Formal verification of privacy for RFID systems

Mayla Brusó, Konstantinos Chatzikokolakis, and Jerry den Hartog

Eindhoven University of Technology

Abstract. RFID tags are being widely employed in a variety of applications, ranging from barcode replacement to electronic passports. Their extensive use, however, in combination with their wireless nature, introduces privacy concerns as a tag could leak information about the owner's behaviour. In this paper we define two privacy notions, untraceability and forward privacy, using a formal model based on the applied pi calculus, and we show the relationship between them. Then we focus on a generic class of simple privacy protocols, giving sufficient and necessary conditions for untraceability and forward privacy for this class. These conditions are based on the concept of frame independence that we develop in this paper. Finally, we apply our techniques to two identification protocols, formally proving their privacy guarantees.

1 Introduction

Radio Frequency Identification (RFID) systems are wireless technology for automatic identification consisting of a set of tags, readers and a backend. The tags are typically very simple devices consisting of a tiny chip and an antenna thus offering very limited resources. The readers are connected with the backend which stores the valuable information about the tags. The tags interact with the readers through identification protocols which aim to get the identity of the tag to the backend system in a secure manner.

The wireless nature of RFID makes access to tags extremely easy. They are commonly used, for example, in supply chain management and are starting to make their way into the consumer realm. One of the main issues that needs to be addressed to make this possible it that of privacy: the fact that access to the tags is so easy also introduces the potential of misuse. The tag's ease of access allows them to be easily analyzed by the attacker. Also, as the tag may travel with its owner, its location is already sensitive information, thus an attack which does not identify a tag but does distinguish it from other tags is already a problem. Finally, the resource constraints of the tags mean that many security protocols cannot be used as the tag is not able to perform the required cryptographic operations.

The problems identified lead to security goals of *untraceability* and *forward privacy* for identification protocols. Untraceability states that an attacker is not able to trace the movement of a tag, i.e. observing past events should not allow an attacker to distinguish between tags. The stronger goal of forward privacy in turn becomes important when the attacker may obtain the tag in question e.g. by stealing it or even simply buying the item it is attached to. As the tags are simple devices, the attacker can likely break the tag to obtain any information stored in it. Still, this should not enable the attacker to trace the tag in retrospect, i.e. to learn its past locations.

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

Fig. 1. Syntax of the applied pi calculus

Because of the resource constraints of the tags, hash functions, which are arguably easy to compute, often play an important role in the identification protocols. Consider, for example, the simple but effective OSK protocol ([1]): It assumes the tag can perform two distinct one-way functions $\mathbf{h}$ and $\mathbf{g}$. Each tag stores a secret which it shares with the backend and 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 $s_{i+1} = \mathbf{h}(s_i)$. Intuitively this protocol meets our security goals; the function $\mathbf{g}$ ensures that the output of the tag is random thus untraceable, and updating the secret with $\mathbf{h}$ ensures that no past secrets or interactions can be found if the tag is broken and secret s_{i+1} is obtained by the attacker. But how can we formally check this?

In this paper we introduce a formal model for RFID privacy, expressing untraceability and forward privacy as equivalences in the applied pi calculus, and we show that forward privacy is stronger. We then study a generic class of single step protocols, giving necessary and sufficient conditions for both properties. These conditions are based on the notion of frame independence that we develop in Section 4. These results are then employed to prove the privacy properties of the OSK protocol mentioned above, as well as another protocol from the literature. We also show how alterations to this protocol cause flaws, breaking forward privacy or even untraceability. We conclude with related and future work. The proof of all results is given in the appendix.

2 The applied pi calculus

The applied pi calculus ([2]) is a language for describing concurrent processes and their interaction. It extends the pi calculus ([3]) adding the possibility to model cryptographic primitives through a signature and an equational theory. In this section we briefly recall its basic notions, for an extended description see [2].

The syntax consists of terms, plain processes and extended processes. A term is a name $(a, b, c, \ldots)$, a variable (x, y, z) or a function application $\mathbf{f}(M_1, \ldots, M_l)$, where $\mathbf{f}$ is a function symbol from a signature Σ, $M_1, \ldots, M_l$ are terms and l is the arity of $\mathbf{f}$. 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 $M \trianglelefteq N$ iff M is a subterm of N and $M \triangleleft N$ iff $M \trianglelefteq N$ and $M \neq N$. We also denote by $M[^N/_x]$ the term obtained by substituting N for x in M.

The syntax of processes is shown in Fig. 1. We have the standard primitives from the π-calculus together with if-then-else and the possibility to output terms, instead of simple names. Processes are extended with active substitutions which replace variables with terms, modelling information known to the environment. We use $fn(P), fv(P)$ to denote the free names and variables of P, with restriction and input being considered binders. An extended process is closed when its variables are bound or defined by an active substitution. Finally, frames ranged over by ψ and φ, are extended processes with no plain sub-process. A frame is in *canonical form* iff $\varphi = \nu\widetilde{n}.\{\widetilde{^M/_x}\}$ where $fv(\widetilde{M}) = 0$ and $\{\widetilde{n}\} \subseteq fn(\widetilde{M})$. The set $\{\widetilde{x}\}$ is called the domain of φ, written $dom(\varphi)$.

The signature Σ is equipped with an equational theory, i.e. an equivalence relation $=_{\mathrm{E}}$ on terms that is closed under substitution of terms for variables. We also require $=_{\mathrm{E}}$ to be closed under one-to-one substitution of names. We write $M =\!\lhd N$ iff $\exists M' : M =_{\mathrm{E}} M' \lhd N$ and $M =\!\unlhd N$ iff $\exists M' : M =_{\mathrm{E}} M' \unlhd N$. The signature together with the equational theory are used to model cryptographic primitives. For example, symmetric encryption can be modelled using two function symbols $\mathbf{enc}, \mathbf{dec}$ together with the equation $\mathbf{dec}(\mathbf{enc}(x, k), k) =_{\mathrm{E}} x$. One-way hash functions can be modelled using a unary function symbol $\mathbf{h}$ with no equations.

The operational semantics is defined by two relations: structural equivalence and internal reduction. An evaluation context is an extended process with a hole replacing an extended sub-process. Structural equivalence $\equiv$ is the smallest equivalence relation on extended processes that is closed by α-conversion on both names and variables, by application of evaluation contexts, and satisfying some structural rules such as associativity and commutativity of $\mid$, binding-operator-like behaviour of ν, and:

$$A \mid 0 \equiv A \quad \nu x.\{^M/_x\} \equiv 0 \quad \{^M/_x\} \mid A \equiv \{^M/_x\} \mid A[^M/_x] \quad \{^M/_x\} \equiv \{^N/_x\}$$

assuming $M =_{\mathrm{E}} N$. It can be shown that any frame φ is structurally equivalent to a frame φ' in canonical form. Internal reduction $\to$ is the smallest relation on extended processes closed by structural equivalence and application of evaluation contexts s.t.:

$$\begin{array}{rl} \bar{a}\langle x\rangle.P \mid a(x).Q \to P \mid Q & \\ \text{if } M = M \text{ then } P \text{ else } Q \to P & \\ \text{if } M = N \text{ then } P \text{ else } Q \to Q & \qquad \text{for any ground terms } M \text{ and } N \text{ s.t. } M \neq_{\mathrm{E}} N \end{array}$$

Several properties of security protocols can be formalised in terms of observational equivalence between processes. We write $A \Downarrow a$ when A can send a message on a channel a, that is, when $A \to^* C[\bar{a}\langle M\rangle.P]$ for some evaluation context C that does not bind a.

Definition 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 $A \to^* A'$ then $B \to^* B'$ and $A'\mathcal{R}B'$ for some B'.*
3. *$C[A]\mathcal{R}C[B]$ for closing evaluation contexts C.*

In [2], a labeled bisimilarity $\approx_l$ is defined and it is proven that $\approx = \approx_l$. Labeled bisimilarity is useful in proofs, since it is much easier to handle that observational equivalence. Finally, static equivalence between frames is defined below.

Definition 2. *Two terms M and N are* equal in the frame φ, *written* $(M = N)\varphi$, *iff* $\varphi \equiv \nu\widetilde{n}.\sigma$, $M\sigma = N\sigma$, *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* $dom(\varphi) = dom(\psi)$ *and when, for all terms* M *and* N, *we have* $(M = N)\varphi$ *iff* $(M = N)\psi$.

3 Formalization of RFID protocols and properties

In this section we discuss how to formalize RFID protocols and their privacy properties in the applied pi calculus. We focus on two privacy properties, *untraceability* (also called *indistinguishability*, *unlinkability* or simply *privacy*, in various contexts) and *forward privacy*. Roughly speaking, a protocol satisfies untraceability if the adversary cannot link two sessions of the protocol to the same tag. For forward privacy, the attacker is allowed to tamper a tag and retrieve the data stored in it. Forward privacy is satisfied if the attacker cannot use the data to link the tag to past sessions, obtained before tampering it.

Untraceability games. Privacy properties are typically defined by means of games in a computational setting. Two similar types of games can be found in the literature. In the first one, used in [4, 1, 5, 6], the untraceability game consists roughly of the following phases: in the beginning the adversary can eavesdrop communications and query all the tags and the readers in the system. Then a tag is chosen randomly from the set $\{T_0, T_1\}$ and the attacker is given access to it. She can then query $\{T_0, T_1\}$ as well as all other tags in the system. The game ends with the adversary announcing her guess of the selected tag. The protocol satisfies untraceability if the adversary cannot detect the selected tag with probability higher than random guessing.

A slightly different idea is used in the definition of unlinkability in [7] as well as the one of privacy in [8]. In this game, the attacker is given access to two tags which can be either independent or linked, meaning that by querying any of them the adversary actually interacts with the same tag. She is then allowed to query these tags as well as all the other tags and readers in the system. Untraceability is satisfied if the adversary cannot distinguish the two cases with probability higher than random guessing.

Our definition, given in Section 3.2, is inspired by the second approach. In our model, tags communicate with the environment through a *tag interface*. A protocol satisfies untraceability if, given two interfaces that she can freely query, the attacker cannot distinguish whether they correspond to the same or different tags. Similar games can be also defined for forward privacy.

3.1 Modelling RFID protocols in the applied pi calculus

The applied pi calculus provides an elegant framework for modelling security protocols, as it allows to specify both the interaction between the various agents, using the communication primitives of the calculus, as well as the cryptographic operations, using a suitable equational theory. In this section we discuss the characteristic features of RFID protocols and how to model them in the applied pi calculus.

An RFID system typically consists of several tags and one or more readers. The readers might also communicate with a centralized backend database, typically through a secure channel. An important property of most RFID protocols is that they are *stateful*, usually as a means to provide the same functionality to a stateless protocol but with simpler cryptographic primitives. Tags have an internal memory which is often updated after each execution of the protocol. Thus, in contrast to traditional stateless protocols, a tag can send different data on each execution, even without the use of randomly generated nonces. All tags execute the same code and they are differentiated only from their state, that is the content of their memory. Each tag typically has a unique secret s, also known to the reader (through the backend database), which distinguishes it from other tags and which allows the reader to identify it. The secret is generated either during the tag's creation or some initialization phase, and it is stored in the tag's state. After that, the secret is never transmitted by the tag in cleartext.

We model the tag's state using a process running in parallel with the tag's main code and which can output the state's content in a channel local to the tag. For clarity, we use the process $St(w, M) \triangleq \overline{w}\langle M \rangle$ to denote the state, where w is the restricted channel used to read the state and M is the state's content. Then, the tag reads the state using an input $w(x)$, and updates the state by putting back a $St(w, M')$ subprocess at the end of its execution. The tag uses also a public channel c to communicate with the reader. We use $P(w, c)$ to denote the process modelling a single session of the tag, with state channel w and public channel c.

Finally, cryptographic primitives are modelled, as usual, using a signature and an equational theory. The protocols analyzed in this paper use only one-way hash functions. These are modelled using a unary function symbol $\mathbf{h}$ with no equations. As a consequence, $\mathbf{h}$ is one-way (cannot be inverted) and collision free. We discuss hash functions in more detail in Section 4.1. Other cryptographic primitives will require proper equations, for example $\mathbf{dec}(\mathbf{enc}(x, k), k) = x$ for symmetric encryption. Several other cryptographic primitives are discussed in [2].

Example 1. As an example, we model the OSK protocol described in the introduction. The protocol uses two hash functions, $\mathbf{h}$ and $\mathbf{g}$, to update the secret and encrypt it respectively. A tag sessoin in this protocol can be modelled as follows:

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

We use $c(\_)$ to denote an input on channel c with a variable not used anywhere in the process. This input on c simply triggers the execution of the protocol and corresponds to the reader asking "Who are you?". Then the tag reads the current content of its state using an input on its private channel w, and outputs the hash $\mathbf{g}$ of the current state on the public channel c. Finally, it updates the state by continuing as $St(w, \mathbf{h}(x))$, which will be read at the next execution of the protocol. Note that this is the encoding of a single tag session. The complete system will include other tags, the initial state of each tag, the reader, etc., as discussed in the next section.

