(P,\mathcal{C},f,L) \\
aug(\mathsf{if}\ M=N\ \mathsf{then}\ P\ \mathsf{else}\ Q,\mathcal{C},f,L) & \triangleq & \mathsf{if}\ M=N\ \mathsf{then}\ \overline{f}\langle L\rangle.aug(P,\mathcal{C},f,L) \\
 & & \mathsf{else}\ \overline{f}\langle L\rangle.aug(Q,\mathcal{C},f,L) \\
aug(u(x).P,\mathcal{C},f,L) & \triangleq & \begin{cases} \overline{f}\langle L\rangle.u(x).aug(P,\mathcal{C},f,L) & \text{if } u\in\mathcal{C} \\ u(x).aug(P,\mathcal{C},f,L) & \text{otherwise} \end{cases} \\
aug(\overline{u}\langle M\rangle.P,\mathcal{C},f,L) & \triangleq & \begin{cases} \overline{f}\langle L\rangle.\overline{u}\langle M\rangle.aug(P,\mathcal{C},f,L) & \text{if } u\in\mathcal{C} \\ \overline{u}\langle M\rangle.aug(P,\mathcal{C},f,L) & \text{otherwise} \end{cases} \\
aug(A\mid B,\mathcal{C},f,L) & \triangleq & aug(A,\mathcal{C},f,L)\mid aug(B,\mathcal{C},f,L) \\
aug(\nu u.A,\mathcal{C},f,L) & \triangleq & \nu u.aug(A,\mathcal{C},f,L) \\
aug(\{M/x\},\mathcal{C},f,L) & \triangleq & \{M/x\}
\end{array}
$$

Fig. 3. The augment operation

the resulting process, we force these two outputs to happen consecutively.

Notation 1: For an annotated process A using channel f for the annotations, we define $A \overset{(id,s,R):\alpha}{\Longrightarrow} B$ to mean the trace $tr \in \mathcal{T}_A$ of the form

$$tr \;=\; A \Rightarrow C_1 \xrightarrow{\nu\ell.\overline{f}\langle\ell\rangle} C_2 \xrightarrow{\alpha} C_3 \Rightarrow B,$$

for some C_1, C_2, C_3 where $\phi(C_2) \equiv \nu\tilde{p}.\sigma$ and $\ell\sigma = \mathsf{from}(id,s,R)$, and $\alpha \neq \nu\ell.\ \overline{f}\langle\ell\rangle$ for any ℓ.

We now introduce the complementary operation *erase*, that in some cases is the inverse of the augment operation. Informally, given a set of channels $\mathcal{C}$ and a set of variables $\mathcal{V}$, applying the erase operation to a process A results in a process A', similar to A, but with its communications restricted to the ones not taking place on channels in $\mathcal{C}$, and its active substitutions restricted to the ones not in $\mathcal{V}$. In other words, this operation erases the actions occurring on channels of $\mathcal{C}$ as well as the active substitutions defining variables in $\mathcal{V}$.

Definition 8: Let A be an extended process, $\mathcal{C}$ a set of channels, and $\mathcal{V}$ a set of variables. From A we build the erased process $A' = erase(A,\mathcal{C},\mathcal{V})$ as defined in Figure 4.

Example 5: In Example 4, the augment operation added actions on the channel c_{fr} to the process T to make the process T'. If we erase the channel c_{fr} from the process T' we obtain T again: $erase(T',\{c_{fr}\},\emptyset) \;=\; T$.

Proposition 1: Given a process A, a set of channels $\mathcal{C}$, a new channel $f \notin (\mathcal{C} \cup Chann(A))$, and a term L

$$erase(aug(A,\mathcal{C},f,L),\{f\},\emptyset) \;=\; A$$

Proof: By structural induction over A. ■

We extend the operation $erase()$ to traces of an annotated process.

Definition 9: Let P be an annotated process using the annotation channel f (not used in the original process), and the annotation variables $\mathcal{L}$ (not used in the original process). The erased trace $erase(tr,\mathcal{L})$ is the same trace but with all the actions on the channel f removed:

$$
\begin{array}{l}
erase(P \xrightarrow{\nu\ell.\ \overline{f}\langle l\rangle} tr, L) = erase(tr, L\cup\{\ell\}) \\
erase(P \xrightarrow{\alpha} tr, L) = erase(P,\{f\},L) \xrightarrow{\alpha} erase(tr,L) \\
\hfill \text{if } (bn(\alpha)\cup fn(\alpha))\cap\{f\}=\emptyset
\end{array}
$$

When the set L is clear from the context, we will drop the L and write $erase(tr)$ for $erase(tr,L)$; and with a slight abuse of notion, we define $erase(P) = erase(P,\{f\},\mathcal{L})$.

B. Formalising unlinkability

Unlinkability is informally defined by the ISO/IEC standard 15408 [1] as *ensuring that a user may make multiple uses of a service or resource without others being able to link these uses together.* In terms of the applied pi calculus, this means that an attacker cannot tell when two transitions of a trace tr are initiated by the same user. This will be the case if there exists another trace tr' of the system that looks the same to the attacker, and in which the two corresponding observable transitions are initiated by different users.

We consider the standard Dolev-Yao attacker that can intercept and replay messages, although we allow the possibility of private channels, which are not observable to the attacker. We define two notions of unlinkability. For our definition of "weak unlinkability" we consider an attack to be one in which the attacker can directly link two particular messages as being part of different sessions executed by the same principal. I.e., if an attacker can tell that a principal has used a service twice, but not when the service was used, we do not consider this a linkability attack. On the other hand, our definition of "strong unlinkability" places a much higher burden on the protocol; it requires that a system in which agents execute multiple times always looks the same as a system in which no agent executes more than once. Therefore, if the attacker learns anything about users repeating sessions strong unlinkability will fail.

To give a formal definition of unlinkability, we first need to augment the protocol we consider, in such a way that in any trace each observable transition will be accompanied with information on its initiator. We will then define unlinkability in terms of conditions on the traces of the augmented protocol (also called annotated or augmented traces hereafter).

$$
\begin{array}{lcll}
erase(0,\mathcal{C},\mathcal{V}) & \triangleq & 0 & \\
erase(!P,\mathcal{C},\mathcal{V}) & \triangleq & !(erase(P,\mathcal{C},\mathcal{V})) & \\
erase(\mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q,\mathcal{C},\mathcal{V}) & \triangleq & \mathsf{if}\ M = N\ \mathsf{then}\ erase(P,\mathcal{C},\mathcal{V})\ \mathsf{else}\ erase(Q,\mathcal{C},\mathcal{V}) & \\
erase(u(x).P,\mathcal{C},\mathcal{V}) & \triangleq & \begin{cases} erase(P,\mathcal{C},\mathcal{V}) & \text{if } u \in \mathcal{C} \\ u(x).erase(P,\mathcal{C},\mathcal{V}) & \text{otherwise} \end{cases} & \\
erase(\overline{u}\langle M\rangle.P,\mathcal{C},\mathcal{V}) & \triangleq & \begin{cases} erase(P,\mathcal{C},\mathcal{V}) & \text{if } u \in \mathcal{C} \\ \overline{u}\langle M\rangle.erase(P,\mathcal{C},\mathcal{V}) & \text{otherwise} \end{cases} & \\
erase(A \mid B,\mathcal{C},\mathcal{V}) & \triangleq & erase(A,\mathcal{C},\mathcal{V}) \mid erase(B,\mathcal{C},\mathcal{V}) & \\
erase(\nu u.A,\mathcal{C},\mathcal{V}) & \triangleq & \nu u.\ erase(A,\mathcal{C},\mathcal{V}) & \\
erase(\{M/x\},\mathcal{C},\mathcal{V}) & \triangleq & \begin{cases} 0 & \text{if } x \in \mathcal{V} \\ \{M/x\} & \text{otherwise} \end{cases} &
\end{array}
$$

