Formal Computational Unlinkability Proofs of RFID Protocols

Hubert Comon*, Adrien Koutsos†,
LSV, CNRS & ENS Paris-Saclay
Cachan, France
e-mail:*comon@lsv.fr, †adrien.koutsos@lsv.fr

***Abstract*—We set up a framework for the formal proofs of RFID protocols in the computational model. We rely on the so-called *computationally complete symbolic attacker model*. Our contributions are:**

1) **To design (and prove sound) axioms reflecting the properties of hash functions (Collision-Resistance, PRF).**
2) **To formalize computational unlinkability in the model.**
3) **To illustrate the method, providing the first formal proofs of unlinkability of RFID protocols, in the computational model.**

I. Introduction

It is important to increase our confidence in the security of protocols. Using formal methods to prove a formal security property is the best way to get a strong confidence. There is however a difficulty: we need not only to specify formally the programs and the security properties, but also the attacker.

One of the most popular attacker model, sometimes called the "Dolev-Yao" attacker, consists in assuming that, in between any message emission and the corresponding reception, the attacker may apply a fixed set of rules modifying the message. In addition, the attacker schedules the communications. More precisely, the messages are terms in a formal algebra and the rules are given by a fixed set of combination abilities, together with a rewrite system specifying how to simplify the terms.

There are several more or less automatic verification tools that rely on such a model. Let us cite ProVerif [8], Tamarin [23] and APTE [11] for instance. Completing a proof with one of these tools will however only prove the security in the corresponding DY model.

Another popular attacker model, the *computational attacker*, gives the same network control to the attacker as in the Dolev-Yao model, but does not limit the attacker computations to the combination of a fixed set of operations: any probabilistic polynomial time computation is possible. More precisely, messages are bitstrings, random numbers are typically bitstrings in $\{0,1\}^\eta$ (where η is the *security parameter*) and the attacker's computation time is bounded by a polynomial in η. This model reflects more accurately a real-world attacker than the Dolev-Yao model, but formal proofs are harder to complete and more error-prone.

There exist several formal verification tools in the computational model. For example EasyCrypt [6] can be used for the construction of provably secure cryptographic primitives, and CryptoVerif [9] and F⋆ [5] have been used for the study of security protocols (e.g. [10], [7]). As expected, such tools are less automatic than the verification tools in the DY model. Using such tools, we may also fail to find a proof while there is one.

We advocate the use of another approach, sketched in [3], [4], which allows to complete formal proofs in the computational model that can be automated and formally checked. This method has many other advantages, some of which are given in these papers: it could be applied, in principle, to more powerful attacker models (such as attackers having access to some side-channels); it can be used to derive "minimal" properties of the primitives that are sufficient to entail the security of the protocol (we will come back to this feature later).

The main technique, which we will recall, is to express the security of a protocol as the unsatisfiability of a formula in first-order logic. The formula contains *axioms*, reflecting the assumed computational properties of the security primitives, and the negation of the security property, applied to terms reflecting the execution(s) of the protocol. This approach is, we believe, simpler than formally specifying computational security games, probabilistic machines and simulations: there is no security parameter, no probabilities, no timing constraints, no Turing machines ... Nonetheless, in case of success, the proof is valid in any model of the axioms, including the computational model.

Compared to EasyCrypt, our logic works at a more abstract level, using only first-order formulas and targeting full automation and a form of completeness (saturating the axioms and the negation of the security property implies that there is an attack).

As it is now, this approach does not provide any quantitative information: the protocol is computationally secure or it is not. We could however extract from a security proof a bound on the success probability of the attacker, depending on its computational power: we would only have to compose the adversary's advantages corresponding to each clause. Also, because the logic does not include the security parameter, we can only prove the security of a number of sessions of the protocol that does not depend on the security parameter (while there is no such limitation in CryptoVerif for instance). Still, it subsumes the symbolic approach, in which the number of sessions is also independent of the security parameter.

A related logic for reachability security properties has been introduced in [3] and is implemented in the prototype tool

SCARY [26]. It has been used for a few experiments. The logic for indistinguishability properties, introduced in [4], is not (yet) implemented. There is only one toy example provided in [4], another one in [2] and a more significant case study developed in [27].

We investigate in this paper the application of this approach to security proofs of RFID protocols, typically proofs of unlinkability. There are a lot of such protocols that appeared in the literature, most of which are very simple, due to the low computing capabilities of a RFID tag: the protocols often use only primitives such as hashing, xoring, pairing and splitting. These protocols have been studied, attacked, patched and automatically proved in the DY model (see for instance [18]). On the computational side, [29] investigates the computational definitions of unlinkability, together with examples of RFID protocols that satisfy (or not) the definitions. There are however very few proofs of security in the computational model and no (up to our knowledge) *formal* security proof. For instance, an RFID protocol is proposed in [21], together with a universally composable (claimed) proof. The proof is however quite informal, and attacks were found on this protocol (see [24]). Admittedly, such attacks can be easily circumvented, but this shows that a formal approach is useful, if not necessary. Similarly, as reported in [19], other RFID protocols that were claimed secure turned out to be broken.