3.2 Defining untraceability and forward privacy

The idea behind our definition is inspired by the second type of untraceability game discussed in Section 3. In our model, the attacker communicates with a tag through a *tag interface*. Typically, this corresponds to the attacker obtaining proximity to the tag and querying it wirelessly, but in general it can refer to any kind of access to the tag that the attacker might obtain. An interface can be queried an arbitrary number of times and each time the same tag is accessed. For example, an attacker can query a tag at the entrance of a building multiple times, always interacting with the same tag. On the other hand, multiple interfaces might provide access to the same tag. For example, an attacker might see a tag at the entrance of building A and a tag at the entrance of building B. This gives him two interfaces to communicate with a tag, however it could be either the same tag in both cases or different ones. In other words, the attacker does not know which physical tag she is accessing each time.

As discussed in the previous section, we denote by $P(w, c)$ the process modelling a single tag session. The channel c is a public channel that the tag uses to communicate with the environment, in other words the tag's interface. Since all tags execute the same code, they are distinguished solely by their state. Consider $P(w, c_1)$ and $P(w, c_2)$. If w is connected to the same state in both processes then we have two tag interfaces linked to the same tag, otherwise they are two separate tags.

Let $InitSt(w, s)$ be a process that initializes the tag's 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 and then stores s in the state. We define:

$$\begin{aligned} Tag(c) &\triangleq \nu w.\nu s.\big(!P(w, c) \mid InitSt(w, s)\big) \\ Tag(c_1, c_2) &\triangleq \nu w.\nu s.\big(!P(w, c_1) \mid !P(w, c_2) \mid InitSt(w, s)\big) \\ ReplTag &\triangleq !\, \nu c.\overline{an}\langle c\rangle.Tag(c) \end{aligned}$$

$Tag(c)$ models a complete tag with interface c. It can perform an unbounded number of protocol executions, starting from the initial state $InitSt(w, s)$. Then $Tag(c_1, c_2)$ models a single tag (it contains a single state) but with two interfaces c_1, c_2. Finally $ReplTag$ models an unbounded number of tags, each with its own interface. To achieve this, a new channel c is created by each replicated copy, then c is announced on the public channel an to make it available to the outside environment. Let also $Reader, DB$ be processes modelling the reader and the backend database, and let $\widetilde{n}$ contain any private channels shared between all parties (eg. used to register a tag). We are now ready to state our definition of untraceability.

Definition 3 (Untraceability). *A protocol satisfies untraceability iff*

$$\begin{gathered} \nu\widetilde{n}.(Tag(c_1, c_2) \mid ReplTag \mid Reader \mid DB) \approx \\ \nu\widetilde{n}.(Tag(c_1) \mid Tag(c_2) \mid ReplTag \mid Reader \mid DB) \end{gathered}$$

Intuitively, this definition requires that the attacker cannot distinguish whether two interfaces correspond to the same tag or two different tags. Note that the adversary is not modelled explicitly, but she is considered part of the environment. Observational equivalence guarantees that no environment will be able to distinguish these two cases.

Note that, because of $ReplTag$, the definition involves an unbounded number of tags and interfaces. However, only two interfaces are linked to the same tag, all others provide access to different tags. This is similar to the untraceability games in which the attacker is provided with two tags to distinguish, even though there are arbitrarily many tags in the system. A slightly different approach would be to link more interfaces together, for example we could have a single tag in the left-hand side with an unbounded number of interfaces. Studying this variation is left as future work.

Synchronization issues. Tags can only run one protocol session at a time. This is due to the fact that the state needs to be updated before starting a new session. However, for protocols with multiple steps, this can lead to a violation of untraceability. Consider the following scenario: an attacker starts communicating with a tag using the interface c_1 (eg. at location A). In the middle of the session she stops, leaving the tag in an intermediate state. Later she accesses a tag using a different interface c_2 (eg. 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 since the tag is in the middle of the 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 or not, violating untraceability.

This type of attacks is captured by our definition of untraceability. $Tag(c_1) \mid Tag(c_2)$ 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. In practise, however, this type of attacks might be prevented by some property of the tag that we do not want to 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. Similarly, the tag might be programmed to run each session for a small amount of time, and then switch off automatically. In such cases, we would like to restrict our attacker model to enforce that a session on the c_1 interface needs to finish before a session on c_2 can start, and vice versa. This can be easily achieved using a token t that is consumed by $Tag(c_1), Tag(c_2)$ in the beginning of an execution, and put back at the end. Thus t acts like a semaphore preventing simultaneous executions. Note that only c_1, c_2 need to be synchronized, the rest of the tags can remain unaltered. We use this technique in Section 5.

Forward privacy. The forward privacy property is modelled in the same way as untraceability, but the adversary is given a further ability: she is now able to break one of the two tags he is given and retrieve the information stored in its state. After that, the tag clearly becomes traceable. However, forward privacy requires that the attacker is still unable to trace protocol sessions that happened before breaking the tag. To capture this notion, once the tag is broken the interfaces c_1, c_2 cannot be used any longer. Thus the attacker can only use information obtained in past sessions to distinguish the two cases. She can still, however, communicate with all the other tags of the system. We define:

$$\begin{aligned} BrTag(c_1, c_2) &\triangleq \nu w.\nu s.\big(!P(w, c_1) \mid !P(w, c_2) \mid InitSt(w, s) \mid br(\_).w(x).\overline{br}\langle x\rangle\big) \\ Tag(w, c) &\triangleq \nu s.\big(!P(w, c) \mid InitSt(w, s)\big) \\ TwoTags(c_1, c_2) &\triangleq \nu w_1.\nu w_2.\big(Tag(w_1, c_1) \mid Tag(w_2, c_2) \mid br(\_).w_1(x).w_2(\_).\overline{br}\langle x\rangle\big) \end{aligned}$$

$BrTag(c_1, c_2)$ models a breakable tag with two interfaces. It is similar to $Tag(c_1, c_2)$ with the addition of the br action. Once triggered, the content of the state is read and sent back to the attacker on the public channel br. Note that reading the state ensures that any active session is finished. The state is not replaced, rendering the interfaces c_1, c_2 unusable since any query on them will lead to an attempt to read the state and the process will be blocked. $Tag(w, c)$ is the same as $Tag(c)$ with the exception that w is not restricted so that it can be used from an outer process. Finally, $TwoTags(c_1, c_2)$ models two independent breakable tags. It is similar to $Tag(c_1) \mid Tag(c_2)$ with the addition of the br action. Once triggered, the content of both states is read. This ensures that both tags have finished any active session. The state of the first tag is then sent to the attacker and the second is discarded. The states are not restored thus deactivating both c_1, c_2. We can now state the definition of forward privacy.

Definition 4 (Forward Privacy). *A protocol satisfies forward privacy iff*

$$\nu\widetilde{n}.(BrTag(c_1, c_2) \mid ReplTag \mid Reader \mid DB) \approx$$
$$\nu\widetilde{n}.(TwoTags(c_1, c_2) \mid ReplTag \mid Reader \mid DB)$$

The definition is similar to the one of untraceability: an adversary should not be able to distinguish a tag with two interfaces from two separate tags. The difference is the possibility to break one of the tags and read its state, but without querying the two tags any longer. It is clear that this extra ability makes this definition stronger.

Proposition 1. *Forward privacy (Def 4) implies untraceability (Def 3).*

4 Frame independence

In this section we discuss a notion that we call *frame independence*. As shown in Section 5, this concept can be used to give sufficient and necessary conditions for untraceability and forward privacy for a generic family of protocols. Nevertheless, the notion itself is generic, hence we develop it on its own, proving some results that will be used later in the paper.

Consider two frames φ_1, φ_2, each containing some free names. We provide both frames to the attacker, after restricting these names. The attacker's goal is to decide whether the terms in both frames contain the same restricted names $\widetilde{s}$, or different. If the attacker is able to distinguish the two cases we say that φ_1, φ_2 are *dependent* wrt $\widetilde{s}$, 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 $\widetilde{s}$. We formalize this idea in the following definition.

Definition 5. *Let φ_1 and φ_2 be closed frames with $dom(\varphi_1) \cap dom(\varphi_2) = \emptyset$. We say that φ_1 is independent of φ_2 with respect to the names $\widetilde{s}$, written $\varphi_1 \perp_{\widetilde{s}} \varphi_2$, iff $\nu\widetilde{s}.(\varphi_1 \mid \varphi_2) \approx_{\mathrm{s}} \nu\widetilde{s}.\varphi_1 \mid \nu\widetilde{s}.\varphi_2$.*

Intuitively, this definition states that φ_1, φ_2 are independent wrt to $\widetilde{s}$ iff their composition under the same restricted names $\widetilde{s}$ is statically equivalent to simply putting them in parallel, each with their own restricted names. The 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 with shared names) is obtained by simply multiplying the marginal distributions (in our case putting in parallel the two frames).

We now state some basic properties of frame independence.

Proposition 2. *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* $\{\widetilde{s}\} \cap fn(\varphi_1') = \emptyset$ *and* $dom(\varphi_1') \cap dom(\psi) = \emptyset$
3. $\varphi_2 \equiv \nu u.\varphi_1$ *for some* $u \notin fv(\psi) \cup fn(\psi)$

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

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 term already present in φ_1. For example, $\varphi_2 = \nu n.\{^{f(n)}/_x, ^{g(f(n))}/_y\}$ is an extension of $\varphi_1 = \nu n.\{^{f(n)}/_x\}$ since $\varphi_2 \equiv \varphi_1 \mid \{^{g(x)}/_y\}$. 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.

4.1 Independence of hash functions

One-way hash functions are commonly used in RFID protocols. Indeed, both protocols analyzed in this paper use solely hash functions as cryptographic primitive. In this section we give some results concerning the independence of frames using hash functions.

In the applied pi calculus, hash functions are typically modelled by a unary function symbol $\mathbf{h}$ with no equational axioms. Still, hash functions can be combined with other cryptographic primitives with their own axioms, so we might end up with an equational theory with an arbitrary set of axioms, the only condition being that they should not contain $\mathbf{h}$. To use this fact in proofs, we should find properties of hash functions that hold under any such theory. In fact, 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 such a definition which will then be used in proofs involving hash functions.

We fix an equational theory $=_{\mathrm{E}}$, let M, K, L be terms and let $\mathbf{h}$ a unary function. We define $M[^L/_{\mathbf{h}(=K)}]$ as the substitution of $\mathbf{h}$-terms equal to $\mathbf{h}(K)$ by L. More precisely:

$$M[^L/_{\mathbf{h}(=K)}] = \begin{cases} M & \text{if } M = x \text{ or } M = n \\ \mathbf{f}(M_1[^L/_{\mathbf{h}(=K)}], \ldots, M_l[^L/_{\mathbf{h}(=K)}]) & \text{if } M = \mathbf{f}(M_1, \ldots, M_l), \mathbf{f} \neq \mathbf{h} \\ \mathbf{h}(M_1[^L/_{\mathbf{h}(=K)}]) & \text{if } M = \mathbf{h}(M_1), M_1 \neq_{\mathrm{E}} K \\ L & \text{if } M = \mathbf{h}(M_1), M_1 =_{\mathrm{E}} K \end{cases}$$

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

Definition 6. *We say that a unary function* $\mathbf{h}$ *is a* one-way hash function *wrt* $=_{\mathrm{E}}$ *iff*

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

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

A standard way to construct such an equational theory is using a finite set of axioms that do not contain $\mathbf{h}$. The idea of this definition is that $\mathbf{h}$ could appear in an equation $K =_{\mathrm{E}} L$ but only as a "generic term", the equation should not depend on the fact that $\mathbf{h}(M)$ is a hashed value. The following lemma shows that hash functions behave as expected.

Lemma 1. *Let* $\mathbf{h}$ *be a hash function (Def. 6) 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 equation that inverts* $\mathbf{h}$*, i.e.* $\mathbf{invh}(\mathbf{h}(x)) =_{\mathrm{E}} x$
4. *there is no equation that checks a hashed value, i.e.* $\mathbf{checkh}(M) =_{\mathrm{E}} \mathbf{ok}$ *iff* $M =_{\mathrm{E}} \mathbf{h}(M')$.

We are now ready to give a generic result, showing a sufficient condition for the independence of frames containing hashed terms.