Fig. 4. The restrict operation

Definition 10 (Protocol transformation $\overline{P}$): Let Σ be a signature and E an equational theory for this signature. Let P be a well-formed p-party protocol over Σ of the form described in Definition 5. Let $\mathcal{C}$ be the set of public channels of P and c_{fr} a channel not occurring in P. We now consider the signature $\Sigma' = \Sigma \uplus \{\mathsf{from}/3, \mathsf{init}/0, \mathsf{R_1}/0, \ldots, \mathsf{R_n}/0\}$ and keep the equational theory E. We build the protocol $\overline{P}$ over Σ' as follows:

$$
\begin{array}{rrcl}
 & \overline{P} & \triangleq & \nu\tilde{n}.(!R'_1 \mid \ldots \mid !R'_p) \\
 & & & \\
\forall i \in \{1,\ldots,p\} & R'_i & \triangleq & \nu id.\ \nu\tilde{m}.\ init'_i.\ (!\nu s.\ main'_i) \\
 & init'_i & \triangleq & aug(init_i, \mathcal{C}, c_{fr}, \mathsf{from}(\mathsf{R_i}, id, \mathsf{init})) \\
 & main'_i & \triangleq & aug(main_i, \mathcal{C}, c_{fr}, \mathsf{from}(\mathsf{R_i}, id, s))
\end{array}
$$

In $\overline{P}$ the roles are modified as follows: each atomic action of role R'_i ($i \in \{1,\ldots,p\}$) is preceded by an output on the channel c_{fr} of a message $\mathsf{from}(R_i, id, s)$ that will dynamically be instantiated to correspond to the identity of the particular user and its current session.

Example 6: If we consider our protocol of Example 2 and apply the transformation we have just described, we obtain the following protocol:

$$
\begin{array}{lcl}
\overline{P} & \triangleq & \nu k.\ (!R' \mid !T' \mid !K') \\
R' & \triangleq & \nu id.\ !(\nu s.\ \overline{c_{fr}}\langle \mathsf{from}(\mathsf{R}, id, s)\rangle.\ in(x)) \\
T' & \triangleq & \nu id.\ !(\nu s.\ \overline{c_{fr}}\langle \mathsf{from}(T, id, s)\rangle.\ \overline{out_T}\langle \mathsf{aenc}(id, \mathsf{pbk}(k))\rangle) \\
K' & \triangleq & \nu id.\ \overline{c_{fr}}\langle \mathsf{from}(\mathsf{K}, id, \mathsf{init})\rangle.\ \overline{out_K}\langle \mathsf{pbk}(k)\rangle
\end{array}
$$

We can now give a formal definition of unlinkability in the applied pi calculus.

Definition 11 (Unlinkability): Let P be a well-formed protocol of the form given in Definition 5. For $R \in \{R_1,\ldots,R_p\}$, P is said to preserve R's *unlinkability* if for all traces $tr \in \mathcal{T}_{\overline{P}}$ of the form:

$$tr \ = \ \overline{P} \overset{(id_1,s_1,R_1):\alpha_1}{\Longrightarrow} A_1 \ \ldots \ A_{n-1} \overset{(id_n,s_n,R_n):\alpha_1}{\Longrightarrow} A_n$$

and for $i,j \in \{1,\ldots,n\}$ it holds that $R_i = R_j = R$ and $s_i \neq \mathsf{init} \neq s_j$, there exists a trace $tr' \in \mathcal{T}_{\overline{P}}$

$$tr' \ = \ \overline{P} \overset{(id'_1,s'_1,R'_1):\alpha_1}{\Longrightarrow} A'_1 \ \ldots \ A'_{m-1} \overset{(id'_n,s'_n,R'_n):\alpha_n}{\Longrightarrow} A'_n$$

such that

- $erase(tr) \sim erase(tr')$, and
- $R'_i \neq R$ or $R'_j \neq R$ or $id'_i \neq id'_j$ or $s'_i = s'_j$.

This definition captures the following intuition: for all observable transitions of tr (α_i and α_j) that are part of the role we are interested in (R) and not part of the initial setup phase (init), there should exist a trace that looks the same to the attacker ($erase(tr) \sim erase(tr')$) in which the two actions are performed by different agents ($id'_i \neq id'_j$) or they are not the role we are interested in ($R'_i \neq R$ or $R'_j \neq R$) or the two actions in question turn out to be part of the same session ($s'_i = s'_j$). Hence, an intruder facing a trace tr cannot be sure that the i^{th} and the j^{th} observable actions are initiated by different sessions of the same user, taking role R, because the attacker could equally well be observing the trace tr'.

Example 7: Our running example doesn't satisfy unlinkability because a tag identifies itself by sending the same, distinct message in every session. If we consider the trace tr_1 of Example 3, any matching trace tr'_1 that we might try to find to for fill the definition of unlinkability will have to perform the same outputs on the channel out_T, in order to be trace equivalent. If we look at the protocol's specification all the outputs on channel out_T are initiated by tags, and of the form $\mathsf{aenc}(id, \mathsf{pbk}(k))$ for some id distinct for each tag. In our augmented process each observable output on channel out_T of the form $\mathsf{aenc}(id, \mathsf{pbk}(k))$ will be preceded by an observable output on c_{fr} of the form $\mathsf{from}(\mathsf{T}, id, s)$ for some s. Hence, in the augmented traces corresponding to tr'_1 and tr_1 the two outputs on channel out_T will be preceded by two outputs on channel c_{fr} stating that the initiator is the same tag. Hence, this protocol does not satisfy the second condition of the above definition of unlinkability.

Our definition of unlinkability is the direct translation from the informal definition of the ISO/IEC standard 15408 [1] to the applied pi calculus. However, it requires us to augment the processes we analyse and uses trace equivalence, which means that it does not lend itself to automated checking with the existing applied pi-calculus tools. This definition is also relatively weak, the attacker may still learn some linkability information from an unlinkable protocol, e.g., if an attacker could tell that two out of four actions where performed by the same user, but not which two, the protocol would still be unlinkable. To address these concerns we give a stronger characterisation of unlinkability based on observational equivalence.

Informally, a protocol preserves strong unlinkability of the users taking the role R if each session of R looks to an outside observer as if it has been initiated by a different user. In other words, an ideal version of the protocol, with respect to, unlinkability would allow users to execute role R at most once. An outside observer should then not be able to tell the difference between the original protocol and the ideal version of this protocol.

Definition 12 (Strong unlinkability): Let Σ be a signature and E an equational theory for this signature, and let P be a well-formed p-party protocol over Σ of the form described at Definition 5. For all $i \in \{1, \ldots, p\}$, we build the protocol P^{R_i} over Σ as follows:

$$\begin{array}{rcl} P^{R_i} & \triangleq & \nu\tilde{n}.\ (!R_1 \mid \ldots \mid !R_{i-1} \mid !R''_i \mid !R_{i+1} \mid \ldots, \mid !R_p) \\ R''_i & \triangleq & \nu id.\ \nu\tilde{m}.\ init_i.\ main_i \end{array}$$

P preserves *strong unlinkability* of R_i if

$$P \approx_l P^{R_i}$$

In the ideal version of the protocol P^{R_i}, users can only take the role R_i at most once, i.e., they can execute at most one session of R_i ($main_i$); this is enforced by removing the replication in front of $main_i$. The other roles of P^{R_i} (R_j with $j \neq i$) are those of the original protocol P, hence allowing users to take them an unbounded number of times.