A large fraction of RFID protocols, the so-called *Ultra-lightweight* RFID protocols (e.g. [12], [25]), aim at ensuring only weak security properties, and against passive attackers, because of the strong constraints on the number of gates embedded in the RFID tags. We do not consider such protocols in this paper.

a) Contributions: The contributions of this paper are:

1) To design axioms that reflect security assumptions on the primitives that are used in the RFID protocols (typically hash functions, pseudo-random generators and xor), and to prove their correctness.
2) To express formally the computational unlinkability. There are various definitions; we chose to formalize one of them (from [19]). Most other definitions can be expressed in a similar way. The security property is expressed as the indistinguishability of two sequences of terms. These terms are computed from the protocol specification extended with corruption capabilities. We use a specific technique inspired by the folding of transition systems described in [4].
3) To illustrate the proof technique on two examples taken from [28]: KCL and LAK. As far as we know, all published RFID protocols, that do not rely on encryption, are computationally insecure. This is also the case of these two protocols. We propose small modifications of the protocols, which prevent the known attacks. Some of the modified versions are secure in the DY model. Depending on the assumptions on the primitives, they may however be insecure in the computational model. For instance, if we assume the hash function to be pre-image resistant and one-way, the corrected version of LAK, proved in [18], is not necessarily computationally secure: there might be attacks on both authentication and unlinkability. We actually need a family of keyed hash functions, which satisfies the *pseudo-random functions* (PRF) property. With the appropriate implementation assumptions, we formally prove the security of the two protocols. Up to our knowledge, these are the first formal security proofs of RFID protocols in the computational model.

b) Outline: In Section II, we briefly recall the methodology described in [4] and we propose some axioms for the hashing and exclusive or, that depend on the assumptions on the cryptographic libraries. In Section III we recall the definition of privacy of a RFID protocol given by Juels and Weis in [19], and we show how this property translates in the logic. In Section IV we recall the two protocols KCL and LAK, known attacks on them and formally prove the security of fixed versions of the protocols. We also show that relaxing the assumptions yields some attacks. Finally, in Section V, we show (as expected) that abstracting pseudo-random numbers with random numbers is sound, provided that the seed is not used for any other purpose.

II. THE LOGIC

Our goal is to formally study the protocols in the computational model. In order to do this we follow the directions described in [4]: we specify in a first-order logic what the attacker *cannot do*, which yields a set of axioms A. We also compute from the protocol and the security property a formula $\neg\psi$ expressing that there is an attack on the protocol. We know that if $A \cup \{\neg\psi\}$ is unsatisfiable, then the protocol is secure in any model of A. If, in addition, every axiom is computationally sound, then the protocol is computationally secure.

In this section we recall the first-order (indistinguishability) logic and provide a set of axioms A, some of which are valid in any computational model while others require some security assumptions on the cryptographic primitives.

A. Syntax of the logic

a) Terms: In the logic terms are built on a set of function symbols $\mathcal{F}$, a set of function symbols $\mathcal{G}$ (used to represent the attacker's computations), a set of names $\mathcal{N}$ and a set of variables $\mathcal{X}$.

In the examples that are considered in this paper, $\mathcal{F}$ contains at least the following function symbols:

$$\mathsf{H}, \oplus, \langle \_ , \_ \rangle, \mathsf{EQ}(\_;\_), \mathsf{if}\ \_\ \mathsf{then}\ \_\ \mathsf{else}\ \_, \mathsf{true}, \mathsf{false}, \pi_1, \pi_2, \mathbf{0}$$

Each variable and term has a *sort*, which is either bool, short, or message. Every term of sort bool or short, has also the sort message. The typing rules for function symbols are defined as follows:

- $\langle \_ , \_ \rangle$: message $\times$ message $\to$ message
- true, false $:\to$ bool
- π_1, π_2 : message $\to$ message
- EQ(_; _) : message $\times$ message $\to$ bool
- Names have type short
- In the computational interpretation of the terms, we will only need to xor bitstrings of a fixed length: the length

of the random numbers. We therefore restrict the type of $\oplus$ to: $\mathsf{short} \times \mathsf{short} \to \mathsf{short}$
- if _ then _ else _ has three types:
  $\mathsf{bool} \times \mathsf{message} \times \mathsf{message} \to \mathsf{message}$
  $\mathsf{bool} \times \mathsf{bool} \times \mathsf{bool} \to \mathsf{bool}$
  $\mathsf{bool} \times \mathsf{short} \times \mathsf{short} \to \mathsf{short}$
- In our examples, hash values will have a length equal to the security parameter, so H returns a value of sort short. Furthermore, we need a *keyed* hash function, otherwise we cannot state any computationally sound property over the hash function H: $\mathsf{message} \times \mathsf{short} \to \mathsf{short}$
- $\mathbf{0} :\to \mathsf{short}$

We can also use arbitrary additional symbols in $\mathcal{F}$ if needed. Each term has always a least sort (w.r.t. the ordering $\mathsf{bool}, \mathsf{short} < \mathsf{message}$), which we call the *sort of the term*.

b) Formulas: Atomic formulas aim at representing the indistinguishability of two experiment. They are expressions :

$$u_1, \ldots, u_n \sim v_1, \ldots, v_n$$

where $u_1, \ldots, u_n, v_1, \ldots, v_n$ are terms and, for every i, u_i and v_i have the same sort. More formally, for every n, we are using a predicate symbol $\sim_n$ with $2n$ arguments. Formulas then are obtained by combining atomic formulas with Boolean connectives $\wedge, \to, \vee, \neg$ and (first-order) quantifiers.

c) Syntactic shorthands: Displaying the formulas and axioms, we use the notation $\vec{u}$ for a sequence of terms of the appropriate type. We also use "=" as a syntactic sugar: $u = v$ is a shorthand for $\mathsf{EQ}(u; v) \sim \mathsf{true}$. We may also use logical connectives in the conditions, as a shorthand. For instance "if $b_1 \vee b_2$ then x else y" is a shorthand for "if b_1 then x else if b_2 then x else y".

B. Semantics of the logic

We rely on classical first-order interpretations: function symbols are interpreted as functions on an underlying interpretation domain and predicate symbols are interpreted as relations on this domain.