Theorem 1. *Let* $\mathbf{h}_1, \ldots, \mathbf{h}_k, \mathbf{g}_1, \ldots, \mathbf{g}_l$ *be hash functions (Def. 6), not necessarily distinct, and let*

$$\varphi_1 = \{{}^{\mathbf{h}_1(S_1)}/_{x_1}, \ldots, {}^{\mathbf{h}_k(S_k)}/_{x_k}\} \qquad \varphi_2 = \{{}^{\mathbf{g}_1(T_1)}/_{y_1}, \ldots, {}^{\mathbf{g}_l(T_l)}/_{y_l}\}$$

be frames in canonical form. Assume that $\mathbf{h}_i(S_i) \not=\!\!\!\!\not\trianglelefteq \mathbf{g}_j(T_j)$ *and* $\mathbf{g}_j(T_j) \not=\!\!\!\!\not\trianglelefteq \mathbf{h}_i(S_i)$ *for all* $1 \leq i \leq l, 1 \leq j \leq m$*. Then* $\varphi_1 \perp_{\widetilde{s}} \varphi_2$ *for all names* $\widetilde{s}$.

5 Analysis of a generic class of protocols

In this section we focus on a class of protocols that we call "single step" identification protocols. The main characteristic of this class is that each protocol session contains a single message sent from the tag to the reader. The tag is first activated by the reader, without however receiving any information. Then, the tag reads its state, constructs a proper message, possibly containing fresh nonces, and sends it to the reader. This message alone should be sufficient for the reader to identify the tag. Finally, the tag updates its state and the session ends. The simplicity of such protocols will help us understand the fundamental properties needed to satisfy untraceability and forward privacy. Still, as we will see in the next section, two published protocols fall in this class.

We first introduce some notation to simplify the presentation. Let $\pi(x)$ denote a term containing a single free variable x (possibly with multiple occurrences). We define $\pi(M) = \pi(x)[{}^{M}/_{x}]$ which allows us to use function notation, for example $\pi(\pi(M)) = \pi(x)[{}^{\pi(x)[{}^{M}/_{x}]}/_{x}]$. We also write $\pi^n(M)$ for $\pi(\ldots\pi(M))$, n times. To define our class of protocols, we instantiate the Tag processes of Section 3.2, which corresponds to instantiating $P(w, c)$ and $InitSt(w, s)$.

Definition 7. *The class of single step protocols consists of all protocols of the form:*

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

for some terms $\pi(x), \sigma(x), S_0$ *and channels* $\widetilde{\rho}$ *s.t.* $s \notin fn(\pi(x)) \cup fn(\sigma(x))$.

The term $\pi(x)$ is the output of a tag when x is its current state, and it can contain the restricted names $\widetilde{\rho}$ (this corresponds to generating fresh nonces). Similarly, $\sigma(x)$ is the new state of the tag after the execution. For simplicity, we assume that $\{\widetilde{\rho}\} \cap fn(\sigma(x)) = \emptyset$, i.e. fresh nonces are only transmitted, not stored in the state. Finally, S_0 is the initial content of the state, and it could contain the name s, which is the tag's secret. Note that any signature with any equational theory can be used for these terms.

$InitSt(w, s)$ simply initializes the state with S_0. $P(w, c)$ starts with an input on c, which simply triggers the beginning of the session. Then, we use the token technique described in Section 3.2 for synchronization. This aims at simplifying the proofs, even though it is not strictly needed for any of the results in this section. The tag consumes the token t and reads its state in x. Finally it outputs $\pi(x)$ and updates the state with $\sigma(x)$. 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. So, to complete the instantiation of all processes of Def. 3, we set $Reader = DB = 0$ and $\widetilde{n} = \varepsilon$.

Untraceability. Clearly not all single step protocols satisfy untraceability. We start by identifying the possible reasons for violating it. The simplest case is when the i-th and j-th sessions of a tag can be distinguished. Note that a tag outputs $\pi(\sigma^i(S_0))$ on it's $(i{+}1)$-th session. Consider the extreme case where $\pi(\sigma^i(S_0)) = i$ (eg. let $S_0 = 0, \sigma(x) = x + 1, \pi(x) = x$). This gives the information to the attacker of how many sessions the protocol has run so far. Now the attacker can simply run a session on c_1 followed by a session on c_2. If the interfaces correspond to the same tag, the second session will output 2, otherwise it will output 1, allowing the attacker to easily distinguish the two cases.

To simplify the notation we define $\rho_x M \triangleq \nu\widetilde{\rho}.\{^M/_x\}$, that is ρ_x turns a term into a frame. We then define the following property.

Definition 8. *A single-step protocol satisfies* $\mathcal{P}_1$ *iff*

$$\nu s.\rho_x\pi(\sigma^n(s)) \approx_s \nu s.\rho_x\pi(\sigma^m(s)) \quad \forall n, m \in \mathbb{N}$$

Intuitively, $\mathcal{P}_1$ requires that the tag's output on different sessions is indistinguishable. This prevents the simple attack discussed above but is still not sufficient for untraceability. Consider another extreme case where $\pi(\sigma^i(S_0)) = s$ (eg. $S_0 = s, \pi(x) = \sigma(x) = x$). This satisfies $\mathcal{P}_1$ since the output does not depend on i. However untraceability is clearly violated since the tag's secret is sent in cleartext. Running two sessions on c_1, c_2 the attacker will get s_1, s_2 if the interfaces correspond to different tags, otherwise s, s. Protecting the secret with a hash, i.e. $\pi(\sigma^i(S_0)) = \mathbf{h}(s)$ does not help either. Running two sessions on c_1, c_2 will give $\mathbf{h}(s_1), \mathbf{h}(s_2)$ in the first case and $\mathbf{h}(s), \mathbf{h}(s)$ in the second, which can be also distinguished. Indeed, it is clear that if the output on every session is constant, untraceability will always be violated. But even a variable output is no guarantee: consider a single step protocol with $\pi(\sigma^i(S_0)) = \mathbf{h}^{i+1}(s)$. Running two sessions on c_1, c_2 will give $x_1 = \mathbf{h}(s_1), x_2 = \mathbf{h}(s_2)$ in the case of two independent tags and $x_1 = \mathbf{h}(s), x_2 = \mathbf{h}^2(s)$ in the case of a single tag. By checking $\mathbf{h}(x_1) = x_2$ the attacker can distinguish once again the two cases.

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 exactly in the notion of frame independence, which brings us to the definition of the property $\mathcal{P}_2$.

Definition 9. *A single-step protocol satisfies $\mathcal{P}_2$ iff*

$$\prod_{i=0}^{n-1} \rho_{x_i}\pi(\sigma^i(s)) \perp_s \rho_{x_n}\pi(\sigma^n(s)) \quad \forall n \in \mathbb{N}$$

Intuitively $\mathcal{P}_2$ requires that the tag's output in the first n sessions is independent from the output of the $n+1$-th session, wrt to the tag's secret s.

Note that $\mathcal{P}_1$ and $\mathcal{P}_2$ are incomparable: the first extreme case, $\pi(\sigma^i(S_0)) = i$, satisfies $\mathcal{P}_2$ but not $\mathcal{P}_1$ while the second extreme case $\pi(\sigma^i(S_0)) = s$ satisfies $\mathcal{P}_1$ but not $\mathcal{P}_2$. There are two inherently different flaws of $\pi(\sigma^i(S_0))$ that the attacker can exploit: a dependency on i and a dependency on s. $\mathcal{P}_1$ disallows the first while $\mathcal{P}_2$ disallows the second. Together they capture untraceability for single step protocols.

Theorem 2. *A single step protocol satisfies untraceability iff it satisfies $\mathcal{P}_1$ and $\mathcal{P}_2$.*

The complete proof is given in the appendix. The main part is to show that $\mathcal{P}_1, \mathcal{P}_2$ are sufficient for untraceability, we only sketch the main idea here. Note that, since the reader and the backend database are not modelled explicitly, Def 3 is greatly simplified. It is sufficient to show that $Tag(c_1, c_2) \approx Tag(c_1) \mid Tag(c_2)$, as we can add $ReplTag$ using the congruence of $\approx$. The dynamics of these processes is simple and both are able to perform the same transitions. The challenging part of the proof is to show that the produced frames are statically equivalent. Assume that n sessions are run on c_1 and m sessions on c_2. Then $Tag(c_1, c_2)$ will produce

$$\nu s.\left(\prod_{i=0}^{n+m-1} \rho_{x_i}\pi(\sigma^i(S_0))\right)$$

since both interfaces are connected to the same tag. Using $\mathcal{P}_2$ we can show that this is statically equivalent to

$$\prod_{i=0}^{n+m-1} \nu s.\rho_{x_i}\pi(\sigma^i(S_0))$$

that is, to the same output performed by $n+m$ separate tags. Now we can use $\mathcal{P}_1$ to freely change the exponents of σ, and we get

$$\prod_{i=0}^{n-1} \nu s.\rho_{x_i}\pi(\sigma^i(S_0)) \mid \prod_{j=0}^{m-1} \nu s.\rho_{x_{n+j}}\pi(\sigma^j(S_0))$$

Finally, we can use $\mathcal{P}_2$ again to "join" the tags, and finally obtain:

$$\nu s.\left(\prod_{i=0}^{n-1} \rho_{x_i}\pi(\sigma^i(S_0))\right) \mid \nu s.\left(\prod_{j=0}^{m-1} \rho_{x_{n+j}}\pi(\sigma^j(S_0))\right)$$

which is exactly the frame produced by $Tag(c_1) \mid Tag(c_2)$, consisting of n outputs of $Tag(c_1)$ and m outputs of $Tag(c_2)$.

Forward privacy. For forward privacy we need to strengthen our conditions, since the adversary now has an extra capability, namely to reveal the state of a tag. The attacker might try to link the state to the output of another tag, so we have to ensure that the state is independent from all previous output. This brings us to the property $\mathcal{P}_3$.

Definition 10. *A single step protocol satisfies $\mathcal{P}_3$ iff*

$$\prod_{i=0}^{n-1} \rho_x \pi(\sigma^i(s)) \perp_s \{{}^{\sigma^n(s)}/_y\} \quad \forall n \in \mathbb{N}$$

$\mathcal{P}_3$ is similar to $\mathcal{P}_2$, but instead of requiring that the $(n+1)$-th output is independent from the first n, it requires that the contents of the state after the n-th session is independent from the first n outputs. In fact, this is strictly stronger.

Proposition 3. *For all single step protocols, $\mathcal{P}_3 \Rightarrow \mathcal{P}_2$.*

We can now state the corresponding result for forward privacy.

Theorem 3. *A single step protocol satisfies forward privacy iff it satisfies $\mathcal{P}_1$ and $\mathcal{P}_3$.*

The proof is similar to the one for untraceability. Note that the above theorem together with Prop. 3 shows that forward privacy implies untraceability for single step protocols, which was already expected from Prop. 1.

6 Case studies

In this section we apply the results for single step protocols to two existing ones from the literature. The first is the OSK protocol ([1]), already discussed in the introduction and formalized in Section 3.1. We also discuss some variations of the protocol, where we weaken some aspects of the protocol to examine how privacy is affected. Finally we analyze a basic hash protocol of [9], which falls in the same class even though it is quite different in spirit that the OSK protocol.

6.1 The OSK protocol

In the OSK protocol [1], tags can compute two distinct hash functions $\mathbf{g}, \mathbf{h}$. The state of each tag is initialized with a randomly generated secret which is also known to the backend. On 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. As a consequence, 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 tags, so it can compute $\mathbf{g}(\mathbf{h}^{i-1}(s))$ for all secrets and thus identify the tag. For efficiency, the backend can precompute the expected output for the next run of all tags, and perform a fast search during identification. Once the tag is identified, its expected output can be updated.

The OSK protocol can be modelled as a single-step protocol (Def. 7) with:

$$S_0 = s \qquad \pi(x) = \mathbf{g}(x) \qquad \sigma(x) = \mathbf{h}(x) \qquad \widetilde{\rho} = \varepsilon$$

Thus, proving forward privacy for OSK reduces to proving the properties $\mathcal{P}_1, \mathcal{P}_3$.

Proposition 4. *The OSK protocol satisfies properties $\mathcal{P}_1, \mathcal{P}_3$, namely:*

$$
\begin{array}{lll}
\mathcal{P}_1 & \nu s.\{^{\mathbf{g}(\mathbf{h}^n(s))}/_x\} \approx_s \nu s.\{^{\mathbf{g}(\mathbf{h}^m(s))}/_x\} & \forall n, m \in \mathbb{N} \\
\mathcal{P}_3 & \prod_{i=0}^{n-1}\{^{\mathbf{g}(\mathbf{h}^i(s))}/_{x_i}\} \perp_s \{^{\mathbf{h}^n(s)}/_y\} & \forall n \in \mathbb{N}
\end{array}
$$

The challenging proof is the one of $\mathcal{P}_3$ which follows from Theorem 1, since for all $i < n$ no subterm of $\mathbf{g}(\mathbf{h}^i(s))$ is equal to $\mathbf{h}^n(s)$ and vice versa. Then by Theorem 3 we conclude that OSK satisfies forward privacy (and as a consequence also untraceability).