Note that unlinkability is trivially satisfied for R_i in the ideal protocol P^{R_i}. As each user executes at most one session of role R_i, there are never two sessions of the same user taking role R_i in the system that could be linked. Hence, if $P \approx_l P^{R_i}$, P will necessarily preserve R_i's unlinkability.

Example 8: In the ideal version of our running example we allow tags to execute themselves at most once. Consider again the trace tr_1 of the protocol given in Example 3. If P and P^T were observationally equivalent then, there would be a trace tr'_1 of P^T equivalent to tr_1. However, this is not the case; in any trace of P^T all observable outputs on channel out_T are initiated by tags and of the form $\mathsf{aenc}(id, \mathsf{pbk}(k))$ for some id. Since each tag executes itself at most once, all outputs on channel out_T of any trace of P^T contain a different identity in the place of id and hence are distinct. In other words, there is no trace of P^T with two equal observable outputs on channel out_T, and thus there is no trace of P^T equivalent to tr_1.

In order to support our observational-equivalence-based definition we need to compare the two given definitions of unlinkability. We will show that unlinkability is strictly weaker than strong unlinkability.

Theorem 1 (Unlinkability $\not\Rightarrow$ Strong unlinkability): Let P be a well-formed p-party protocol of the form described in Definition 5, such that P preserves R_i's unlinkability. It is not necessarily the case that P preserves strong unlinkability of R_i.

Proof: To prove this we show that there exists a role of a protocol P preserving unlinkability but not strong unlinkability. Consider the following protocol:

$$\begin{array}{rcl} P & \triangleq & \nu c_{pv}.\ (!R \mid !T) \\ R & \triangleq & \nu id.\ !(\nu s.\ c_{pv}(x).\ c_{pv}(y).\ \mathsf{if}\ x = y\ \mathsf{then}\ \overline{c_{pb}}\langle beep\rangle) \\ T & \triangleq & \nu id.\ !(\nu s.\ \overline{c_{pv}}\langle id\rangle) \end{array}$$

Informally this protocol works as follows: tags (T) identify themselves to readers (R) by sending their identity id on the private, unobservable channel c_{pv}. When a reader identifies the same tag twice it may beep.

The role T preserves unlinkability. Indeed, since all the messages output by tags, are output on the private channel c_{pv}, an outside observer cannot see and thus cannot link any two messages sent by the same tag. More precisely, let tr be a trace of $\overline{P}$. And let's consider tr itself for the trace tr' in the Definition 11. The second condition of the definition of unlinkability only concerns observable actions initiated by tags in tr, and since messages output by tags are not observable (they all occur on the private channel c_{pv}), we have that tr and $tr'(= tr)$ satisfy the second condition. Moreover, since we have considered $tr' = tr$ the first condition of this definition trivially holds ($tr' = tr \sim tr$).

However, T doesn't preserve strong unlinkability. While in the real protocol the reader can *beep* (after seeing the same tag twice), in the ideal version readers never *beep*s (as they will never see the same id more than once). ■

This counter-example also shows that, in some cases, strong unlinkability may be stronger than would be desired. Although an outside observer can learn that a tag has identified itself to a reader at least twice it cannot identify the actions of the tags.

Theorem 2 (Strong unlinkability $\Rightarrow$ Unlinkability): Let P be a well-formed p-party protocol of the form described at Definition 5, and such that P preserves strong unlinkability of the role R_i, then P also preserves unlinkability of R_i.

Proof: Due to lack of space, we only present a sketch of the proof. Let tr_1 be a trace of $\mathcal{T}_{\overline{P}}$, and $tr'_1 = erase(tr_1)$ its corresponding trace in $Traces(P)$. By strong unlinkability we know that there exists a trace $tr'_2 \in Traces(P^{R_i})$ such that $tr'_2 \sim tr'_1$. By construction of the augmented protocol $\overline{P^{R_i}}$ and by definition of $\mathcal{T}_{\overline{P^{R_i}}}$, we have that the corresponding trace tr_2 of tr'_2 is such that if two messages of the form $\mathsf{from}(R_i, id, s_1)$ and $\mathsf{from}(R_i, id, s_2)$ are outputted on channel c_{fr}, then $s_1 = s_2$. Now because each trace of $\overline{P^{R_i}}$ can be mimicked by an equivalent trace of $\overline{P}$, we know that there exists a trace tr_3 of $\overline{P}$ such that $tr_3 \sim tr_2$ and such that if two messages of the form $\mathsf{from}(R_i, id, s_1)$ and $\mathsf{from}(R_i, id, s_2)$ are output on channel c_{fr}, then $s_1 = s_2$; hence tr_1 and tr_3 satisfy the second condition of Definition 11. Finally since $tr_2 \in \mathcal{T}_{\overline{P^{R_i}}}$ and $tr_3 \sim tr_2$, we can conclude that $tr_3 \in \mathcal{T}_{\overline{P}}$ and $erase(tr_3) \sim erase(tr_2) (\sim erase(tr_1))$. Thus tr_1 and tr_3 also satisfy the first condition. ■

C. Formalising anonymity

Anonymity is informally defined by the ISO/IEC standard 15408 [1] as *ensuring that a user may use a service or resource without disclosing the user's identity.* In this sense, anonymity is not intended to protect the subject's identity, but rather the link between a use of a service and the identity of the user. In terms of the applied pi calculus, this means that an attacker should not be able to tell when a transition of a trace is initiated by one user or by another. This will be the case if, for every trace, there exists another trace of the system that looks the same to the attacker, and in which the corresponding observable transitions are initiated by a different user.

To give a formal definition of anonymity, we need to augment the considered protocol with information on the initiator of each action. However we will proceed in a slightly different way, because in order for anonymity to be broken for a user u, the intruder needs to have known u's identity. Hence, to express that the role R_i preserves anonymity we need to give the intruder access to the identity of a particular user taking this role.

Definition 13 (Protocol Transformations P_{R_i}): Let Σ be a signature and E an equational theory for this signature. Let P be a well-formed p-party protocol over Σ of the form described above in Definition 5. From P, we build the protocol P_{R_i} over Σ as follows:

$$\begin{array}{rcl} P_{R_i} & \triangleq & \nu\tilde{n}.\ (!R_1 \mid \ldots \mid !R_p \mid R_w) \\ R_w & \triangleq & \nu\tilde{m}.\ init_w.\ !(\nu s.\ main_w) \\ init_w & \triangleq & init_i\{id_w/id\} \\ main_w & \triangleq & main_i\{id_w/id\} \end{array}$$

where id_w is a name not occurring in P.

R_w models a user with identity id_w taking role R_i. This identity id_w is public. Intuitively, P will be said to preserve anonymity of R_i if in any trace of P_{R_i}, an outside observer cannot distinguish id_w's actions.