Though, we assume that the interpretation satisfies the generic axioms of the Figure 1, which do not depend on the actual cryptographic libraries. These axioms are computationally valid, as we will see.

One particular class of interpretations of the logic are the so-called *computational semantics*. We recall here the main features (the complete definition can be found in [4]):

- The domain of a computational model is the set of deterministic polynomial time Turing machines equipped with an input (and working) tape and two random tapes (one for honestly generated random values, the other for random values directly available to the attacker). The length of the machine input is the security parameter η.
- A name $n \in \mathcal{N}$ is interpreted as a machine that, on input of length η, extracts a word of length η from the first random tape ρ_1. Furthermore, different names extract disjoint parts of ρ_1.
- true, false, $\mathsf{EQ}(\_;\_)$, if _ then _ else _, $\oplus$ are always interpreted in the expected way. For instance, if _ then _ else _, takes three machines M_1, M_2, M_3 and returns a machine M such that, on input $w, \rho_1.\rho_2$, M returns $M_2(w, \rho_1, \rho_2)$ if $M_1(w, \rho_1, \rho_2) = 1$ and returns $M_3(w, \rho_1, \rho_2)$ otherwise.
- Other function symbols f in $\mathcal{F}$ (including H and $\langle \_, \_ \rangle$, π_1, π_2) are interpreted as arbitrary deterministic polynomial time Turing machines. Though, we may restrict the class of interpretations using assumptions on the implementations of the primitives. This is discussed in the next section.
- A function symbol $g \in \mathcal{G}$ with n arguments is interpreted as a function $[\![g]\!]$, such that there is a polynomial time Turing machine $\mathcal{A}_g$ such that for every n machines $d_1, \ldots, d_n$ in the interpretation domains, and for every inputs w, ρ_1, ρ_2,

$$[\![g]\!](d_1, \ldots, d_n)(w, \rho_1, \rho_2) = \mathcal{A}_g(d_1(w, \rho_1, \rho_2), \ldots, d_n(w, \rho_1, \rho_2), \rho_2)$$

  In particular, the machine does not use directly the random tape ρ_1.
- The interpretation is lifted to terms: given an assignment σ of the variables of a term t to the domain values, we write $[\![t]\!]^{\sigma}_{\eta,\rho_1,\rho_2}$ the computational interpretation of the term, with respect to η, ρ_1, ρ_2. The assignment σ is omitted when empty. We may also omit the other parameters when they are irrelevant.
- The predicates $\sim$ are interpreted as *computational indistinguishability* $\approx$, defined by:

$$d_1, \ldots, d_n \approx d'_1, \ldots, d'_n$$

  iff, for every deterministic polynomial time Turing machine $\mathcal{A}$,

$$\big|\mathbf{Pr}(\rho_1, \rho_2 : \mathcal{A}(d_1(1^\eta, \rho_1, \rho_2), \ldots, d_n(1^\eta, \rho_1, \rho_2)) = 1) - \mathbf{Pr}(\rho_1, \rho_2 : \mathcal{A}(d'_1(1^\eta, \rho_1, \rho_2), \ldots, d'_n(1^\eta, \rho_1, \rho_2)) = 1)\big|$$

  is negligible when ρ_1, ρ_2 are drawn according to the uniform distribution.

C. Computationally valid axioms

We only consider the purely universal fragment of first-order logic: every variable is implicitly universally quantified. A formula is *computationally valid* if it is valid in all computational interpretations.

Proposition 1. *The axioms displayed in the figure 1 are computationally valid.*

Most of the axioms have already been proven valid elsewhere (for instance in [4], [2]). Only the axioms $\vec{u}, x \oplus \mathsf{n} \sim \vec{u}, y \oplus \mathsf{n}$ and $\mathsf{EQ}(\mathsf{n}; x) = \mathsf{false}$ are new, but their computational validity is quite straightforward to prove (e.g. for every x, the distributions $x \oplus \mathsf{n}$ and $y \oplus \mathsf{n}$ are both uniform distributions). EASYCRYPT is using a similar rule through the rnd tactic.

- $(Refl), (Sym), (Trans)$: $\sim$ is reflexive, symmetric and transitive.
- For all $\vec{u}, \vec{v}, t, t'$ of appropriate types:

$$\vec{u}, t \sim \vec{v}, t \quad \Rightarrow \quad \vec{u}, t, t \sim \vec{v}, t', t' \qquad (Dup)$$

- $(Congr) : =$ is a congruence
- For any permutation π of $1, \ldots, n$ and for all $x_1, \ldots, x_n, y_1, \ldots, y_n$ of appropriate types:

$$x_1 \ldots, x_n \sim y_1, \ldots, y_n \\ \Rightarrow x_{\pi(1)}, \ldots, x_{\pi(n)} \sim y_{\pi(1)}, \ldots, y_{\pi(n)} \qquad (Perm)$$

- For every function symbol f of appropriate type, for all $\vec{x}, \vec{y}, \vec{x'}, \vec{y'}$:

$$\vec{x}, \vec{y} \sim \vec{x'}, \vec{y'} \quad \Rightarrow \quad f(\vec{x}), \vec{y} \sim f(\vec{x'}), \vec{y'} \qquad (FA)$$