Note that proving $\mathcal{P}_1, \mathcal{P}_3$ involves proving an infinite number of static equivalences. However, each one of them can be proven automatically using the ProVerif tool ([10]). Proving these equivalences up to a fixed n corresponds to proving forward privacy up to a fixed number of tag sessions. We used ProVerif successfully to prove the above equivalences for up to 1000 sessions, which only took a few minutes. On the other hand, even though ProVerif is capable of automatically proving observational equivalence in some cases ([11]), it was unable to directly prove forward privacy using the Def. 4.

Weak OSK protocol. We might ask the question of whether both hash functions of OSK are needed. We examine here the effects of relaxing the conditions on $\mathbf{h}, \mathbf{g}$. First consider the case where h is not one-way, that is there exists a function $\mathbf{invh}$ and an equation $\mathbf{invh}(\mathbf{h}(x)) =_{\mathrm{E}} x$. Intuitively, this breaks forward privacy since from $\mathbf{h}^n(s)$ the attacker can compute s which can be then used to link past sessions to the tag. Indeed, Theorem 1 can be no longer applied to $\mathbf{h}^n(s)$ and property $\mathcal{P}_3$ is violated. On the other hand, if $\mathbf{h}(x) = x+1$, an invertible function, then property $\mathcal{P}_2$ is still satisfied: Theorem 1 can be applied to show that $\prod_{i=0}^{n-1}\{^{\mathbf{g}(\mathbf{h}^i(s))}/_{x_i}\} \perp_s \{^{\mathbf{g}(\mathbf{h}^n(s))}/_{x_n}\}$. Thus the protocol satisfies only untraceability.

On the other hand, if the inverse of $\mathbf{g}$ exists then both properties 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 adversary can first extract $\mathbf{h}^i(s)$ and $\mathbf{h}^j(s)$. Then, since $\mathbf{h}$ is a public hash function, she can apply it $j - i$ times to the first value: if it coincides with the second the adversary can conclude that the outputs belong to the same tag. Indeed, both properties $\mathcal{P}_2, \mathcal{P}_3$ are violated (even though $\mathcal{P}_1$ is still satisfied).

6.2 Basic hash protocol of [9]

The basic hash protocol of [9] is also a single-step protocol, although quite different in spirit than the OSK protocol. It uses a random number generator and a hash function $\mathbf{h}$. The state of each tag is initialized with a randomly generated secret, known to the backend, and is never updated. Instead, on each run 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 known tags, and compares it with the given value to identify the tag.

The simple hash protocol can be modelled as a single-step protocol (Def. 7) with:

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

Then we can prove untraceability by proving the properties $\mathcal{P}_1, \mathcal{P}_2$.

Proposition 5. *The simple hash protocol satisfies properties $\mathcal{P}_1, \mathcal{P}_2$, namely*

$$\begin{array}{lll} \mathcal{P}_1 & \nu s.\nu r.\{^{(r,\mathbf{h}(s,r))}/_{x}\} \approx_s \nu s.\nu r.\{^{(r,\mathbf{h}(s,r))}/_{x}\} & \\ \mathcal{P}_2 & \prod_{i=0}^{n-1} \nu r.\{^{(r,\mathbf{h}(s,r))}/_{x_i}\} \perp_s \nu r.\{^{(r,\mathbf{h}(s,r))}/_{x_n}\} & \forall n \in \mathbb{N} \end{array}$$

$\mathcal{P}_1$ follows trivially from the reflexivity of $\approx_s$. For $\mathcal{P}_2$ we can use Theorem 1 together with Prop. 2. Similarly to OSK, each one of the infinite equivalences that we need to show can be proven automatically by ProVerif.

Thus, by Theorem 2, we conclude that the protocol satisfies untraceability. On the other hand, forward privacy is intuitively violated. Tampering the tag the attacker obtains s which can be then used to link any previous session. Indeed, $\mathcal{P}_3$ is clearly not satisfied.

7 Related and future work

Related work. Several papers ([4, 1, 5, 12, 6, 13, 14]) analyze privacy properties for RFID systems, in various levels of formality. All of them, however, define privacy in a computational setting, typically in terms of games. Our work, on the other hand, takes place in a symbolic setting using the formal language of the applied pi calculus. In Section 3 we briefly describe two types of untraceability games found in the literature and explain that the spirit of our definition is comparable to them. The advantage of using a symbolic model is the clarity of the models and definitions that a formal language provides, the rigorousness of the proofs and the possibility of automatic verification using tools like ProVerif ([10]). On the other hand, a symbolic analysis might miss attacks that exploit weaknesses of the cryptographic primitives.

The work that is closest to ours is the one of Arapinis et al., who independently developed a definition of untraceability in the applied pi calculus. In their recently published paper ([15]), they define the properties of strong and weak untraceability. The former is a strong property requiring that the RFID system is equivalent to one where each tag executes only one session. This is possible because in their model, the attacker cannot choose which tag to communicate with, instead she might get a response from any tag. In our model, however, a tag interface always corresponds to the same tag, thus it is impossible to satisfy such a property, unless a different interpretation is given to "single session". Weak untraceability, on the other hand, bears some similarities to our definition of untraceability, but also several differences. In general, [15] provides interesting alternatives to our definitions, hence we plan to investigate their relation in the near future. Note also that our work provides several results that are outside the scope of [15] which, being a short paper, only states the definitions.

Finally, Deursen et al ([16]) also define untraceability in a symbolic setting. Their model and definitions, however, are quite different than ours, defined in terms of traces.

Future work. There are various directions for future work. Defining the notion of *self-stabilizing backwards privacy* from [17] is a natural extension of our definition of forward privacy. Moreover, we plan to give general results for classes wider than the one of single step protocols, allowing the tag to receive input from the reader. We also aim at automatic verification, using the ProVerif tool (already used in Section 6 in a limited setting). Finally, we plan at studying the relation of our work to the definitions of [15].

References

1. Ohkubo, M., Suzuki, K., Kinoshita, S.: Cryptographic Approach to “Privacy-Friendly” Tags. In: RFID Privacy Workshop, MIT, Massachusetts, USA (2003)
2. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication (2001)
3. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes. I and II. Information and Computation **100** (1989)
4. Juels, A., Weis, S.: Defining Strong Privacy for RFID. In: International Conference on Pervasive Computing and Communications – PerCom 2007, New York City, New York, USA, IEEE, IEEE Computer Society Press (2007) 342–347
5. Avoine, G.: Adversary Model for Radio Frequency Identification. Technical Report LASEC-REPORT-2005-001, Swiss Federal Institute of Technology (EPFL), Security and Cryptography Laboratory (LASEC), Lausanne, Switzerland (2005)
6. Ouafi, K., Phan, R.C.W.: Privacy of Recent RFID Authentication Protocols. In: 4th International Conference on Information Security Practice and Experience – ISPEC 2008. Volume 4991 of Lecture Notes in Computer Science., Sydney, Australia, Springer (2008) 263–277
7. Chatmon, C., van Le, T., Burmester, M.: Secure Anonymous RFID Authentication Protocols. Technical Report TR-060112, Florida State University, Department of Computer Science, Tallahassee, Florida, USA (2006)
8. Nohl, K., Evans, D.: Privacy through Noise: A Design Space for Private Identi?cation. In: Annual Computer Security Applications Conference (ACSAC 2009). (2009)
9. Weis, S., Sarma, S., Rivest, R., Engels, D.: Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In Hutter, D., Müller, G., Stephan, W., Ullmann, M., eds.: International Conference on Security in Pervasive Computing – SPC 2003. Volume 2802 of Lecture Notes in Computer Science., Boppard, Germany, Springer-Verlag (2003) 454–469
10. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: CSFW, IEEE Computer Society (2001) 82–96
11. Blanchet, B.: Automatic proof of strong secrecy for security protocols. In: IEEE Symposium on Security and Privacy, IEEE Computer Society (2004) 86–
12. Avoine, G.: Cryptography in Radio Frequency Identification and Fair Exchange Protocols. PhD thesis, EPFL, Lausanne, Switzerland (2005)
13. Burmester, M., Le, T.v., Medeiros, B.d.: 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, IEEE (2006)
14. Vaudenay, S.: On Privacy Models for RFID. In: Advances in Cryptology - Asiacrypt 2007. Volume 4833 of Lecture Notes in Computer Science., Kuching, Malaysia, Springer-Verlag (2007) 68–87
15. Arapinis, M., Chothia, T., Ritter, E., Ryan, M.: Untraceability in the applied pi calculus. In: proc. of the 1st Int. Workshop on RFID Security and Cryptography. (2009) To appear.
16. van Deursen, T., Mauw, S., Radomirovic, S.: 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., Springer (2008) 115
17. Garcia, F.D., van Rossum, P.: Modeling Privacy for Off-line RFID Systems. In: Workshop on RFID Security – RFIDSec’09, Leuven, Belgium (2009)

A Proofs

Proposition 1 Forward privacy (Def 4) implies untraceability (Def 3).

Proof. The idea is that the processes in the definition of forward privacy are the same as the ones of untraceability with the addition of the action br. By restricting br we can retrieve the definition of untraceability.

Assume that forward privacy holds. We restrict the name br and from the congruence of $\approx$ we get

$$\nu br.\nu\widetilde{n}.(BrTag(c_1,c_2) \mid \mathit{ReplTag} \mid \mathit{Reader} \mid DB) \approx$$
$$\nu br.\nu\widetilde{n}.(TwoTags(c_1,c_2) \mid \mathit{ReplTag} \mid \mathit{Reader} \mid DB)$$

Since $\nu br.br(\_).X \approx 0$, we have that $\nu br.BrTag(c_1,c_2) \approx Tag(c_1,c_2)$ and $\nu br.TwoTags(c_1,c_2) \approx Tag(c_1) \mid Tag(c_2)$, so we conclude that

$$\nu\widetilde{n}.(Tag(c_1,c_2) \mid \mathit{ReplTag} \mid \mathit{Reader} \mid DB) \approx$$
$$\nu\widetilde{n}.(Tag(c_1) \mid Tag(c_2) \mid \mathit{ReplTag} \mid \mathit{Reader} \mid DB)$$

which is the definition of untraceability. □

Proposition 2 Let $\varphi_1, \varphi_2, \psi$ be closed frames such that $\varphi_1 \perp_{\widetilde{s}} \psi$. If either of the following holds:

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

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

Proof. From $\varphi_1 \perp_{\widetilde{s}} \psi$ by definition we get $\nu\widetilde{s}.(\varphi_1 \mid \psi) \approx_s \nu\widetilde{s}.\varphi_1 \mid \nu\widetilde{s}.\psi$.

1. Since $\approx_s$ is closed under application of closing evalution contexts, we have

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

   thus $\varphi_2 \perp_{\widetilde{s}} \psi$.
2. Note that $\varphi_2 = \varphi_1 \mid \varphi_1'$ is closed, so by congruence we get $\nu\widetilde{s}.(\varphi_1 \mid \psi) \mid \varphi_1' \approx_s \nu\widetilde{s}.\varphi_1 \mid \nu\widetilde{s}.\psi \mid \varphi_1'$. Since $\{s\} \cap fn(\varphi_1') = \emptyset$ and $\approx_s$ is closed by $\equiv$, we get $\nu\widetilde{s}.(\varphi_1 \mid \varphi_1' \mid \psi) \approx_s \nu\widetilde{s}.(\varphi_1 \mid \varphi_1') \mid \nu\widetilde{s}.\psi$ and thus $\nu\widetilde{s}.(\varphi_2 \mid \psi) \approx_s \nu\widetilde{s}.\varphi_2 \mid \nu\widetilde{s}.\psi$ which implies $\varphi_2 \perp_{\widetilde{s}} \psi$
3. Again by simple application of the congruence property. □

Lemma 2. *Let $\varphi = \nu\widetilde{n}_1.\sigma_1, \psi = \nu\widetilde{n}_2.\sigma_2$ be frames in canonical form.*

1. *If M, N are two terms such that $\{\widetilde{n}_1\} \cap (fn(M) \cup fn(N)) = \emptyset$, then $(M = N)\varphi$ iff $M\sigma_1 = N\sigma_1$.*

2. *If $\varphi \not\approx_{\mathrm{s}} \psi$ then there exist terms M, N such that $\{\widetilde{n}_1, \widetilde{n}_2\} \cap (fn(M) \cup fn(N)) = \emptyset$, $M\sigma_1 = N\sigma_1$ and $M\sigma_2 \neq N\sigma_2$.*

Lemma 1 Let $\mathbf{h}$ be a hash function (Def. 6) 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 equation that inverts $\mathbf{h}$, i.e. $\mathbf{invh}(\mathbf{h}(x)) =_{\mathrm{E}} x$
4. there is no equation that checks a hashed value, i.e. $\mathbf{checkh}(\mathbf{h}(x)) =_{\mathrm{E}} \mathbf{ok}$

Proof. 1. Assume $\mathbf{h}(M) =_{\mathrm{E}} \mathbf{h}(N)$ and $M \neq_{\mathrm{E}} N$. Then by Def. 6 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 $\mathbf{h}(x)$. □

Depending on the equational theory, there might exist terms M, N such that $M =\!\lhd N$ and $N =\!\lhd M$. For example take $\mathbf{f}(\mathbf{g}(a)), \mathbf{g}(\mathbf{f}(b))$ with $\mathbf{f}(x) = \mathbf{f}(y), \mathbf{g}(x) = \mathbf{g}(y)$. The following lemma says that for any sequence of terms, we can replace them with equal terms and put them in an order such that no term is equal to a subterm of any subsequent term. In the previous example such terms would be $\mathbf{f}(\mathbf{g}(a)), \mathbf{g}(a)$. The lemma is needed for the proof of Theorem 1.