Definition 14 (Protocol Transformations: $\overline{P_{R_i}}$): Let Σ be a signature and E an equational theory for this signature. Let P be a well-formed p-party protocol over Σ, let $\mathcal{C}$ be its set of public channels, and c_{fr} a channel not occurring in P. We consider the signature $\Sigma' = \Sigma \uplus \{\mathsf{from}/3, \mathsf{init}/0, \mathsf{R_1}, \ldots, \mathsf{R_p}\}$ and keep the equational theory E. We extend the $\overline{\cdot}$ operation to processes of the form P^{R_i} as follows:

$$\begin{array}{rcl} \overline{P_{R_i}} & \triangleq & \nu\tilde{n}.(!R'_1 \mid \ldots \mid !R'_p \mid R'_w) \\ & & \\ R'_w & \triangleq & \nu\tilde{m}.init'_w.!(\nu s.\ main'_w) \\ init'_w & \triangleq & aug(init_w, \mathcal{C}, c_{fr}, \mathsf{from}(\mathsf{R_i}, id_w, \mathsf{init})) \\ main_w & \triangleq & aug(main_w, \mathcal{C}, c_{fr}, \mathsf{from}(\mathsf{R_i}, id_w, s)) \end{array}$$

where R'_j for all $j \in \{1, \ldots, p\}$ are as defined in Definition 10.

Example 9: Continuing with our running example:

$$\begin{array}{rcl} \overline{P_T} & \triangleq & \nu k.\ (!R' \mid !T' \mid !K' \mid T'_w) \\ T'_w & \triangleq & !(\nu s.\ \overline{c_{fr}}\langle \mathsf{from}(\mathsf{T}, id_w, s)\rangle.\ \overline{out_T}\langle \mathsf{aenc}(id_w, \mathsf{pbk}(k))\rangle) \end{array}$$

with R', T', and K' are as defined in Example 6.

Definition 15 (Anonymity): Let P be a well-formed p-party protocol of the form described at Definition 5. Let $R \in \{R_1, \ldots, R_p\}$, P is said to preserve R's anonymity if for all traces $tr \in \mathcal{T}_{\overline{P_R}}$

$$tr\ = \overline{P_R} \overset{(id_1, s_1, R_1):\alpha_1}{\Longrightarrow} A_1\ \ldots\ A_{n-1} \overset{(id_n, s_n, R_n):\alpha_1}{\Longrightarrow} A_n$$

and for all $j \in \{0, \ldots, n\}$ such that $id_j = id_w$, $s_j \neq \mathsf{init}$ and $R_j = R$ there exists a trace $tr' \in \mathcal{T}_{\overline{P_R}}$:

$$tr'\ = \overline{P_R} \overset{(id'_1, s'_1, R'_1):\alpha_1}{\Longrightarrow} A'_1\ \ldots\ A'_{m-1} \overset{(id'_n, s'_n, R'_n):\alpha_n}{\Longrightarrow} A'_n$$

such that $erase(tr) \sim erase(tr')$, and $id'_j \neq id_w$.

This definition captures the following: for all observable transitions (α_j) originating from a session (s) of the user with identity id_w taking role R, there exists another trace of the system tr' which looks the same to the attacker ($erase(tr) \sim erase(tr')$) but in which the matching action was initiated by some other user ($id'_j \neq id_w$). Hence, attackers faced with the trace $erase(tr)$ cannot be sure that it is id_w initiating the j^{th} transition; as they could just as well be facing trace $erase(tr')$.

Example 10: We illustrate this definition by showing that our running example does not preserve a tag's anonymity. Consider the trace tr where the intruder gets $\mathsf{pbk}(k)$ on out_K, and then the tag with identity id_w executes one session of the protocol:

$$\begin{array}{rcl} tr & = & P_T \\ \xrightarrow{\nu z_1.\ \overline{out_K}\langle z_1\rangle} \quad A_1 & = & \begin{array}{l} \nu id_K.\ \nu k. \\ \left(\begin{array}{l} !R \mid !T \mid !K \mid T_w \mid \\ \{\mathsf{pbk}(k)/z_1\} \end{array} \right) \end{array} \\ \xrightarrow{\nu z_2.\ \overline{out_T}\langle z_2\rangle} \quad A_2 & = & \begin{array}{l} \nu id_K.\ \nu s_w.\ \nu k. \\ \left(\begin{array}{l} !R \mid !T \mid !K \mid T_w \mid \\ !(\nu s.\ \overline{out_T}\langle \mathsf{aenc}(id_w, \mathsf{pbk}(k))\rangle) \mid \\ \{\mathsf{pbk}(k)/z_1\} \mid \\ \{\mathsf{aenc}(id_w, \mathsf{pbk}(k))/z_2\} \end{array} \right) \end{array} \end{array}$$

Figure 5 shows the corresponding augmented trace of $\mathcal{T}_{\overline{P_T}}$. If there exists a trace $tr'' \in Traces(P_T)$ such that $tr \sim tr''$, then tr'' must contain exactly one output on the channel out_K followed by one output on channel out_T. By inspecting the

$$
\begin{array}{rl}
tr' \;=\; & \overline{P_T} \\
\xrightarrow{\qquad} & \nu id_K.\ \nu k. \\
& \left(\begin{array}{l} !R' \mid !T' \mid !K' \mid T'_w \mid \\ !(c_{revoke}(y).\ \overline{c_{grant}}\langle\star\rangle) \mid \\ \overline{out_K}\langle\mathsf{pbk}(k)\rangle. \\ \qquad \overline{c_{from}}\langle\mathsf{from}(\mathsf{K}, id_K, \mathsf{init})\rangle.\ \overline{c_{revoke}}\langle\star\rangle \end{array} \right) \\
\xrightarrow{\nu z_1.\ \overline{out_K}\langle z_1\rangle} & \nu id_K.\ \nu k. \\
& \left(\begin{array}{l} !R' \mid !T' \mid !K' \mid T'_w \mid \\ !(c_{revoke}(y).\ \overline{c_{grant}}\langle\star\rangle) \mid \\ \overline{c_{from}}\langle\mathsf{from}(\mathsf{K}, id_K, \mathsf{init})\rangle.\ \overline{c_{revoke}}\langle\star\rangle \mid \\ \{\mathsf{pbk}(k)/z_1\} \end{array} \right) \\
\xrightarrow{\nu z_2.\ \overline{c_{from}}\langle z_2\rangle} & \nu id_K.\ \nu k. \\
& \left(\begin{array}{l} !R' \mid !T' \mid !K' \mid T'_w \mid \\ !(c_{revoke}(y).\ \overline{c_{grant}}\langle\star\rangle) \mid \\ \overline{c_{revoke}}\langle\star\rangle \mid \\ \{\mathsf{pbk}(k)/z_1\} \mid \{\mathsf{from}(\mathsf{K}, id_K, \mathsf{init})/z_2\} \end{array} \right) \\
\xrightarrow{\qquad} & \nu id_K.\ \nu k. \\
& \left(\begin{array}{l} !R' \mid !T' \mid !K' \mid S \mid T'_w \mid \\ \{\mathsf{pbk}(k)/z_1\} \mid \{\mathsf{from}(\mathsf{K}, id_K, \mathsf{init})/z_2\} \end{array} \right) \\
\xrightarrow{\qquad} & \nu id_K.\ \nu s_w .\nu k. \\
& \left(\begin{array}{l} !R' \mid !T' \mid !K' \mid T'_w \mid \\ \overline{out_T}\langle\mathsf{aenc}(id_w, \mathsf{pbk}(k))\rangle. \\ \qquad \overline{c_{from}}\langle\mathsf{from}(\mathsf{R_i}, id_w, s_w)\rangle.\ \overline{c_{revoke}}\langle\star\rangle \mid \\ !(c_{revoke}(y).\ \overline{c_{grant}}\langle\star\rangle) \mid \\ \{\mathsf{pbk}(k)/z_1\} \mid \{\mathsf{from}(\mathsf{K}, id_K, \mathsf{init})/z_2\} \end{array} \right) \\
\xrightarrow{\nu z_3.\ \overline{out_T}\langle z_3\rangle} & \nu id_K.\ \nu s_w .\nu k. \\
& \left(\begin{array}{l} !R' \mid !T' \mid !K' \mid T'_w \mid \\ \overline{c_{from}}\langle\mathsf{from}(\mathsf{R_i}, id_w, s_w)\rangle.\ \overline{c_{revoke}}\langle\star\rangle \mid \\ !(c_{revoke}(y).\ \overline{c_{grant}}\langle\star\rangle) \mid \\ \{\mathsf{pbk}(k)/z_1\} \mid \{\mathsf{from}(\mathsf{K}, id_K, \mathsf{init})/z_2\} \mid \\ \{\mathsf{aenc}(id_w, \mathsf{pbk}(k))/z_3\} \end{array} \right) \\
\xrightarrow{\nu z_4.\ \overline{c_{from}}\langle z_4\rangle} & \nu id_K.\ \nu s_w .\nu k. \\
& \left(\begin{array}{l} !R' \mid !T' \mid !K' \mid T'_w \mid \\ \overline{c_{revoke}}\langle\star\rangle \mid \\ !(c_{revoke}(y).\ \overline{c_{grant}}\langle\star\rangle) \mid \\ \{\mathsf{pbk}(k)/z_1\} \mid \{\mathsf{from}(\mathsf{K}, id_K, \mathsf{init})/z_2\} \mid \\ \{\mathsf{aenc}(id_w, \mathsf{pbk}(k))/z_3\} \mid \\ \{\mathsf{from}(\mathsf{R_i}, id_w, s_w)/z_4\} \end{array} \right) \\
\xrightarrow{\qquad} & \nu id_K.\ \nu s_w .\nu k. \\
& \left(\begin{array}{l} !R' \mid !T' \mid !K' \mid S \mid T'_w \mid \\ \{\mathsf{pbk}(k)/z_1\} \mid \{\mathsf{from}(\mathsf{K}, id_K, \mathsf{init})/z_2\} \mid \\ \{\mathsf{aenc}(id_w, \mathsf{pbk}(k))/z_3\} \mid \\ \{\mathsf{from}(\mathsf{R_i}, id_w, s_w)/z_4\} \end{array} \right)
\end{array}
$$