- For every $b, x, y, z, \vec{w}$ of appropriate types:

$$\begin{array}{rcl} \mathsf{EQ}(x; x) & = & \mathsf{true} \\ \mathsf{if\ true\ then}\ x\ \mathsf{else}\ y & = & x \\ \mathsf{if\ false\ then}\ x\ \mathsf{else}\ y & = & y \\ \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ x & = & x \\ \mathsf{if}\ b\ \mathsf{then\ if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ y\ \mathsf{else}\ z & = & \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z \\ \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else\ if}\ b\ \mathsf{then}\ y\ \mathsf{else}\ z & = & \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z \end{array}$$

$$\mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ z = \\ \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ y\ \mathsf{else}\ z)$$

$$\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ y\ \mathsf{else}\ z) = \\ \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z)$$

$$f(\vec{x}, \mathsf{if}\ b\ \mathsf{then}\ y\ \mathsf{else}\ z, \vec{w}) = \\ \mathsf{if}\ b\ \mathsf{then}\ f(\vec{x}, y, \vec{w})\ \mathsf{else}\ f(\vec{x}, z, \vec{w})$$

$$\begin{array}{rcl} x \oplus y & = & y \oplus x \\ x \oplus x & = & \mathbf{0} \\ \mathbf{0} \oplus x & = & x \\ x \oplus (y \oplus z) & = & (x \oplus y) \oplus z \end{array}$$

- If n does not occur in $x, y, \vec{u}, \vec{v}$:

$$\vec{u} \sim \vec{v} \Rightarrow \vec{u}, x \oplus \mathsf{n} \sim \vec{v}, y \oplus \mathsf{n} \qquad (Indep)$$

$$\mathsf{EQ}(\mathsf{n}; x) = \mathsf{false} \qquad (EqIndep)$$

- If n does not occur in $\vec{u}$ and $\mathsf{n'}$ does not occur in $\vec{u'}$:

$$\vec{u} \sim \vec{u'} \Rightarrow \vec{u}, \mathsf{n} \sim \vec{u'}, \mathsf{n'} \qquad (FreshNonce)$$

- For every $x, y, z, \vec{w}, \vec{x'}$ of appropriate types and for every context t:

$$\mathsf{if}\ \mathsf{EQ}(x; y)\ \mathsf{then}\ t[x]\ \mathsf{else}\ z, \vec{w} \sim \vec{x'} \\ \Rightarrow \mathsf{if}\ \mathsf{EQ}(x; y)\ \mathsf{then}\ t[y]\ \mathsf{else}\ z, \vec{w} \sim \vec{x'} \qquad (IfThen)$$

- (CS) : For every $b, b', x, x', y, y', z, z', \vec{u}, \vec{u'}$ of appropriate types:

$$b, \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z, \vec{u} \sim b', \mathsf{if}\ b'\ \mathsf{then}\ x'\ \mathsf{else}\ z', \vec{u'} \\ \wedge b, \mathsf{if}\ b\ \mathsf{then}\ z\ \mathsf{else}\ y, \vec{u} \sim b', \mathsf{if}\ b'\ \mathsf{then}\ z'\ \mathsf{else}\ y', \vec{u'} \\ \Rightarrow \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ y, \vec{u} \sim \mathsf{if}\ b'\ \mathsf{then}\ x'\ \mathsf{else}\ y', \vec{u'}$$

Fig. 1. Axioms that are independent of the cryptographic primitives

D. Assumptions on primitives

Some of our axioms reflect implementation assumptions, that is, identities that must be satisfied by any implementation of these functions. For example we will assume that the projections of a pair return the components of the pair:

$$\pi_1(\langle x\ ,\ y\rangle) = x \qquad \pi_2(\langle x\ ,\ y\rangle) = y$$

We do not make any further assumption on the implementation of pairing.

Other axioms reflect cryptographic assumptions on the primitives. For example, for hash functions, we need to express some computational hardness properties. An example of such a property is the Collision Resistance property, that we recall below:

Definition 1 (CR-HK [17])**.** A hash function H is said to be *collision resistant under hidden-key attacks* if and only if for all probabilistic polynomial time adversary $\mathcal{A}$ with access to an oracle returning the keyed hash value of the request,

$$\mathbf{Pr}(\mathsf{k}, \rho :\ \mathcal{A}^{\mathsf{H}(\cdot,\mathsf{k})}(1^\eta) = \langle m_1\ ,\ m_2\rangle \wedge \mathsf{H}(m_1, \mathsf{k}) = \mathsf{H}(m_2, \mathsf{k}))$$

is negligible, where k is drawn uniformly at random in $\{0,1\}^\eta$ and ρ is the random tape of the attacker.

This property can be expressed in the logic by the following CR axiom:

If the only occurrences of k in $t, t', \vec{u}$ are as a second argument of H:

$$\vec{u}, \mathsf{if}\ \mathsf{EQ}(t; t')\ \mathsf{then\ false\ else}\ \mathsf{EQ}(\mathsf{H}(t, \mathsf{k}); \mathsf{H}(t', \mathsf{k})) \\ \sim \\ \vec{u}, \mathsf{false}$$

Remark 1. Note that we need the conditional here: we cannot simply state that when t and t' are distinct,

$$\mathsf{EQ}(\mathsf{H}(t, \mathsf{k}); \mathsf{H}(t', \mathsf{k})) \sim \mathsf{false}$$

For instance, take $t = g(u)$ and $t' = g(u')$ where u, u' are distinct and g is an attacker's function. Then even though t and t' are syntactically distinct, the function g can be interpreted as a function that discards its argument and always return the same value. In such a case, the computational interpretations of t and t' are identical.