Lemma 3. *Let $M_1, \ldots, M_l$ be terms. There exist terms $M'_1, \ldots, M'_l$ and a permutation π of $\{1, \ldots, l\}$ such that $M'_i = M_i, 1 \leq i \leq l$ and*

$$M'_{\pi(i)} \neq\!\lhd M'_{\pi(j)} \quad \forall 1 \leq i \leq j \leq l$$

Theorem 1 Let $\mathbf{h}_1, \ldots, \mathbf{h}_k, \mathbf{g}_1, \ldots, \mathbf{g}_l$ be hash functions (Def. 6), not necessarily distinct, and let

$$\varphi_1 = \{^{\mathbf{h}_1(S_1)}/_{x_1}, \ldots, ^{\mathbf{h}_k(S_k)}/_{x_k}\} \qquad \varphi_2 = \{^{\mathbf{g}_1(T_1)}/_{y_1}, \ldots, ^{\mathbf{g}_l(T_l)}/_{y_l}\}$$

be frames in canonical form. Assume that $\mathbf{h}_i(S_i) \neq\!\trianglelefteq \mathbf{g}_j(T_j)$ and $\mathbf{g}_j(T_j) \neq\!\trianglelefteq \mathbf{h}_i(S_i)$ for all $1 \leq i \leq l, 1 \leq j \leq m$. Then $\varphi_1 \perp_s \varphi_2$ for all names s.

Proof. We begin by making some assumptions, without loss of generality. Let σ_1^i be the same as φ_1 with x_i removed. First, we assume that no $\mathbf{h}_i(S_i)$ is redundant in the sense that there is no term M with $fn(M) \cap \{s\} = \emptyset$ such that $M\sigma^i = \mathbf{h}_i(S_i)$. Otherwise we have that $\varphi_1 \equiv \sigma_1^i \mid \{^M/_{x_i}\}$, so we can apply this theorem to σ_1^i and use Lemma 2 to extend it to φ_1. Similarly for $\mathbf{g}_j(T_j)$. Then we assume that there exist permutations π, ρ such that

$$\mathbf{h}_{\pi_i}(S_{\pi_i}) \not\trianglelefteq \mathbf{h}_{\pi_j}(S_{\pi_j}) \qquad \forall 1 \le i \le j \le l \tag{1}$$

$$\mathbf{g}_{\rho_i}(T_{\rho_i}) \not\trianglelefteq \mathbf{g}_{\rho_j}(T_{\rho_j}) \qquad \forall 1 \le i \le j \le m \tag{2}$$

Otherwise, by Lemma 3 we can find $M_1, \dots, M_k$ such that $M_i = \mathbf{h}_i(S_i)$ $1 \le i \le l$, and a permutation with the desired property. Then we can create a frame φ_1' from φ_1 by replacing $\mathbf{h}_i(S_i)$ by M_i and apply the theorem to φ_1', since $\varphi_1 \approx_s \varphi_1'$ and by Lemma 2 we have $\varphi_1 \perp_s \varphi_2 \Leftrightarrow \varphi_1' \perp_s \varphi_2$. Note that since $M_i = \mathbf{h}_i(T_i)$ and $M_i \not\trianglelefteq M_i$, we have by Lemma 1 that M_i is of the form $\mathbf{h}_i(M_i')$ so the theorem can be applied. Similarly for (2).

Assume that $\varphi_1 \not\perp_s \varphi_2$, then by definition

$$\nu s.(\varphi_1 \mid \varphi_2) \not\approx_s \nu s.\varphi_1 \mid \nu s.\varphi_2 \tag{3}$$

Let $T_j' = T_j[^{s'}/_{s}]$. By renaming the second occurrence of s in the right-hand side of (3) to some $s' \notin fn(\varphi_1) \cup fn(\varphi_2)$ we get

$$\begin{aligned} &\nu s.\sigma_1 \not\approx_s \nu s.\nu s'.\sigma_2 \quad \text{where} \\ &\sigma_1 = \{^{\mathbf{h}_1(S_1)}/_{x_1}, \dots, {}^{\mathbf{h}_n(S_n)}/_{x_n}, {}^{\mathbf{g}_1(T_1)}/_{y_1}, \dots, {}^{\mathbf{g}_l(T_l)}/_{y_l}\} \\ &\sigma_2 = \{^{\mathbf{h}_1(S_1)}/_{x_1}, \dots, {}^{\mathbf{h}_n(S_n)}/_{x_n}, {}^{\mathbf{g}_1(T_1')}/_{y_1}, \dots, {}^{\mathbf{g}_l(T_l')}/_{y_l}\} \end{aligned}$$

Then, by Lemma 2, there exist terms M, N not containing s, s', such that $M\sigma_1 = N\sigma_1$ but $M\sigma_2 \neq N\sigma_2$.

The idea is to first replace $\mathbf{g}_j(T_j)$ in $M\sigma_1, N\sigma_1$ by x_j and then apply σ_2 to replace x_j by $\mathbf{g}_j(T_j')$. We do the substitutions according to the permutation ρ of (2). This is to avoid substituting a "part" of a term $\mathbf{g}_j(T_j)$ when replacing some other $\mathbf{g}_i(T_i)$. We want to show that

$$M\sigma_1[^{y_i}/_{\mathbf{g}_{\rho_i}(=T_{\rho_i})}]_{i=1}^m \, \sigma_2 = M\sigma_2 \qquad N\sigma_1[^{y_i}/_{\mathbf{g}_{\rho_i}(=T_{\rho_i})}]_{i=1}^m \, \sigma_2 = N\sigma_2 \tag{4}$$

Note that $M\sigma_1, M\sigma_2$ differ only in the terms $\mathbf{g}_j(T_j), \mathbf{g}_j(T_j')$ that are substituted for y_j. However, $M\sigma_1[^{y_j}/_{\mathbf{g}_j(=T_j)}]$ replaces *all* subterms equal to $\mathbf{g}_j(T_j)$, not just those substituted for y_j, so some attention is needed.

The proof of (4) is by induction on the structure of M:

– M is a variable. We have three sub-cases:
  - if $M = y_j$ then $M\sigma_1 = \mathbf{g}_j(T_j)$. Let q be such that $\rho_q = j$. Because of (2), $\mathbf{g}_j(T_j)$ will remain unaffected by the first $q-1$ substitutions, that is

$$M\sigma_1[^{y_i}/_{\mathbf{g}_{\rho_i}(=T_{\rho_i})}]_{i=1}^{q-1} = \mathbf{g}_j(T_j)$$

    Then the q-th substitution will replace $\mathbf{g}_j(T_j)$ by y_j and the remaining substitutions again will have no effect. Thus $M\sigma_1[^{y_i}/_{\mathbf{g}_{\rho_i}(=T_{\rho_i})}]_{i=1}^m = y_j$ and $M\sigma_1[^{y_i}/_{\mathbf{g}_{\rho_i}(=T_{\rho_i})}]_{i=1}^m\sigma_2 = \mathbf{g}_j(T_j') = M\sigma_2$.

  - if $M = x_i$ then $M\sigma_1 = \mathbf{h}_i(S_i)$ and since we assume that $\mathbf{g}_j(T_j) \not\trianglelefteq \mathbf{h}_i(S_i)$, $1 \leq j \leq m$, we have that $\mathbf{h}_i(S_i)$ remains unaffected by all substitutions, thus $M\sigma_1[{}^{y_i}/_{\mathbf{g}_{\rho_i}(=T_{\rho_i})}]_{i=1}^{m}\sigma_2 = \mathbf{h}_i(S_i) = M\sigma_2$.
  - If M is equal to some other variable z then $M\sigma_1 = z$ and $M\sigma_1[{}^{y_i}/_{\mathbf{g}_{\rho_i}(=T_{\rho_i})}]_{i=1}^{m}\sigma_2 = z = M\sigma_2$.
- If $M = n$ for some name n then $M\sigma_1 = n$ and $M\sigma_1[{}^{y_i}/_{\mathbf{g}_{\rho_i}(=T_{\rho_i})}]_{i=1}^{m}\sigma_2 = n = M\sigma_2$.
- $M = \mathbf{g}(M_1, \ldots, M_k)$ with $k > 1$ or $\forall 1 \leq j \leq m : (\mathbf{g} \neq \mathbf{g}_j$ or $M_1\sigma_1 \neq T_j)$. Then

  $$M\sigma_1[{}^{y_i}/_{\mathbf{g}_{\rho_i}(=T_{\rho_i})}]_{i=1}^{m}\sigma_2 = \mathbf{g}(M_1\sigma_1[{}^{y_i}/_{\mathbf{g}_{\rho_i}(=T_{\rho_i})}]_{i=1}^{m}\sigma_2, \ldots, M_k\sigma_1[{}^{y_i}/_{\mathbf{g}_{\rho_i}(=T_{\rho_i})}]_{i=1}^{m}\sigma_2)$$

  and the proof follows directly from the induction hypothesis.
- The only remaining case is when $M = \mathbf{g}_j(M')$ with $M'\sigma_1 = T_j$. We are going to show that this case is in fact not possible under the assumptions we have made. We replace all occurrences of $\mathbf{h}_i(S_i)$ in $M'\sigma_1$ by a fresh variable z_i, in the order defined by the permutation π of (2). We also replace $\mathbf{g}_j T_j$ by z. Let

  $$K = M'\sigma_1[{}^{z_i}/_{\mathbf{h}_{\pi_i}(=S_{\pi_i})}]_{i=1}^{l}[{}^{z}/_{\mathbf{g}_j(=T_j)}]$$

  Since $\mathbf{h}_i(T_i) \not\trianglelefteq \mathbf{g}_j(T_j)$, T_j is not affected by $[{}^{z_i}/_{\mathbf{h}_{\pi_i}(=S_{\pi_i})}]_{i=1}^{l}$. Moreover, $\mathbf{g}_j(T_j) \not\trianglelefteq T_j$ (from (2)) so T_j is also not affected by $[{}^{z}/_{\mathbf{g}_j(=T_j)}]$. Thus from $M'\sigma_1 = T_j$ and Def 6 we get $K = T_j$ and thus $\mathbf{g}_j(K) = \mathbf{g}_j(T_j)$. But it is easy to see that K has no occurrence of $s, \widetilde{m}$ except from inside some $\mathbf{g}_k(T_k), k \neq j$ which contradicts the assumption that $\mathbf{g}_j(T_j)$ is not redundant.

Having proven (4), from $M\sigma_1 = N\sigma_1$, Def. 6 and the fact that equations are closed under substitution, we get $M\sigma_2 = N\sigma_2$ which is a contradiction. □

Theorem 2 A single step protocol satisfies untraceability iff it satisfies both $\mathcal{P}_1, \mathcal{P}_2$.

Proof.
- ($\mathcal{P}_1 \wedge \mathcal{P}_2 \Rightarrow$ untraceability)
  Assumptions:
  1. $\mathcal{P}_1$: $\nu s.\rho_x \pi(\sigma^n(s)) \approx_s \nu s.\rho_x \pi(\sigma^m(s)) \forall n, m \in \mathbb{N}$
  2. $\mathcal{P}_2$: $\prod_{i=0}^{n-1} \rho_{x_i} \pi(\sigma^i(s)) \perp_s \rho_{x_n} \pi(\sigma^n(s)) \forall n \in \mathbb{N}$

  We want to prove that:

  $$\nu w.\nu s.(!P(w, c_1) \mid !P(w, c_2) \mid St(w, s)) \mid \bar{t}\langle\_\rangle$$
  $$\approx$$
  $$\nu w.\nu s.(!P(w, c_1) \mid St(w, s)) \mid \nu w.\nu s.(!P(w, c_2) \mid St(w, s)) \mid \bar{t}\langle\_\rangle$$

  Since it has been proved that observational bisimilarity is equivalent to labelled bisimilarity ($\approx_l$), for the sake of simplicity we use the second approach to prove the theorem. Moreover we synchronize the processes by means of a token t, so that it is easier to show the relationship between untraceability and forward privacy for this class of protocols.
  To prove a labelled bisimilarity between processes we have to show that there exists a relation $\mathscr{R}$ on closed extended processes such that $A \mathscr{R} B$ implies:

1. $A \approx_s B$;
2. if $A \to A'$, then $\exists B'$ s.t. $B \to^* B'$ and $A' \mathscr{R} B'$;
3. if $A \xrightarrow{\alpha} A'$ and $fv(\alpha) \subseteq dom(A)$ and $bn(\alpha) \cap fn(B) = \emptyset$, then $\exists B'$ s.t. $B \to^* \xrightarrow{\alpha} \to^* B'$ and $A' \mathscr{R} B'$.