Fig. 5. Augmented trace of Example 10

specification of the P_T we know that all traces tr'' would then be of the form:

$$
\begin{array}{rl}
tr'' \;=\; & P_T \\
\Rightarrow\xrightarrow{\nu z_1.\ \overline{out_K}\langle z_1\rangle} & A'_1 \equiv \nu\tilde{n}.\ (Q \mid \{\mathsf{pbk}(k)/z_1\}) \\
\Rightarrow\xrightarrow{\nu z_2.\ \overline{out_T}\langle z_2\rangle} & A'_2 \equiv \nu\tilde{m}. \left(\begin{array}{l} R \mid \{\mathsf{pbk}(k)/z_1\} \mid \\ \{\mathsf{aenc}(id, \mathsf{pbk}(k))/z_2\} \end{array} \right)
\end{array}
$$

If $id = id_w$ then the corresponding augmented trace will not satisfy the second condition of Definition 15, and otherwise $tr'' \not\sim tr$ because $\phi(A_2) \not\approx_s \phi(A'_2)$. Indeed, if we consider z_2 and $\mathsf{aenc}(id_w, z_1)$, these two terms are equal in the frame $\phi(A_2)$ but not in the frame $\phi(A'_2)$ if $id \neq id_w$.

Now, for the same reasons that lead us to the stronger definition of unlinkability, we give a stronger characterisation of anonymity in terms of observational equivalence.

Definition 16 (Strong anonymity): Let P be a well-formed p-party protocol of the form described at Definition 5. P is said to preserve strong anonymity of R_i's if

$$P \;\approx_l\; P_{R_i}$$

where P_{R_i} is as defined at Definition 15.

Defined in this way, strong anonymity ensures that an outside observer does not see the difference between the system P_{R_i} (with user id_w taking role R_i) and the original system P (where the user id_w is not present). In this case, P is the ideal version of the protocol for user id_w. Indeed, since id_w is not present in the system P, her anonymity is trivially preserved.

Example 11: To illustrate this definition, we will show that our protocol P of Example 2 does not preserve strong Anonymity for tags. For this, consider the trace tr from Example 10. If P did preserve strong anonymity of tags, there would be a trace tr' of P such that $tr' \sim tr$. However, all traces tr' of P with exactly one observable output on channel out_K followed by an observable output on channel out_T are of the form of tr'' presented in the previous Example 10. We are thus faced with the same situation as in Example 10, and can mimic the same argument to conclude that there exists no trace tr' of P such that $tr' \sim tr$. In consequence, $P_T \not\approx_l P$ and strong anonymity does not hold.

To support our observational-equivalence-based definition we compare the two given definitions of anonymity. We show that anonymity is strictly weaker than strong anonymity:

Theorem 3 (Anonymity $\not\Rightarrow$ Strong anonymity): Let P be a well-formed p-party protocol of the form described at Definition 5, such that P preserves R_i's anonymity. It is not necessarily the case that P preserves R_i's strong anonymity.

Proof: To prove this we show that there exists a role of a protocol P preserving anonymity but not strong anonymity. Consider the following protocol:

$$
\begin{array}{rcl}
P & \triangleq & \nu c_{pv}.\ (!R \mid !T) \\
R & \triangleq & \nu id.\ !(\nu s.\ c_{pv}(x).\overline{c_{pb}}\langle x\rangle) \\
T & \triangleq & \nu id.\ !(\nu s.\ \overline{c_{pv}}\langle id\rangle)
\end{array}
$$

Informally this protocol works as follows: tags (T) identify themselves to readers (R) by sending their identity id on the private channel c_{pv}. Readers can output the identity of tags they have previously identified.

Tags' anonymity is preserved by this protocol. Indeed, since all the messages output by the tags, are outputted on the private channel c_{pv}, an outside observer cannot see and thus cannot link any message sent by a tag to its identity. More precisely, let tr be a trace of $\mathcal{T}_{\overline{P_T}}$ and we consider tr as the trace tr' in the Definition 15. The second condition of the definition of anonymity only concerns observable actions initiated by tags in tr, and since messages output by tags are not observable (they all occur on the private channel c_{pv}), it is necessary that tr and $tr' = tr$ satisfy the second condition. Moreover, since we have considered $tr' = tr$, the first condition of this definition trivially holds.

However, T doesn't preserve strong anonymity. While in P_T the readers can output id_w, in P they can output any identity except id_w. Hence the traces outputting id_w on c_{pb} cannot be mimicked by any trace of P, and since id_w is public, an outside observer can detect this. In consequence, $P_T \not\approx_l P$ and strong anonymity does not hold. ■

Theorem 4 (Strong anonymity $\Rightarrow$ Anonymity): Let P be a well-formed p-party protocol of the form described in Definition 5, and such that the role P preserves strong anonymity of R_i, then P also preserves R_i's anonymity.