Proposition 2. *The CR axiom is sound in any computational model $\mathcal{M}_c$ where the hash function H is collision resistant under hidden-key attacks.*

Proof: Let b_{CR} be the following term:

$$\mathsf{if}\ \mathsf{EQ}(t; t')\ \mathsf{then\ false\ else}\ \mathsf{EQ}(\mathsf{H}(t, \mathsf{k}); \mathsf{H}(t', \mathsf{k}))$$

Let $\mathcal{M}_c$ be a computational model such that H is interpreted by a collision-resistant keyed hash function, and assume that there exists an adversary $\mathcal{A}$ such that:

$$\left|\mathbf{Pr}\left(\rho : \mathcal{A}([\![b_{CR}]\!]_{\eta,\rho})\right) - \mathbf{Pr}\left(\rho : \mathcal{A}([\![\mathsf{false}]\!]_{\eta,\rho})\right)\right|$$

is not negligible in η. Since b_{CR} is a boolean term, $[\![b_{CR}]\!]_{\eta,\rho} \in \{\mathsf{true}, \mathsf{false}\}$, hence the existence of $\mathcal{A}$ is equivalent to

$\mathbf{Pr}\,(\rho : [\![b_{CR}]\!]_{\eta,\rho} = \mathsf{true})$ is non-negligible

We are going to define a probabilistic polynomial time adversary with access to an hash oracle with a non negligible chance of winning the CR-HK game.

Since the only occurrences of k in $t, t', \vec{u}$ are as a second argument of H, we can define $\mathcal{A}'$ to be the CR-HK-adversary that computes and returns $[\![t]\!]_{\eta,\rho}$ and $[\![t']\!]_{\eta,\rho}$ (names different from k are computed by uniform sampling, and subterms of the form $\mathsf{H}(u,\mathsf{k})$ are computed by calling the hash oracle on the inductively computed $[\![u]\!]_{\eta,\rho}$). Then we have:

$$\begin{aligned}
& \mathbf{Pr}\big(\mathsf{k} :\ \mathcal{A}'^{\mathsf{H}(\cdot,\mathsf{k})}() = \langle m_1\,,\, m_2\rangle \\
& \qquad\qquad \wedge \mathsf{H}(m_1,\mathsf{k}) = \mathsf{H}(m_2,\mathsf{k}) \wedge m_1 \neq m_2\big) \\
= \ & \mathbf{Pr}\big(\rho :\ \langle [\![t]\!]_{\eta,\rho}\,,\, [\![t']\!]_{\eta,\rho}\rangle = \langle m_1\,,\, m_2\rangle \wedge \\
& \qquad \mathsf{H}(m_1, [\![\mathsf{k}]\!]_{\eta,\rho}) = \mathsf{H}(m_2, [\![\mathsf{k}]\!]_{\eta,\rho}) \wedge m_1 \neq m_2\big) \\
= \ & \mathbf{Pr}\big(\rho :\ \mathsf{H}([\![t]\!]_{\eta,\rho}, [\![\mathsf{k}]\!]_{\eta,\rho}) = \mathsf{H}([\![t]\!]_{\eta,\rho}, [\![\mathsf{k}]\!]_{\eta,\rho}) \\
& \qquad\qquad \wedge [\![t]\!]_{\eta,\rho} \neq [\![t']\!]_{\eta,\rho}\big) \\
= \ & \mathbf{Pr}\big(\rho :\ [\![\mathsf{H}(t,\mathsf{k})]\!]_{\eta,\rho} = [\![\mathsf{H}(t',\mathsf{k})]\!]_{\eta,\rho} \wedge [\![t]\!]_{\eta,\rho} \neq [\![t']\!]_{\eta,\rho}\big) \\
= \ & \mathbf{Pr}\big(\rho :\ [\![b_{CR}]\!]_{\eta,\rho} = \mathsf{true}\big)
\end{aligned}$$

which we assumed to be non negligible in η. ∎

Unfortunately, this property is not sufficient to prove the security of the RFID protocols considered in this paper (we will show attacks when only CR-HK is assumed). Therefore we need to consider a stronger property:

Definition 2 (PRF[15], [16])**.** A family of keyed hash functions: $H(\cdot,k) : \{0,1\}^* \to \{0,1\}^\eta$ is a *Pseudo Random Function* family if, for any PPT adversary $\mathcal{A}$ with access to an oracle $\mathcal{O}_f$:

$$|\mathbf{Pr}(k,\rho :\ \mathcal{A}^{\mathcal{O}_{H(\cdot,k)}}(1^\eta) = 1) - \mathbf{Pr}(g,\rho :\ \mathcal{A}^{\mathcal{O}_{g(\cdot)}}(1^\eta) = 1)|$$

is negligible. Where

- k is drawn uniformly in $\{0,1\}^\eta$.
- ρ is the random tape of the attacker $\mathcal{A}$ (drawn uniformly).
- g is drawn uniformly in the set of all functions from $\{0,1\}^*$ to $\{0,1\}^\eta$.

We express this property in the logic using the following recursive schema of axioms:

$$\vec{u}, \mathsf{if}\ c\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ \mathsf{H}(t,k) \sim \vec{u}, \mathsf{if}\ c\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ \mathsf{n} \quad (PRF_n)$$

where:

- the occurrences of H (and k) in $\vec{u}, t$ are $\mathsf{H}(t_1,\mathsf{k}), \ldots, \mathsf{H}(t_n,\mathsf{k})$
- n is a name, that does not occur in $\vec{u}, t, c$
- $c \equiv \bigvee_{i=1}^{n} \mathsf{EQ}(t_i; t)$.