For the sake of readability we use the following notation:

- $P_{i,j} = P(w_i, c_j) = c_j(\_).t(\_).w_i(x).(\nu\rho.\bar{c_j}\langle\pi(x)\rangle \mid St(w_i, \sigma(x)))$;
- $Q_{i,j} = Q(w_i, c_j) = t(\_).w_i(x).(\nu\rho.\bar{c_j}\langle\pi(x)\rangle \mid St(w_i, \sigma(x)))$;
- $\psi_l, \psi_{r1}, \psi_{r2}$ are frames with the following forms:
  1. $\psi_l = \prod_{i=0}^{k+l+1} \rho_{x_i}\pi(\sigma^i(s_0))$
  2. $\psi_{r1} = \prod_{i=0}^{k} \rho_{x_{\alpha_i}}\pi(\sigma^i(s_1))$
  3. $\psi_{r2} = \prod_{i=0}^{l} \rho_{x_{\beta_j}}\pi(\sigma^j(s_2))$

  $\forall$ increasing sequences α_i, β_j s.t. $\{\alpha_i \mid 0 \le i \le k\} \cup \{\beta_j \mid 0 \le j \le l\} = [0, ..., k+l+1]$

The relation $\mathscr{R}$ is the following:

$\mathscr{R} = \{$

$(\nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q_{0,1}^n \mid Q_{0,2}^m \mid \bar{t}\langle\_\rangle \mid St(w_0, \sigma^{k+l+1}(s_0))),$
$\nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q_{1,1}^n \mid St(w_1, \sigma^k(s_1))) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q_{2,2}^m \mid St(w_2, \sigma^l(s_2))) \mid \bar{t}\langle\_\rangle),$

$(\nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q_{0,1}^n \mid Q_{0,2}^m \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid$
$w_0(x).\nu\rho.\bar{c_1}\langle\pi(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(x)))),$
$\nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q_{1,1}^n \mid St(w_1, \sigma^k(s_1)) \mid$
$w_1(x).\nu\rho.\bar{c_1}\langle\pi(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_1, \sigma(x)))) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q_{2,2}^m \mid St(w_2, \sigma^l(s_2)))),$

$(\nu s_0, w_0.(\psi_l \mid !P_{0_1} \mid !P_{0,2} \mid Q_{0,1}^n \mid Q_{0,2}^m \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid$
$w_2(x).\nu\rho.\bar{c_2}\langle\pi(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_2, \sigma(x)))),$
$\nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q_{1,1}^n \mid St(w_1, \sigma^k(s_1))) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q_{2,2}^m \mid St(w_2, \sigma^l(s_2))) \mid$
$w_2(x).\nu\rho.\bar{c_2}\langle\pi(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_2, \sigma(x)))),$

$(\nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q_{0,1}^n \mid Q_{0,2}^m \mid$
$\nu\rho.\bar{c_1}\left\langle\pi(\sigma^{k+l+1}(s_0))\right\rangle.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(\sigma^{k+l+1}(s_0))))),$
$\nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q_{1,1}^n \mid \nu\rho.\bar{c_1}\left\langle\pi(\sigma^k(s_1))\right\rangle.(\bar{t}\langle\_\rangle \mid$
$St(w_1, \sigma(\sigma^k(s_1))))) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q_{2,2}^m \mid St(w_2, \sigma^l(s_2)))),$

$(\nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q_{0,1}^n \mid Q_{0,2}^m \mid$
$\nu\rho.\bar{c_2}\left\langle\pi(\sigma^{k+l+1}(s_0))\right\rangle.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(\sigma^{k+l+1}(s_0))))),$
$\nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q_{1,1}^n \mid St(w_1, \sigma^k(s_1))) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q_{2,2}^m \mid \bar{c_2}\left\langle\pi(\sigma^l(s_2))\right\rangle.(\bar{t}\langle\_\rangle \mid St(w_2, \sigma(\sigma^l(s_2))))))$

s.t. $\forall k, l, n, m \in \mathbb{N}$ s.t. :
k, l are the numbers of key updates of $P_{1,1}$ and $P_{2,2}$ respectively
n, m are the numbers of processes of the form $Q_{1,1}$ and $Q_{2,2}$ respectively

}

Now we have to show that the relation $\mathscr{R}$ is a bisimulation and all its pairs are statically equivalent.
Cases:

- $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid \bar{t}\langle\_\rangle \mid St(w_0, \sigma^{k+l+1}(s_0)))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1))) \mid$
  $\quad \nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2))) \mid \bar{t}\langle\_\rangle$
  The possible transitions for L leads to new forms in which the token synchronizes with $Q^n_{0,1}$ or $Q^m_{0,2}$. In both cases these transitions can be matched by R, generating a new pair which belongs again to $\mathscr{R}$:
  * $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^{n-1}_{0,1} \mid Q^m_{0,2} \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid$
    $\quad w_0(x).\nu\rho.\bar{c_1}\langle\pi(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(x))))$
    $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^{n-1}_{1,1} \mid St(w_1, \sigma^k(s_1)) \mid$
    $\quad w_1(x).\nu\rho.\bar{c_1}\langle\pi(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_1, \sigma(x)))) \mid$
    $\quad \nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))$
  * $L = \nu s_0, w_0.(\psi_l \mid !P_{0_1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^{m-1}_{0,2} \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid$
    $\quad w_2(x).\nu\rho.\bar{c_2}\langle\pi(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_2, \sigma(x))))$
    $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1))) \mid$
    $\quad \nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^{m-1}_{2,2} \mid St(w_2, \sigma^l(s_2))) \mid$
    $\quad w_2(x).\nu\rho.\bar{c_2}\langle\pi(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_2, \sigma(x)))$
- $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid$
  $\quad w_0(x).\nu\rho.\bar{c_1}\langle\pi(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(x))))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1)) \mid$
  $\quad w_1(x).\nu\rho.\bar{c_1}\langle\pi(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_1, \sigma(x)))) \mid$
  $\quad \nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))$
  L can perform only an internal reduction, namely it can synchronize the input and output processes, because the token blocks the processes $Q^n_{0,1}$ and $Q^m_{0,2}$. Again R matches the same reduction and the new processes belong to $\mathscr{R}$:
  $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid$
  $\quad \nu\rho.\bar{c_1}\left\langle\pi(\sigma^{k+l+1}(s_0))\right\rangle.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(\sigma^{k+l+1}(s_0)))))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid \nu\rho.\bar{c_1}\left\langle\pi(\sigma^k(s_1))\right\rangle.(\bar{t}\langle\_\rangle \mid$
  $\quad St(w_1, \sigma(\sigma^k(s_1))))) \mid$
  $\quad \nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))$
- $L = \nu s_0, w_0.(\psi_l \mid !P_{0_1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid$
  $\quad w_2(x).\nu\rho.\bar{c_2}\langle\pi(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_2, \sigma(x))))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1))) \mid$
  $\quad \nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2))) \mid$
  $\quad w_2(x).\nu\rho.\bar{c_2}\langle\pi(x)\rangle.(\bar{t}\langle\_\rangle \mid St(w_2, \sigma(x)))$
  This case is similar to the previous one.
- $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid$
  $\quad \nu\rho.\bar{c_1}\left\langle\pi(\sigma^{k+l+1}(s_0))\right\rangle.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(\sigma^{k+l+1}(s_0)))))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid \nu\rho.\bar{c_1}\left\langle\pi(\sigma^k(s_1))\right\rangle.(\bar{t}\langle\_\rangle \mid$
  $\quad St(w_1, \sigma(\sigma^k(s_1))))) \mid \nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))$
  Also in this case, L can perform only a transition, namely it can send $\pi(\sigma^{k+l+1}(s_0))$

on the channel c_1. The new pair will be:
$L = \nu s_0, w_0.(\psi'_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid \bar{t}\langle\_\rangle \mid St(w_0, \sigma(\sigma^{k+l+1}(s_0))))$
$R = \nu s_1, w_1.(\psi'_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma(\sigma^k(s_1)))) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2))) \mid \bar{t}\langle\_\rangle$
The first process is equivalent to the L process analyzed in the first case of this proof. The only difference is in the frame, which is correctly updated adding a new substitution related to the last output, since k is increased by one; the substitution does not change the form of the frame. Using the same reasoning and the structural equivalence it is possible to map the second process to the corresponding R.

- $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid$
  $\nu\rho.\bar{c_2}\left\langle\pi(\sigma^{k+l}(s_0))\right\rangle.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(\sigma^{k+l+1}(s_0)))))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1))) \mid$
  $\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid \bar{c_2}\left\langle\pi(\sigma^l(s_2))\right\rangle.(\bar{t}\langle\_\rangle \mid St(w_2, \sigma(\sigma^l(s_2)))))$
  This case is similar to the previous one.

Note that in all the cases we should also consider that the processes L and R might spawn a copy of $Q_{0,1}$ and $Q_{1,1}$ or $Q_{0,2}$ and $Q_{2,2}$ respectively. They trivially belong to the relation, because they only increase the powers of the corresponding processes, keeping the same form.
Since the frames in all the pairs are always in the same form, the static equivalence can be shown proving only that:

$$\nu s_0.(\psi_l) \approx_s \nu s_1.(\psi_{r_1}) \mid \nu s_2.(\psi_{r_2})$$

Note that whenever the property $\mathcal{P}_2$ holds, it is possible to split a frame of the form $\nu s.\prod_{i=0}^n \rho_{x_i}\pi(\sigma^i(s))$ in a new frame $\prod_{i=0}^n \nu s.\rho_{x_i}\pi(\sigma^i(s))$:

$$\begin{aligned}
& \nu s.(\textstyle\prod_{i=0}^{n-1} \rho_{x_i}\pi(\sigma^i(s)), \rho_{x_n}\pi(\sigma^n(s))) \\
\approx_s\ & \nu s.\textstyle\prod_{i=0}^{n-1} \rho_{x_i}\pi(\sigma^i(s)) \mid \nu s.\rho_{x_n}\pi(\sigma^n(s)) \\
\approx_s\ & \nu s.\textstyle\prod_{i=0}^{n-2} \rho_{x_i}\pi(\sigma^i(s)) \mid \nu s.\rho_{x_{n-1}}\pi(\sigma^{n-1}(s)) \mid \nu s.\rho_{x_n}\pi(\sigma^n(s)) \\
\approx_s\ & \cdots \\
\approx_s\ & \textstyle\prod_{i=0}^{n} \nu s.\rho_{x_i}\pi(\sigma^i(s))
\end{aligned}$$

Assuming properties $\mathcal{P}_1$ and $\mathcal{P}_2$, we can conclude that:

$$\begin{aligned}
\nu s_0.(\psi_l) =\ & \nu s_0.\textstyle\prod_{i=0}^{k+l+1} \nu\rho_{x_i}.\pi(h^i(s_0)) & \\
\approx_s\ & \textstyle\prod_{i=0}^{k+l+1} \nu s_0.\nu\rho_{x_i}.\pi(\sigma^i(s_0)) & \text{by } \mathcal{P}_2 \\
\approx_s\ & \textstyle\prod_{i=0}^{k} \nu s_0.\nu\rho_{x_{\alpha_i}}.\pi(\sigma^i(s_0)) \mid & \\
& \textstyle\prod_{j=0}^{l} \nu s_0.\nu\rho_{x_{\alpha_j}}.\pi(\sigma^j(s_0)) & \text{by } \mathcal{P}_1 \\
\approx_s\ & \nu s_0.\textstyle\prod_{i=0}^{k} \nu\rho_{x_{\alpha_i}}.\pi(\sigma^i(s_0)) \mid & \\
& \nu s_0.\textstyle\prod_{j=0}^{l} \nu\rho_{x_{\alpha_j}}.\pi(\sigma^j(s_0)) & \text{by } \mathcal{P}_2 \\
\approx_s\ & \nu s_1.\textstyle\prod_{i=0}^{k} \nu\rho_{x_{\alpha_i}}.\pi(\sigma^i(s_1)) \mid & \\
& \nu s_2.\textstyle\prod_{j=0}^{l} \nu\rho_{x_{\alpha_j}}.\pi(\sigma^j(s_2)) & \text{by } \alpha\text{-renaming} \\
=\ & \nu s_1.(psi_{r1}) \mid \nu s_2.(\psi_{r2}) &
\end{aligned}$$

$\forall$ increasing sequences α_i, β_j s.t. $\{\alpha_i \mid 0 \leq i \leq k\} \cup \{\beta_j \mid 0 \leq j \leq l\} = [0, ..., k+l+1]$ Therefore the equivalence between frames holds.