Proof: Due to lack of space, we only present a sketch of the proof. Let tr_1 be a trace of $\mathcal{T}_{\overline{P_{R_i}}}$, and $tr'_1 = erase(tr_1)$ its corresponding trace in $Traces(P_{R_i})$. By strong anonymity we know that there exists a trace $tr'_2 \in Traces(P)$ such that $tr'_2 \sim tr'_1$. By construction of the augmented protocol $\overline{P}$ we have that the corresponding trace tr_2 of tr'_2 is such that if a message is of the form $\mathsf{from}(R_i, id, s)$ is output on channel c_{fr}, then $id \neq id_w$. Now, because each trace of $\overline{P}$ can be mimicked by an equivalent trace of $\overline{P_{R_i}}$, we know that there exists a trace tr_3 of $\overline{P_{R_i}}$ such that $tr_3 \sim tr_2$ and such that if a messages of the form $\mathsf{from}(R_i, id, s)$ is output on channel c_{fr}, then $id \neq id_w$; hence tr_1 and tr_3 satisfy the second condition of Definition 15. Finally since $tr_2 \in \mathcal{T}_{\overline{P}}$ and $tr_3 \sim tr_2$, we can conclude that $tr_3 \in \mathcal{T}_{\overline{P_{R_i}}}$ and $erase(tr_3) \sim erase(tr_2)(\sim erase(tr_1))$. Thus tr_1 and tr_3 also satisfy the first condition of anonymity. ■

D. *Unlinkability* vs. *Anonymity*

Having shown that unlinkability (*resp.* anonymity) is strictly weaker than strong unlinkability (*resp.* strong anonymity), we now complete the picture by comparing unlinkability (*resp.* strong unlinkability) with anonymity (*resp.* strong anonymity). In particular, we show that unlinkability does not imply anonymity, contradictory to what had been suggested by other authors [13].

Theorem 5 (Unlinkability $\not\Rightarrow$ Anonymity): Let P be a well-formed p-party protocol of the form described at Definition 5, and such that P preserves (strong) unlinkability of R_i. It is not necessarily the case that P preserves R_i's (strong) anonymity.

Proof: To prove this, we show that there exists a protocol P preserving strong unlinkability (and unlinkability by Theorem 2) of one of its roles R, but not their anonymity (nor their strong anonymity, by Theorem 4). Consider the following protocol:

$$\begin{array}{lcl} P & \triangleq & \nu c_{pv}.\ \nu d_{pv}.\ (!T \mid !R) \\ R & \triangleq & \nu id.\ !(\nu s.\ c_{pv}(x).\ \overline{d_{pv}}\langle x\rangle) \\ T & \triangleq & \nu id.\ \overline{c_{pv}}\langle id\rangle.\ !(\nu s.\ d_{pv}(x).\ \mathsf{if}\ x = id\ \mathsf{then}\ \overline{c}\langle id\rangle) \end{array}$$

Informally, this protocol works as follows: tags (T) register themselves to a reader (R) through channel c_{pv}. The readers grant permission through channel d_{pv} to tags that have previously registered themselves to publicly output their identity. However, each tag gets permission to publicly output its identity only once.

Now, let's consider a trace of the protocol. What an intruder observes is a sequence of distinct tag identifiers output on channel c_{pb}. All these outputs are distinct and are initiated by distinct tags. Hence this protocol preserves strong unlinkability (and hence unlinkability by Theorem 2). However, since tags publicly output their identity (only once in their whole life), tags' anonymity (and tags' strong anonymity by Theorem 4) is not preserved by this protocol.

In consequence unlinkability does not imply anonymity. ■

Theorem 6 (Anonymity $\not\Rightarrow$ Unlinkability): Let P be a well-formed p-party protocol of the form described in Definition 5 such that P preserves (strong) anonymity of R_i. It is not necessarily the case that P preserves R_i's (strong) unlinkability.

Proof: To prove this, we show that there exists a protocol P preserving strong anonymity (and hence anonymity by Theorem 4) of one of its roles R, but not their unlinkability (nor their strong unlinkability, by Theorem 2). Consider the following protocol:

$$P \triangleq \nu k.\ !T \qquad\qquad T \triangleq \nu id.\ !(\nu s.\ \overline{c}\langle \mathsf{senc}(id, k)\rangle)$$

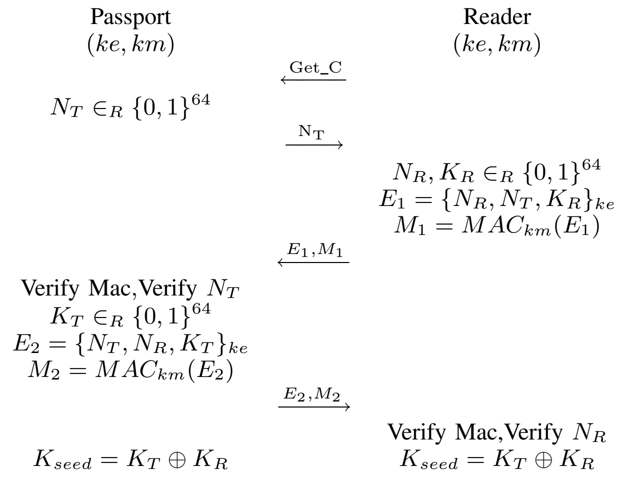
Informally, this protocol works as follows: tags T share a common symmetric key k, and identify themselves by publicly outputting their identity symmetrically encrypted with k.

It is obvious that this protocol doesn't preserve strong unlinkability of tags (and by Theorem 2 nor their unlinkability). Indeed, in each session, tags identify themselves with the a distinct message.

However, this protocol preserves strong anonymity (and thus anonymity) since the considered intruder in this example doesn't control any tag and hence doesn't know the key k. Therefore, although it knows when two messages come from the same tag, it can not infer the tag's identity. ■

We conclude this section with the following picture that summarises our results on anonymity and unlinkability.

$$\begin{array}{ccc} \text{Strong Unlinkability} & \begin{array}{c}\not\Rightarrow\\ \not\Leftarrow\end{array} & \text{Strong Anonymity} \\ \Downarrow\ \ \not\Uparrow & & \Downarrow\ \ \not\Uparrow \\ \text{Unlinkability} & \begin{array}{c}\not\Rightarrow\\ \not\Leftarrow\end{array} & \text{Anonymity} \end{array}$$



a) In Alice & Bob notation

```
$Reader \triangleq$
  $c_k(x_k).$ let $ke = \pi_1(x_k)$
      in let $km = \pi_2(x_k)$
          in $\overline{c}\langle get\_challenge\rangle.\ d(nt).\ \nu nr.\ \nu kr.$
              let $m = \mathsf{enc}((nr, nt, kr), ke)$
              in $\overline{c}\langle m, \mathsf{mac}(m, km)\rangle.\ d(y)$

$MainFR \triangleq$
  $\overline{c_k}\langle (ke, km)\rangle.$
  $d(x).$ if $x = get\_challenge$
      then $\nu nt.\ \overline{c}\langle nt\rangle.\ d(y).$
          let $me = \pi_1(y)$
          in let $m_m = \pi_2(y)$
              in if $m_m = \mathsf{mac}(m_e, km)$
                  then let $nr = \pi_1(\mathsf{dec}(m_e, ke))$
                      in let $nt' = \pi_1(\pi_2(\mathsf{dec}(m_e, ke)))$
                          in if $nt' = nt$
                              then $\nu kt.$
                                  let $m = \mathsf{enc}((nt, nr, kt), ke)$
                                  in $\overline{c}\langle (m, \mathsf{mac}(m, km))\rangle$
                              else $\overline{c}\langle 6A80\rangle$
                  else $\overline{c}\langle 6300\rangle$

$SystemFR \triangleq\ \nu c_k.\ (!Reader \mid !\nu ke.\ \nu km.\ !MainFR)$
```

b) In applied pi calculus

Fig. 6. The Basic Access Control Protocol

IV. Case Study: The e-Passport

An e-Passport is a passport with an embedded RFID tag; over 40 countries have, together, issued many millions of e-Passports. The RFID tag stores the information printed on the passport and a JPEG copy of the picture. The tags also have the capacity to store fingerprints and iris scans, although these are rarely used. The ICAO publishes the specification for e-Passports [21] and each nation has implemented their own version. As the specification is not completely comprehensive, each nation's passport has subtle differences.