Proposition 3. *For any n, (PRF_n) is computationally sound if $\mathsf{H}(\cdot,k)$ is a PRF family.*

The proof can be found in [13], and is a little more complicated than the previous one. Note that k cannot appear in t_i itself, unless $\mathsf{H}(t_j,\mathsf{k})$ is a subterm of t_i for some j; the statement of the axiom does not cover "key cycles" such as in the expression $\mathsf{H}(\mathsf{k},\mathsf{k})$. Extending the axiom to cover such situations would require a Random Oracle assumption for the hash function.

$$\begin{array}{lcl} R & : & \mathsf{n}_R \xleftarrow{\$} \\ T_\mathsf{A} & : & \mathsf{n}_T \xleftarrow{\$} \end{array}$$

$$\begin{array}{lcl} 1 : R \longrightarrow T_\mathsf{A} & : & \mathsf{n}_R \\ 2 : T_\mathsf{A} \longrightarrow R & : & \langle \mathsf{A} \oplus \mathsf{n}_T\,,\, \mathsf{n}_T \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k}_\mathsf{A})\rangle \end{array}$$

Fig. 2. The original KCL protocol

III. Security properties

Radio Frequency IDentification (RFID) systems allow to wirelessly identify objects. These systems are composed of *readers* and *tags*. Readers are radio-transmiters connected through a secure channel to a single server hosting a database with all the tracked objects information. Tags are wireless transponders attached to physical objects that have a limited memory and computational capacity (so as to reduce costs). For the sake of simplicity we will assume that there is only one reader, which will represent the database server as well as all the physical radio-transmitters.

Example 1. In order to illustrate the properties, we introduce a very simple version of the protocol KCL. The original protocol from [20] is (informally) described in Figure 2. The key k_A is a shared secret key between the tag T_A and the reader R. For simplicity we assume, for now, that names $\mathsf{n}_T, \mathsf{n}_R$ are randomly generated by each party ($\xleftarrow{\$}$ is used to denote random sampling); this will be justified in Section V. The protocol is expected to ensure both authentication and unlinkability.

A. Privacy of RFID protocols

We use the notion of Privacy for RFID protocols defined in [19], that we informally recall here. This is a game-based definition, in which the adversary is a probabilistic polynomial time Turing machine and interacts with a reader R and a finite set of tags $\{T_1, \ldots, T_n\}$ (also probabilistic Turing machines) through fixed communication interfaces, which are informally described below and in Figure 3:

- A tag T_i can store a secret key k_i, an id Id_i, a session identifier sid and the previous l challenge-response pairs of the current session. It has the following interface:
  - SetKey: Corrupts the tag by returning its old key k_i and id Id_i, and allows the adversary to send a new key k'_i and a new id Id'_i of its choice.
  - TagInit: Initialize a tag with a session identifier sid'. The tag deletes the previous session identifier and the logged challenge-response pairs.
  - TagMsg: The tag receives a challenge c_i and returns a response r_i (that was computed using the key, the session identifier and the logged challenge-response pairs). Additionally the tag logs the challenge-response pair (c_i, r_i).
- The reader R stores some private key material (for example a master secret key, the tags private keys ...) and entries of the form $(sid, status, c_0, r_0, \ldots, c_l)$ where $status$ is either $open$ or $closed$ depending on whether the

session is completed or on-going. It has the following interface:
  - READERINIT: Returns a fresh session identifier sid along with the first challenge c_o. The reader also stores a new entry of the form $(sid, open, c_o)$.
  - READERMSG: The reader receives a session identifier sid and a response r_i. It looks for a data entry of the form $(sid, open, c_0, r_0, \ldots, c_i)$, appends the message r_i to the data entry, and either closes the session (by changing the status from $open$ to $closed$) and outputs a new challenge message c_{i+1} (possibly 0) and appends it to the data entry.

The adversary is allowed to corrupt (by a SETKEY command) up to $n-2$ tags. At the end of a first phase of computations and interactions with the reader R and tags $\{T_1, \ldots, T_n\}$, the adversary chooses two uncorrupted tags T_{i_0} and T_{i_1}, which are removed from the set of available tags. Then one of these tags is chosen uniformly at random by sampling a bit b and made accessible to the adversary as an oracle. The adversary performs a second phase of computations and interactions with the reader R, the tags $\{T_1, \ldots, T_n\} \backslash \{T_{i_0}, T_{i_1}\}$, as well as the randomly selected tag T_{i_b} (obviously the adversary is not allowed to corrupt T_{i_b}). Finally the adversary outputs a bit b', and wins if it guessed the chosen tag (that is if $b = b'$). A protocol is said to verify m-Privacy if any adversary $\mathcal{A}$ using at most m calls to the interfaces, has a probability of winning the game bounded by $\frac{1}{2} + f_{\mathcal{A}}(\eta)$, where $f_{\mathcal{A}}$ is a negligible function in the security parameter.

Remark 2. Our definition of privacy is slightly different from the one in [19]:

- We do not assume that the reader answers "reject" or "accept" when a session is completed. We can easily encode this feature by adding an answer from the reader at the end of the protocol with the corresponding message. Not taking this as the default behavior allows to model adversaries that are less powerful and do not have access to the result of the protocol.
- We use m-Privacy, whereas they use (r, s, t)-Privacy where r and s are a bound on the number of calls to READERINIT and TAGINIT respectively, and t is a bound on the running time. We dropped the explicit mention of t as we are only interested in proving privacy against any polynomial time adversary. Moreover using m or r, s is equivalent, as, for a given protocol, the number of calls to the interfaces is bounded by the number of calls to READERINIT and to TAGINIT, and conversely.

B. Bounded Session Privacy

a) Protocol Description: We let Γ_n be the set of possible actions of an adversary interacting with n tags:

$$\Gamma_n = \{\text{SETKEY}_i, \text{TAGINIT}_i, \text{TAGMSG}_i, \text{READERINIT}, \text{READERMSG} \mid 1 \leq i \leq n\}$$

We are going to use a substitution σ to save the state of the reader R and the tags $T_1, \ldots, T_n$ internal memories. For instance, for each tag T_i, we use two locations $x_{\mathsf{k}_i}, x_{Id_i}$ to store respectively the values of the key and of the id.