– ($\neg(\mathcal{P}_1 \wedge \mathcal{P}_2) \Rightarrow \neg$ untraceability)
  Assumptions: $\neg\mathcal{P}_1 \vee \neg\mathcal{P}_2$
  We want to prove that:

$$\neg(\nu w.\nu s.(!P(w,c_1) \mid !P(w,c_2) \mid St(w,s)) \mid \bar{t}\langle\_\rangle$$
$$\approx_l$$
$$\nu w.\nu s.(!P(w,c_1) \mid St(w,s)) \mid \nu w.\nu s.(!P(w,c_2) \mid St(w,s)) \mid \bar{t}\langle\_\rangle)$$

  Cases:

  1. ($\neg\mathcal{P}_1$)
     We know that $\mathcal{P}_1$ does not hold. This means that exist k and l ($k < l$) such that:
     $$\neg(\nu s.\rho_x\pi(\sigma^k(s)) \approx_s \nu s.\rho_x\pi(\sigma^l(s)))$$
     Now the adversary is able to break untraceability querying k times the first interface, $l-k$ the second, and again the first, obtaining two frames that are no longer statically equivalent. In fact, the labelled equivalence is broken, since the frames have two values in correspondence to the same variable that are not statically equivalent by assumption:
     (a) $\nu s_0.(\rho_{x_0}\pi(s) \mid \ldots \mid \rho_{x_{k-1}}\pi(\sigma^{k-1}(s)) \mid \rho_{x_k}\pi(\sigma^k(s)) \mid \ldots \mid \rho_{x_{l-1}}\pi(\sigma^{l-1}(s)) \mid \rho_{x_l}\pi(\sigma^l(s)))$
     (b) $\nu s_1.(\rho_{x_0}\pi(s) \mid \ldots \mid \rho_{x_{k-1}}\pi(\sigma^{k-1}(s)) \mid \rho_{x_l}\pi(\sigma^k(s))) \mid \nu s_2.(\rho_{x_k}\pi(s) \mid \ldots \mid \rho_{x_{l-1}}\pi(\sigma^{l-1}(s)))$
  2. ($\neg\mathcal{P}_2$)
     We know that $\mathcal{P}_2$ does not hold. This means that exists n such that:
     $$\neg(\textstyle\prod_{i=0}^{n-1} \rho_{x_i}\pi(\sigma^i(s)) \perp_s \rho_{x_n}\pi(\sigma^n(s)))$$
     Cases:
     (a) ($\neg\mathcal{P}_1$)
        Untraceability does not hold by (1.).
     (b) ($\mathcal{P}_1$)
        Again we break the frame equivalence querying n times the first interface and once the second. We obtain the following frames:
        i. $\nu s_0.(\rho_{x_0}\pi(s) \mid \ldots \mid \rho_{x_n-1}\pi(\sigma^{n-1}(s)) \mid \rho_{x_n}\pi(\sigma^n(s)))$
        ii. $\nu s_1.(\rho_{x_0}\pi(s) \mid \ldots \mid \rho_{x_n-1}\pi(\sigma^{n-1}(s))) \mid \nu s_2.\rho_{x_n}\pi(s)$
            $\equiv \nu s_1.(\rho_{x_0}\pi(s) \mid \ldots \mid \rho_{x_n-1}\pi(\sigma^{n-1}(s))) \mid \nu s_2.\rho_{x_n}\pi(\sigma^n(s))$ by $\mathcal{P}_1$
        By assumption we know that these frames are not statically equivalent, and this breaks the labelled bisimilarity.

□

Proposition 3 For all single step protocols, $\mathcal{P}_3 \Rightarrow \mathcal{P}_2$.

Proof. The intuition is that $\mathcal{P}_3$ requires the state content after n runs ($\sigma^n(S^0)$) to be independent from the output of the n runs. $\mathcal{P}_2$ requires instead the independence of the $(n+1)$-th output ($\pi(\sigma^n(S^0))$). However, since the only occurrences of the secret s in the $(n+1)$-th come from $\sigma^n(S^0)$, it cannot introduce a dependency on s since $\sigma^n(S^0)$ is independent.

Formally, assuming $\mathcal{P}_3$ we have

$$\textstyle\prod_{i=0}^{n-1} \rho_x \pi(\sigma^i(s)) \perp_s \{^{\sigma^n(s)}/_y\} \quad \forall n \in \mathbb{N}$$

we extend the right-hand side with $\{^{\pi(y)}/_{x_n}\}$ (note that $s \notin \pi(y)$) and we restrict y. From Prop. 2 we get

$$\begin{array}{lll} \prod_{i=0}^{n-1} \rho_x \pi(\sigma^i(s)) \perp_s \nu y.(\{^{\sigma^n(s)}/_y\} \mid \{^{\pi(\sigma^n(s))}/_{x_n}\}) & \forall n \in \mathbb{N} & \Rightarrow \\ \prod_{i=0}^{n-1} \rho_{x_i} \pi(\sigma^i(s)) \perp_s \{^{\pi(\sigma^n(s))}/_{x_n}\} & \forall n \in \mathbb{N} & \end{array}$$

Finally, since $\{\widetilde{\rho}\} \cap fn(\prod_{i=0}^{n-1} \rho_{x_i} \pi(\sigma^i(s))) = \emptyset$ we can restrict by Prop. 2 the channels $\widetilde{\rho}$ and get

$$\textstyle\prod_{i=0}^{n-1} \rho_{x_i} \pi(\sigma^i(s)) \perp_s \rho_{x_n} \pi(\sigma^n(s)) \quad \forall n \in \mathbb{N}$$

which is $\mathcal{P}_1$. □

Theorem 3 A single step protocol satisfies forward privacy iff it satisfies $\mathcal{P}_1$ and $\mathcal{P}_3$.

Proof. – ($\mathcal{P}_1 \wedge \mathcal{P}_3 \Rightarrow$ forward privacy)

Assumptions:

1. $\mathcal{P}_1$: $\nu s.\rho_x \pi(\sigma^n(s)) \approx_s \nu s.\rho_x \pi(\sigma^m(s)) \forall n, m \in \mathbb{N}$
2. $\mathcal{P}_3$: $\prod_{i=0}^{n-1} \rho_{x_i} \pi(\sigma^i(s)) \perp_s \{^{\sigma^n(s)}/_{x_n}\} \forall n \in \mathbb{N}$

We want to prove that:

$$\begin{array}{c} \nu w.\nu s.(!P(w,c_1) \mid !P(w,c_2) \mid St(w,s) \mid Break(w_0)) \mid \bar{t}\langle\_\rangle \\ \approx \\ \nu w.\nu s.(!P(w,c_1) \mid St(w,s) \mid Break(w_1)) \mid \\ \nu w.\nu s.(!P(w,c_2) \mid St(w,s)) \mid \bar{t}\langle\_\rangle \end{array}$$

where $Break(w) = br(\_).t(\_).w(x).\overline{br}\langle x\rangle$. The method to prove labelled bisimilarity and the notation used are the same of the previous section. As in the previous proof we use labelled bisimilarity to prove the theorem, since it is equivalent to observational bisimilarity. For the sake of readability we use $Break(w)$ with two meanings, namely $br(\_).t(\_).w(x).\overline{br}\langle x\rangle$ and $t(\_).w(x).\overline{br}\langle x\rangle$. The first process models the ability of the adversary who can ask for the secret of the first tag; the process gets the token, reads the secret and publishes it on the public channel when the adversary can read it. The second process is exactly the same process, but already triggered and waiting for the token.

The relation $\mathscr{R}$ is the following:

$\mathscr{R} = \{$

$(\nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q_{0,1}^n \mid Q_{0,2}^m \mid \bar{t}\langle\_\rangle \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid Break(w_0)),$
$\nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q_{1,1}^n \mid St(w_1, \sigma^k(s_1)) \mid Break(w_1)) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q_{2,2}^m \mid St(w_2, \sigma^l(s_2))) \mid \bar{t}\langle\_\rangle),$

$(\nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q_{0,1}^n \mid Q_{0,2}^m \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid w(x).\overline{br}\langle x\rangle),$
$\nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q_{1,1}^n \mid St(w_1, \sigma^k(s_1)) \mid w(x).\overline{br}\langle x\rangle) \mid$

$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))),$

$(\nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid \overline{br}\langle\sigma^{k+l+1}(s_0)\rangle),$
$\nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid \overline{br}\langle\sigma^k(s_1)\rangle) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))),$

$(\nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid \{^{\sigma^{k+l+1}(s_0)}/_{x_{k+l}}\}),$
$\nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid \{^{\sigma^k(s_1)}/_{x_{k+l}}\}) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))),$

$(\nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid$
$w_0(x).\nu\rho.\bar{c_1}\,\langle\pi(x)\rangle\,.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(x))) \mid Break(w_0)),$
$\nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1)) \mid$
$w_1(x).\nu\rho.\bar{c_1}\,\langle\pi(x)\rangle\,.(\bar{t}\langle\_\rangle \mid St(w_1, \sigma(x))) \mid Break(w_1)) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))),$

$(\nu s_0, w_0.(\psi_l \mid !P_{0_1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid$
$w_2(x).\nu\rho.\bar{c_2}\,\langle\pi(x)\rangle\,.(\bar{t}\langle\_\rangle \mid St(w_2, \sigma(x))) \mid Break(w_0)),$
$\nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1)) \mid Break(w_1)) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2))) \mid$
$w_2(x).\nu\rho.\bar{c_2}\,\langle\pi(x)\rangle\,.(\bar{t}\langle\_\rangle \mid St(w_2, \sigma(x)))),$

$(\nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid$
$\nu\rho.\bar{c_1}\,\left\langle\pi(\sigma^{k+l+1}(s_0))\right\rangle.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(\sigma^{k+l+1}(s_0)))) \mid Break(w_0)),$
$\nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid \nu\rho.\bar{c_1}\,\left\langle\pi(\sigma^k(s_1))\right\rangle.(\bar{t}\langle\_\rangle \mid$
$St(w_1, \sigma(\sigma^k(s_1)))) \mid Break(w_1)) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))),$

$(\nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid$
$\nu\rho.\bar{c_2}\,\left\langle\pi(\sigma^{k+l+1}(s_0))\right\rangle.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(\sigma^{k+l+1}(s_0)))) \mid Break(w_0)),$
$\nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1)) \mid Break(w_1)) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid \bar{c_2}\,\left\langle\pi(\sigma^l(s_2))\right\rangle.(\bar{t}\langle\_\rangle \mid St(w_2, \sigma(\sigma^l(s_2))))))$

s.t. $\forall k, l, n, m \in \mathbb{N}$ s.t. :
k, l are the numbers of key updates of $P_{1,1}$ and $P_{2,2}$ respectively
n, m are the numbers of processes of the form $Q_{1,1}$ and $Q_{2,2}$ respectively
}

Now we have to show that the relation $\mathscr{R}$ is a bisimulation and all its pairs are statically equivalent.
Cases:

- $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid \bar{t}\langle\_\rangle \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid Break(w_0))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1)) \mid Break(w_1)) \mid$
  $\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2))) \mid \bar{t}\langle\_\rangle$
  The possible transitions for L leads to new forms in which the token synchro-

nizes with $Q^n_{0,1}$, $Q^m_{0,2}$ or $Break(w_0)$. In all the cases these transitions can be matched by R, generating a new pair which belongs again to $\mathscr{R}$:

- $* \ L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid$
  $w_0(x).\nu\rho.\bar{c_1}\,\langle\pi(x)\rangle\,.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(x))) \mid Break(w_0))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1)) \mid$
  $w_1(x).\nu\rho.\bar{c_1}\,\langle\pi(x)\rangle\,.(\bar{t}\langle\_\rangle \mid St(w_1, \sigma(x))) \mid Break(w_1)) \mid$
  $\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))$
- $* \ L = \nu s_0, w_0.(\psi_l \mid !P_{0_1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid$
  $w_2(x).\nu\rho.\bar{c_2}\,\langle\pi(x)\rangle\,.(\bar{t}\langle\_\rangle \mid St(w_2, \sigma(x))) \mid Break(w_0))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1)) \mid Break(w_1)) \mid$
  $\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2))) \mid$
  $w_2(x).\nu\rho.\bar{c_2}\,\langle\pi(x)\rangle\,.(\bar{t}\langle\_\rangle \mid St(w_2, \sigma(x)))$
- $* \ L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid w(x)_0.\overline{br}\langle x\rangle)$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1)) \mid w_1(x).\overline{br}\langle x\rangle) \mid$
  $\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))$