RFID tags use a unique identifier (UID) to establish a communication channel. The French passport randomises its UID for each new session. Read access to the data on the passport is protected by the Basic Access Control (BAC) protocol. This protocol produces a session key by using another key derived from the date of birth, date of expiry and the passport number printed on the document. The aim of this design is to make the passport unlinkable and to ensure that only parties with physical access to the passport can read the data.

BAC is a three-pass key establishment protocol, as shown in Figure 6.*a*. Here $\{-\}_K$ denotes Triple-DES encryption with the key K and $MAC_K(-)$ denotes a cryptographic checksum Message Authentication Code. The passport stores two keys: ke and km. The reader derives these keys using the machine-readable information on the passport, which has, in theory, been scanned before the wireless communication begins. The reader initiates the protocol by sending a challenge to the tag and the tag replies with a random 64-bit string N_T. The reader then creates its own random nonce and some new random key material, both 64-bits. These are encrypted, along with the tag's nonce and sent back to the reader. A MAC is computed using the km key and sent along with the message, to ensure the message is received correctly. The tag receives this message, verifies the MAC, decrypts the message and checks that its nonce is correct; this guarantees to the tag that the message from the reader is not a replay of an old message. The tag then generates its own random 64-bits of key material and sends this back to the reader in a similar message, except this time the order of the nonces is reversed, this stops the reader's message being replayed directly back to the reader. The reader checks the MAC and its nonce, and both the tag and the reader use the xor of the key material as the seed for a session key, with which to encrypt the rest of the session.

The ICAO e-Passport standard [21] specifies that the passport must reply with an error message to every ill formed or incorrect message from the reader, but it does not specify what the error message should be. To find the error messages, we experimented with a French passport and found that it responded to an incorrect MAC with the error code "6300", which means "no information given". If the MAC was correct, and the passport went on to find that the nonce did not match then it responded with an error code "6A80", meaning "incorrect parameters".

In Figure 6.*b* we give our applied pi-calculus model of the French implementation of the BAC protocol. We used the notation let $u = M$ in P for the process obtained by substituting in P all the free occurrences of the name or variable u by the term M, *i.e.* for $P\{M/u\}$. The process

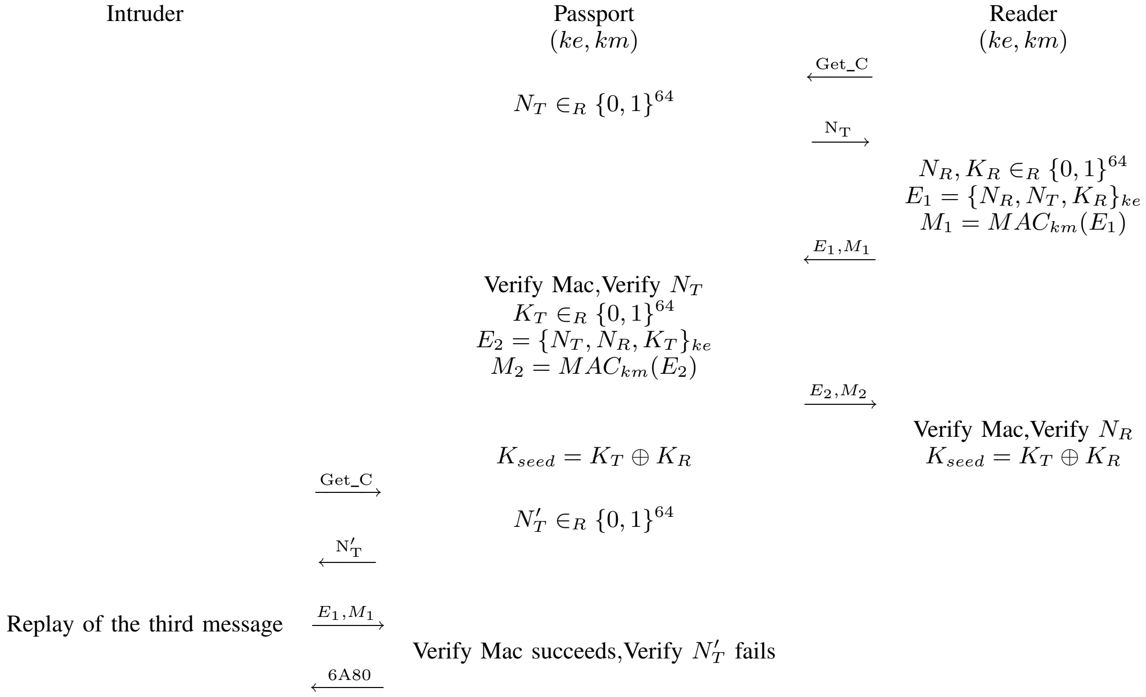

Intruder
Passport
(ke, km)
Reader
(ke, km)
Get_C
$N_T \in_R \{0,1\}^{64}$
N_T
$N_R, K_R \in_R \{0,1\}^{64}$
$E_1 = \{N_R, N_T, K_R\}_{ke}$
$M_1 = MAC_{km}(E_1)$
E_1, M_1
Verify Mac,Verify N_T
$K_T \in_R \{0,1\}^{64}$
$E_2 = \{N_T, N_R, K_T\}_{ke}$
$M_2 = MAC_{km}(E_2)$
E_2, M_2
Verify Mac,Verify N_R
$K_{seed} = K_T \oplus K_R$
$K_{seed} = K_T \oplus K_R$
Get_C
$N'_T \in_R \{0,1\}^{64}$
N'_T
Replay of the third message
E_1, M_1
Verify Mac succeeds,Verify N'_T fails
6A80

Fig. 7. Attack on the French implementation of BAC

SystemFR represents an arbitrary number of passports each being read an arbitrary number of times. The tag starts off by passing its encryption and MAC keys to the reader on a private channel, this represents the reader generating the keys from the machine readable information on the passport, which, when done correctly, should not be observable by the attacker. All following messages between the tag and the reader are sent on the public channel c, reflecting that these messages are sent wirelessly. In this model we are looking at just the BAC protocol, a more comprehensive model for the e-passport could include the transfer of data that happens after the BAC protocol, so allowing an attacker to see if the reader has accepted the last message from the tag.

A. Attack on the French implementation of the BAC protocol

Due to lack of space, we only give an informal description of the unlinkability attack on the French implementation of the BAC protocol. According to Definition 11, such an attack consists in a trace of *SystemFR* with two of its observable actions initiated by the same passport tag, but which isn't equivalent to any trace of *SystemFR* with the corresponding observable actions not initiated by the same passport tag. The attack trace is given in Alice & Bob notation in Figure 7, and works as follows.

First, the protocol runs, and all the messages generated by the passport and the reader are recorded by the intruder. Then, in the *SystemFR* process, the attacker starts another run of the protocol with the same tag. After sending the $get\_challenge$ message, the message from the reader in the previous round is replayed to the passport. At this point the *SystemFR* process broadcasts the 6A80 error code. In this trace the 3^{rd} message is received by the same passport tag that has sent the error message at the 8^{th} step. However, there is no possible equivalent trace of *SystemFR* with the corresponding two actions not initiated by the same passport tag.