For every adversarial frame ϕ, for every action $\alpha \in \Gamma$ and for every internal memory state σ we can build from the protocol description a term $t^\sigma_\alpha(\phi)$, which represents the answer of the reader or tag to the action α. We also build a internal memory update $\theta^\sigma_\alpha(\phi)$ such that $\sigma\,\theta^\sigma_\alpha(\phi)$ represents the updated memory of the reader and tags after the action has been performed. We also build from the protocol description the initial internal memory σ_{init}.

For the sake of simplicity, we assume that all names appearing in a term $t^\sigma_\alpha(\phi)$ are fresh. If one wants to reuse a name (e.g. in the LAK protocol the reader uses the same name in the first and second challenge), then we will store it in the internal memory σ.

b) Folding of the Protocol: Given a subset of actions S of Γ_n, we construct a term $t^\sigma_S(\phi)$ and a substitution $\theta^\sigma_S(\phi)$ representing (respectively) the message sent over the network and the memory update, depending on the action chosen by the attacker in S (under memory σ and frame ϕ). Intuitively, such terms gather together all possible interleavings of actions using the folding technique explained in [4].

Given an (arbitrary) enumeration of $S = \{\alpha_1, \ldots, \alpha_r\}$, $t^\sigma_S(\phi)$ and $\theta^\sigma_S(\phi)$ are defined using intermediate terms $u^\sigma_i(\phi)$ and $\tau^\sigma_i(\phi)$ (for all $i \in \{1, \ldots, r\}$), relying on an attacker function symbol $to \in \mathcal{G}$ (representing the scheduling choice of the attacker):

$$u^\sigma_1(\phi) = t^\sigma_{\alpha_1}(\phi) \qquad \tau^\sigma_1(\phi) = \theta^\sigma_{\alpha_1}(\phi)$$
$$u^\sigma_{i+1}(\phi) = \mathsf{if}\ \mathsf{EQ}(to(\phi); \alpha_{i+1})\ \mathsf{then}\ t^\sigma_{\alpha_{i+1}}(\phi)\ \mathsf{else}\ u^\sigma_i(\phi)$$
$$\tau^\sigma_{i+1}(\phi) = \mathsf{if}\ \mathsf{EQ}(to(\phi); \alpha_{i+1})\ \mathsf{then}\ \theta^\sigma_{\alpha_{i+1}}(\phi)\ \mathsf{else}\ \tau^\sigma_i(\phi)$$

where $\mathsf{if}\ b\ \mathsf{then}\ \theta_1\ \mathsf{else}\ \theta_2$ denotes the substitution θ defined by: for every variable x, $x\theta = \mathsf{if}\ b\ \mathsf{then}\ x\theta_1\ \mathsf{else}\ x\theta_2$.

We then let $t^\sigma_S(\phi) = u^\sigma_r(\phi)$ and $\theta^\sigma_S(\phi) = \tau^\sigma_r(\phi)$.

Example 2. Let us return to the example 1.

Each tag T_{A_i} has an identifier A_i and a key k_{A_i}. In the KCL protocol the TAGINIT_i call is useless because the tag has only one message to send in a round of the protocol (TAGINIT_i is used to tell a tag to stop the current round of the protocol and to start a new one). Since we consider a finite number n of interface calls, there is at most n sessions executed in parallel. We will use a variable nb to store the index of the next session to start (initialized to 1), and variables $c^1_0, \ldots, c^n_0$ and $c^1_1, \ldots, c^n_1$ (initialized to 0) where c^i_0 and c^i_1 store respectively the first challenge and the second challenge of session i.

- $t^\sigma_{\text{SETKEY}_i}(\phi) = \langle \sigma(x_{\mathsf{k}_i})\,,\, \sigma(x_{Id_i}) \rangle$: the data of the tag i are disclosed.
- $\theta^\sigma_{\text{SETKEY}_i}(\phi) = \{x_{\mathsf{k}_i} \mapsto g_{\text{KEY}_i}(\phi), x_{Id_i} \mapsto g_{\text{ID}_i}(\phi)\}$: the key and id of the tag i are set to values chosen by the attacker ($g_{\text{KEY}_i}, g_{\text{ID}_i} \in \mathcal{G}$).
- $t^\sigma_{\text{TAGMSG}_i}(\phi) =$

$$\langle \sigma(x_{Id_i}) \oplus \mathsf{n}_T\,,\, \mathsf{n}_T \oplus \mathsf{H}(g_{\text{TMSG}_i}(\phi), \sigma(x_{\mathsf{k}_i})) \rangle$$

  The reply of the tag i follows the protocol, according to its local store. $g_{\text{TMSG}_i}(\phi)$ is the message, forged by the adversary, replacing the expected name n_R; $g_{\text{TMSG}_i} \in \mathcal{G}$.