- $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid w_0(x).\overline{br}\langle x\rangle)$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1)) \mid w_1(x).\overline{br}\langle x\rangle) \mid$
  $\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))$
  Since the token is not available, the only possible transition for L is the communication on the channel w_0; this step is matched by R which performs the communication on the channel w_1, obtaining exactly the pair analyzed in the following case.
- $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid \overline{br}\langle\sigma^{k+l+1}(s_0)\rangle)$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid \overline{br}\langle\sigma^k(s_1)\rangle) \mid$
  $\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))$
  As in the previous case, there is only one possible transition for both L and R, namely the sending of the secret on the public channel br. After this transition we obtain the next pair, in which the frames contain also the secret disclosed.
- $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid \{^{\sigma^{k+l+1}(s_0)}/_{x_{k+l}}\})$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid \{^{\sigma^k(s_1)}/_{x_{k+l}}\}) \mid$
  $\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))$
  No transition are possible, since the token is no longer available for any process.
- $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid$
  $w_0(x).\nu\rho.\bar{c_1}\,\langle\pi(x)\rangle\,.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(x))) \mid Break(w_0))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1)) \mid$
  $w_1(x).\nu\rho.\bar{c_1}\,\langle\pi(x)\rangle\,.(\bar{t}\langle\_\rangle \mid St(w_1, \sigma(x))) \mid Break(w_1)) \mid$
  $\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))$
  L can perform only an internal reduction, namely it can synchronize the input and output processes, because the token blocks the processes $Q^n_{0,1}$ and $Q^m_{0,2}$. Again R matches the same reduction and the new processes belong to $\mathscr{R}$:
  $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid$
  $\nu\rho.\bar{c_1}\,\left\langle\pi(\sigma^{k+l}(s_0))\right\rangle\,.(\bar{t}\langle\_\rangle \mid St(w_0, \sigma(\sigma^{k+l+1}(s_0)))) \mid Break(w_0))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid \nu\rho.\bar{c_1}\,\left\langle\pi(\sigma^k(s_1))\right\rangle\,.(\bar{t}\langle\_\rangle \mid$

$St(w_1, \sigma(\sigma^k(s_1)))) \mid Break(w_1)) \mid$
$\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))$

- $L = \nu s_0, w_0.(\psi_l \mid !P_{0_1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid St(w_0, \sigma^{k+l+1}(s_0)) \mid$
  $w_2(x).\nu\rho.\bar{c_2}\,\langle \pi(x) \rangle\,.(\bar{t}\langle \_ \rangle \mid St(w_2, \sigma(x))) \mid Break(w_0))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1)) \mid Break(w_1)) \mid$
  $\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2))) \mid$
  $w_2(x).\nu\rho.\bar{c_2}\,\langle \pi(x) \rangle\,.(\bar{t}\langle \_ \rangle \mid St(w_2, \sigma(x)))$
  This case is similar to the previous one.
- $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid$
  $\nu\rho.\bar{c_1}\,\langle \pi(\sigma^{k+l}(s_0)) \rangle\,.(\bar{t}\langle \_ \rangle \mid St(w_0, \sigma(\sigma^{k+l+1}(s_0)))) \mid Break(w_0))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid \nu\rho.\bar{c_1}\,\langle \pi(\sigma^k(s_1)) \rangle\,.(\bar{t}\langle \_ \rangle \mid$
  $St(w_1, \sigma(\sigma^k(s_1)))) \mid Break(w_1)) \mid$
  $\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))$
  Also in this case, L can perform only a transition, namely it can send $\pi(\sigma^{k+l+1}(s_0))$ on the channel c_1. The new pair will be:
  $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid$
  $\nu\rho.(\bar{t}\langle \_ \rangle \mid St(w_0, \sigma(\sigma^{k+l+1}(s_0)))) \mid Break(w_0))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid \nu\rho.(\bar{t}\langle \_ \rangle \mid$
  $St(w_1, \sigma(\sigma^k(s_1)))) \mid Break(w_1)) \mid$
  $\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid St(w_2, \sigma^l(s_2)))$
  The first process is equivalent to the L process analyzed in the first case of this proof. The only difference is in the frame, which is correctly updated adding a new substitution related to the last output, since k is increased by one; the substitution does not change the form of the frame. Using the same reasoning and the structural equivalence it is possible to map the second process to the corresponding R.
- $L = \nu s_0, w_0.(\psi_l \mid !P_{0,1} \mid !P_{0,2} \mid Q^n_{0,1} \mid Q^m_{0,2} \mid$
  $\nu\rho.\bar{c_2}\,\langle \pi(\sigma^{k+l+1}(s_0)) \rangle\,.(\bar{t}\langle \_ \rangle \mid St(w_0, \sigma(\sigma^{k+l+1}(s_0)))) \mid Break(w_0))$
  $R = \nu s_1, w_1.(\psi_{r1} \mid !P_{1,1} \mid Q^n_{1,1} \mid St(w_1, \sigma^k(s_1)) \mid Break(w_1)) \mid$
  $\nu s_2, w_2.(\psi_{r2} \mid !P_{2,2} \mid Q^m_{2,2} \mid \bar{c_2}\,\langle \pi(\sigma^l(s_2)) \rangle\,.(\bar{t}\langle \_ \rangle \mid St(w_2, \sigma(\sigma^l(s_2)))))$
  This case is similar to the previous one.

Note that in all the above cases we should also consider that the processes L and R might spawn a copy of $Q_{0,1}$ and $Q_{1,1}$ or $Q_{0,2}$ and $Q_{2,2}$ respectively. They trivially belong to the relation, because they only increase the powers of the corresponding processes, thus they keep the same form. Moreover, whenever $Break(w) = br(\_).t(\_).w(x).\overline{br}\langle x \rangle$, the process can be triggered becoming $t(\_).w(x).\overline{br}\langle x \rangle$, but again it is in the same form.
All the frames, with the exception of the last one, are in the following form:

$$\nu s_0.(\psi_l) \approx_s \nu s_1.(\psi_{r_1}) \mid \nu s_2.(\psi_{r_2})$$

This equivalence has been proven for the untraceability property using $\mathcal{P}_1$ and $\mathcal{P}_2$; in this case we assume $\mathcal{P}_1$ and $\mathcal{P}_3$, but we know that $\mathcal{P}_3$ implies $\mathcal{P}_2$, so we can conclude that the static equivalence holds. Thus we need to prove only the static equivalence of the last frame:

$$\nu s_0.(\psi_l \mid \{^{\sigma^{k+l+1}(s_0)}/_{x_{k+l+1}}\}) \approx_s \nu s_1.\nu\rho.(\{\psi_{r_1}\} \mid \{^{\sigma^k(s_1)}/_{x_{k+l}}\}) \mid \nu s_2.\nu\rho.\{\psi_{r_2}\}$$

This equivalence holds because the property $\mathcal{P}_3$ allows us to split the left hand side of the equivalence in two parts, one with only the secret and one with all the other substitutions. Then we apply the same steps seen in the proof for the untraceability property, and finally we merge the first frame and the frame containing the secret only, and the result is the right hand side of the equation.

– ($\neg(\mathcal{P}_1 \wedge \mathcal{P}_3) \Rightarrow \neg$ forward privacy)
  Assumptions: $\neg\mathcal{P}_1 \vee \neg\mathcal{P}_3$
  We want to prove that:

$$\neg(\nu w.\nu s.(!P(w,c_1) \mid !P(w,c_2) \mid St(w,s)) \mid \bar{t}\langle\_\rangle$$
$$\approx_l$$
$$\nu w.\nu s.(!P(w,c_1) \mid St(w,s)) \mid \nu w.\nu s.(!P(w,c_2) \mid St(w,s)) \mid \bar{t}\langle\_\rangle)$$

  Cases:
  1. ($\neg\mathcal{P}_1$)
     See untraceability proof.
  2. ($\neg\mathcal{P}_3$)
     We know that $\mathcal{P}_3$ does not hold. This means that exists n such that:
     $$\neg(\textstyle\prod_{i=0}^{n-1} \rho_{x_i}\pi(\sigma^i(s)) \perp_s \{^{\sigma^n(s)}/_{x_n}\} \; \forall n \in \mathbb{N} \text{ and } \forall \text{ secret } s)$$
     Cases:
     (a) ($\neg\mathcal{P}_1$)
         Forward privacy does not hold by (1).
     (b) ($\mathcal{P}_1$)
         We break the frame equivalence querying n times the first interface and once the second. We obtain the following frames:
         i. $\nu s_0.(\rho_{x_0}\pi(s) \mid \ldots \mid \rho_{x_{n-1}}\pi(\sigma^{n-1}(s)) \mid \rho_{x_n}\sigma^n(s))$
         ii. $\nu s_1.(\rho_{x_0}\pi(s) \mid \ldots \mid \rho_{x_{n-1}}\pi(\sigma^{n-1}(s))) \mid \nu s_2.\rho_{x_n}\sigma^n(s)$
             $\equiv \nu s_1.(\rho_{x_0}\pi(s) \mid \ldots \mid \rho_{x_{n-1}}\pi(\sigma^{n-1}(s))) \mid \nu s_2.\rho_{x_n}\sigma^n(s)$ by $\mathcal{P}_1$
         By $\mathcal{P}_3$ we know that the frames are not statically equivalent, and this breaks the labelled bisimilarity.

□

Proposition 4 The OSK protocol satisfies properties $\mathcal{P}_1, \mathcal{P}_3$, namely:

$\mathcal{P}_1$	$\nu s.\{^{\mathbf{g}(\mathbf{h}^n(s))}/_x\} \approx_s \nu s.\{^{\mathbf{g}(\mathbf{h}^m(s))}/_x\}$	$\forall n, m \in \mathbb{N}$
$\mathcal{P}_3$	$\prod_{i=0}^{n-1}\{^{\mathbf{g}(\mathbf{h}^i(s))}/_{x_i}\} \perp_s \{^{\mathbf{h}^n(s)}/_y\}$	$\forall n \in \mathbb{N}$

Proof. $\mathcal{P}_3$: $\mathbf{h}, \mathbf{g}$ are assumed to be one-way hash functions. Then it is easy to see that for $i < n$, no subterm of $\mathbf{g}(\mathbf{h}^i(s))$ is equal to $\mathbf{h}^n(s)$. This follows from the definition of hash function and the fact that a subterm of $\mathbf{g}(\mathbf{h}^i(s))$ contains at most $n-1$ occurrences of $\mathbf{h}$. Similarly no subterm of $\mathbf{h}^n(s)$ is equal to $\mathbf{g}(\mathbf{h}^i(s))$ since it does not contain $\mathbf{g}$. Then $\mathcal{P}_3$ follows directly from Theorem 1. □

Proposition 5 The simple hash protocol satisfies properties $\mathcal{P}_1, \mathcal{P}_2$, namely

$$\begin{array}{lll} \mathcal{P}_1 & \nu s.\nu r.\{^{(r,\mathbf{h}(s,r))}/_{x}\} \approx_s \nu s.\nu r.\{^{(r,\mathbf{h}(s,r))}/_{x}\} & \\ \mathcal{P}_2 & \prod_{i=0}^{n-1} \nu r.\{^{(r,\mathbf{h}(s,r))}/_{x_i}\} \perp_s \nu r.\{^{(r,\mathbf{h}(s,r))}/_{x_n}\} & \forall n \in \mathbb{N} \end{array}$$

Proof. $\mathcal{P}_1$ is trivial by reflexivity of $\approx_{\mathrm{s}}$.

For $\mathcal{P}_2$, let $r_1, \ldots, r_n$ be distinct channels. Since $\mathbf{h}(s, r_i) \neq_{\mathrm{E}} \mathbf{h}(s, r_j)$ for $i \neq j$, we have by Theorem 1 that

$$\prod_{i=0}^{n-1} \{^{\mathbf{h}(s,r_i)}/_{y_i}\} \perp_s \{^{\mathbf{h}(s,r_n)}/_{y_n}\}$$

We now have $\mathbf{h}(s, r_i)$ in the exported terms, but we need $(r_i, \mathbf{h}(s, r_i))$. For this, we extend the left-hand side with $\prod_{i=0}^{n-1} \{^{(r_i,y_i)}/_{x_i}\}$ and we restrict the y_i's. Similarly we extend the right-hand side with $\{^{(r_n,y_n)}/_{x_n}\}$ and we restrict y_n. Thus from Prop 2 we get

$$\prod_{i=0}^{n-1} \{^{(r_i,\mathbf{h}(s,r_i))}/_{x_i}\} \perp_s \{^{(r_n,\mathbf{h}(s,r_n))}/_{x_n}\}$$

Finally, again by Prop 2, we can restrict $r_1, \ldots, r_n$ and α-rename all frames to get

$$\prod_{i=0}^{n-1} \nu r.\{^{(r,\mathbf{h}(s,r))}/_{x_i}\} \perp_s \nu r.\{^{(r,\mathbf{h}(s,r))}/_{x_n}\}$$

which is $\mathcal{P}_2$. $\square$