This failing of unlinkability in our framework leads directly to a real life attack on anyone carrying a French e-Passport. To detect a particular French e-Passport the attacker must first record the encrypted message sent to the passport by a legitimate reader. That passport can then be distinguished from any other passport by first sending a $get\_challenge$ message and then replaying the recorded message, if the passport replies with a 6300 error code then we know that the MAC check, which uses the passport's unique MAC key failed, and therefore it's not the target passport. If the attacker sees the 6A80 error code, they know that the MAC check passed and then the nonce check failed, meaning that the passport is using the targets unique MAC key. We have tested this attack and found that it works in practice, as well as in theory. The range of our RFID reader was only 9cm meaning that the danger of the attack is limited, however work by Juels et al. [20] shows that more powerful readers can have a much greater range.

B. Thwarting the attack of the French e-passport

We also tested e-passports from the UK, Germany, Ireland and Russia. All of these passports replied with the same error code for a failed nonce and a failed MAC check (usually a 6300 message, never a 6A80 message). We can model these passports by changing the error codes in Figure 6.b to both be 6300, leading to the process *SystemUK*:

$$
\begin{aligned}
MainUK &\triangleq \text{let } A680 = 6300 \text{ in } MainFR \\
SystemUK &\triangleq \nu c_k.\ (!Reader \mid !\nu ke.\ \nu km.\ !MainUK)
\end{aligned}
$$

and its idealised version *SystemUK'*:

$$SystemUK' \triangleq \nu c_k.\ (!Reader \mid !\nu ke.\ \nu km.\ MainUK)$$

Checking the bisimulation by hand, we find that $SystemUK \approx_l SystemUK'$ holds: A repeating tag in the *SystemUK* process is matched by a new tag in the idealised *SystemUK'* version of the system. As both error messages are the same, these processes are now indistinguishable to the attacker for all possible inputs. Therefore, *SystemUK* $\approx_l$ *SystemUK'* and the protocol is safe since the stronger definition of unlinkability holds.

Finding that unlinkability holds for the protocol does not rule out faults introduced by the implementation. Many authors have written about implementational flaws in e-Passports; problems include low entropy keys, the inability to revoke access, the ability to detect the country that issued the passport, and time-based linkability attacks. We give more details on these in a previous paper [11].

V. Conclusion

We have presented an analysis framework for unlinkability and anonymity. Our use of the applied pi-calculus as the basis for this framework has made our definitions precise and our analysis of these definitions rigorous. We defined weak and strong versions of unlinkability and anonymity and shown that anonymity and unlinkability are independent properties. The strong versions hold if a system is bisimilar to an idealised version of itself, and are easy to check automatically. The weak versions are harder to check but a failure of the weak versions implies a practical attack against the system. We show that our strong definitions imply the weak versions. This means that when checking a system we can try to check the strong definitions first, and if they fail we can go on to use the weak definitions to look for attacks.

As with any framework, the true test of our definitions is how useful they are when it comes to analysing real systems. We demonstrate the utility of our framework with case study on the French e-passport. In the course of this work we found a new linkability attack that makes it possible to detect the presence of a particular French e-Passport.

For further work, we intend to use our framework to analyse a range of devices and look for new linkability attacks, we also speculate that it is possible to define “anonymity with a single failure” and show that unlinkability implies at most one failure of anonymity.

Acknowledgement

This work has been partially supported by the EPSRC project *Verifying Interoperability Requirements in Pervasive Systems* (EP/F033540/1).

References

[1] ISO 15408-2: Common Criteria for Information Technology Security Evaluation - Part 2: Security functional components. Final draft, ISO/IEC, July 2009.

[2] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. *SIGPLAN Not.*, 36(3):104–115, 2001.

[3] Myrto Arapinis, Tom Chothia, Eike Ritter, and Mark Ryan. Untraceability in the applied pi-calculus. In *RISC09: Proceedings the 1st International Workshop on RFID Security and Cryptography*. IEEE, 2009.

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

[5] Gildas Avoine, Etienne Dysli, and Philippe Oechslin. Reducing time complexity in RFID systems. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography – SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 291–306, Kingston, Canada, August 2005.

[6] Michael Barbaro and Tom Zeller Jr. A face is exposed for aol searcher no. 4417749. The New York Times, August 9, 2006.

[7] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *CSFW '01: Proceedings of the 14th IEEE workshop on Computer Security Foundations*, page 82, Washington, DC, USA, 2001. IEEE Computer Society.

[8] Mike Burmester, Tri van Le, and Breno de Medeiros. Provably Secure Ubiquitous Systems: Universally Composable RFID Authentication Protocols. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm*, Baltimore, Maryland, USA, August-September 2006. IEEE.

[9] Christopher Caldwell. A pass on privacy? The New York Times, July 17, 2005.

[10] Tom Chothia. Analysing the MUTE anonymous file-sharing system using the pi-calculus. In *FORTE*, volume 4229 of *LNCS*, pages 115–130. Springer, 2006.

[11] Tom Chothia and Vitaliy Smirnov. A traceability attack against e-passports. In *FC10: Proceedings of the 14th International Conference on Financial Cryptography and Data Security 2010*. LNCS, Springer-Verlag, 2010.

[12] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 2009.

[13] Ton van Deursen, Sjouke Mauw, and Saša Radomirović. Untraceability of RFID protocols. In *Information Security Theory and Practices. Smart Devices, Convergence and Next Generation Networks*, volume 5019. 5019 of *Lecture Notes in Computer Science*, page 115. Springer, 2008.

[14] Ton van Deursen and Saša Radomirović. Algebraic attacks on rfid protocols. In *WISTP '09: Proceedings of the 3rd IFIP WG 11.2 International Workshop on Information Security Theory and Practice. Smart Devices, Pervasive Systems, and Ubiquitous Networks*, pages 38–51. Springer-Verlag, 2009.

[15] Flavio D. Garcia, Ichiro Hasuo, Wolter Pieters, and Peter van Rossum. Provable anonymity. In *Proceedings of the 3rd ACM Workshop on Formal Methods in Security Engineering (FMSE05)*, Alexandria, VA, USA, November 2005.

[16] Simson L. Garfinkel, Ari Juels, and Ravi Pappu. RFID privacy: An overview of problems and proposed solutions. *IEEE Security and Privacy*, 3(3):34–43, 2005.

[17] Dan Goodin. Defects in e-passports allow real-time tracking. The Register, 26th January 2010.

[18] Ari Juels. RFID security and privacy: a research survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, 2006.

[19] Ari Juels and Stephen A. Weis. Defining strong privacy for RFID. In *PERCOMW '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 342–347, Washington, DC, USA, 2007. IEEE Computer Society.

[20] Karl Koscher, Ari Juels, Vjekoslav Brajkovic, and Tadayoshi Kohno. Epc rfid tag security weaknesses and defenses: passport cards, enhanced drivers licenses, and beyond. In *ACM Conference on Computer and Communications Security*, pages 33–42, 2009.

[21] PKI Task Force. PKI for machine readable travel documents offering icc read-only access. Technical report, International Civil Aviation Organization, 2004.

[22] Steve Schneider and Abraham Sidiropoulos. Csp and anonymity. In *In European Symposium on Research in Computer Security*, pages 198–218. Springer-Verlag, 1996.

[23] Serge Vaudenay. On privacy models for RFID. In *Advances in Cryptology ASIACRYPT 2007*, pages 68–87, 2007.

[24] Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels. Security and privacy aspects of low-cost radio. In *Hutter, D., Müller, G., Stephan, W., Ullman, M., eds.: International Conference on Security in Pervasive Computing - SPC 2003, volume 2802 of LNCS, Boppard, Germany*, pages 454–469. Springer-Verlag, march 2003.