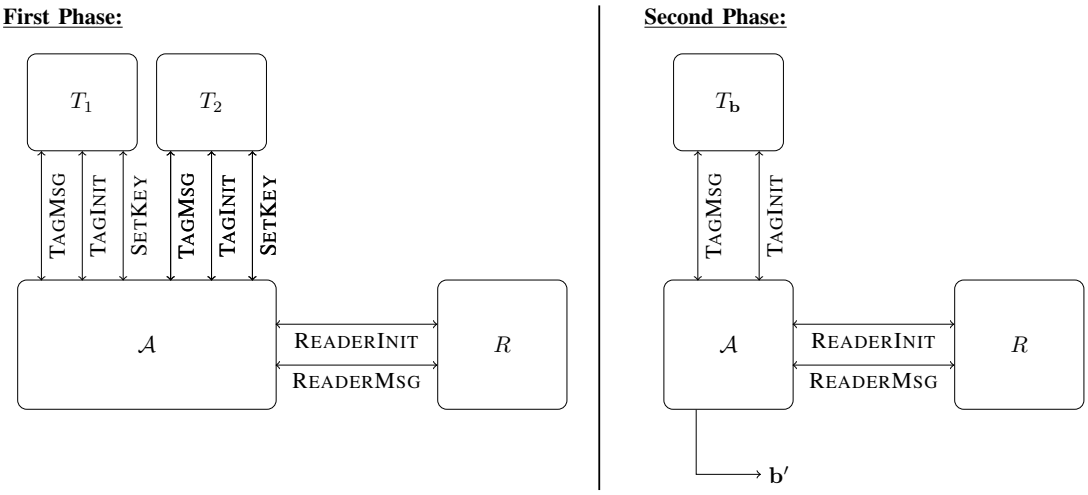

Fig. 3. Privacy game with two tags T_1, T_2. The adversary $\mathcal{A}$ wins if $\mathbf{b} = \mathbf{b}'$.

- $\theta^{\sigma}_{\text{TagMsg}_i}(\phi) = \epsilon$: there is no update in this case (nothing is stored for further verifications in this particular protocol)
- $t^{\sigma}_{\text{ReaderInit}}(\phi) = \langle \sigma(\mathsf{nb}), \mathsf{n}_R \rangle$: when starting a new session, the reader sends a new challenge n_R
- $\theta^{\sigma}_{\text{ReaderInit}}(\phi) =$

$$\begin{cases} \mathsf{nb} \mapsto (\sigma(\mathsf{nb}) + 1) \\ c^j \mapsto \mathsf{if}\ \mathsf{EQ}(\sigma(\mathsf{nb}); j)\ \mathsf{then}\ \mathsf{n}_R\ \mathsf{else}\ \sigma(c^j) \mid 1 \le j \le n \end{cases}$$

  The local memory of the reader is updated.

We now express the Privacy game as a set of equivalence properties. After the first phase, once the attacker has chosen the corrupted tags, we rename the tags in such a way that the challenged tags are T_{n-1} and T_n. In the definition below, p is the number of interactions of the adversary during the first phase and q is the number of interactions of the adversary during the second phase.

Definition 3 (m-Bounded Session Privacy)**.** Given a (computational) interpretation I of the function symbols in $\mathcal{F}$, a protocol satisfies *m-Bounded Session Privacy* if for every p, q such that $p + q = m$ and for every computational model $\mathcal{M}_c^I$ that extends I, we have:

$$\mathcal{M}_c^I \models (t^{\sigma_i}(\phi_i))_{i \le m}, g_{\text{guess}}(\phi_{m+1}) \sim (t^{\tilde{\sigma}_i}(\tilde{\phi}_i))_{i \le m}, g_{\text{guess}}(\tilde{\phi}_{m+1})$$

where $\phi_1 = \epsilon$, $\sigma_1 = \sigma_{\text{init}}$ and for all $1 < i \le m$:

- $\phi_{i+1} = \begin{cases} \phi_i, t^{\sigma_i}_{\Gamma_n}(\phi_i) \text{ if } i \le p \\ \phi_i, t^{\sigma_i}_{\Gamma_{n-1} \setminus \{\text{SetKey}_{n-1}\}}(\phi_i) \text{ if } i \ge p \end{cases}$
- $\sigma_{i+1} = \begin{cases} \sigma_i\, \theta^{\sigma_i}_{\Gamma_n}(\phi_i) & (\text{ if } i \le p) \\ \sigma_i \{x_{\mathsf{k}_{n-1}} \mapsto \sigma_i(x_{\mathsf{k}_{n-1}})\}\, \theta^{\sigma_i}_{\Gamma_{n-1} \setminus \{\text{SetKey}_{n-1}\}}(\phi_i) & \\ & (\text{ if } i = p+1) \\ \sigma_i\, \theta^{\sigma_i}_{\Gamma_{n-1} \setminus \{\text{SetKey}_{n-1}\}}(\phi_i) & (\text{ if } i > p+1) \end{cases}$

$\tilde{\phi}_i$ and $\tilde{\sigma}_i$ are defined similarly, except that the chosen tag is T_n and not T_{n-1} (that is the key substitution used is $\{x_{\mathsf{k}_{n-1}} \mapsto \tilde{\sigma}_i(x_{\mathsf{k}_n})\}$ instead of $\{x_{\mathsf{k}_{n-1}} \mapsto \sigma_i(x_{\mathsf{k}_{n-1}})\}$ in the case $i = p + 1$). $g_{\text{guess}} \in \mathcal{G}$ is the attacker's function guessing with which tag the interaction took place.

Theorem 1. *Given a computational interpretation of function symbols in $\mathcal{F}$, a protocol P satisfies m-Bounded Session Privacy iff it satisfies m-Privacy.*

Proof sketch: If there is an attacker on m-privacy, we can construct an interpretation of the attacker's function symbols that yields a counter-model of the equivalence property. Conversely, if there is a counter-model we can build an attacker that performs the actions and computations specified by the interpretations of these function symbols. These constructions correspond basically to the proofs in [4]. ■

C. Fixed Trace Privacy

The m-Bounded Session Privacy definition is a bit cumbersome, since the terms gather together all possible trace interleavings of the protocol with n tags and m calls to the interfaces. It is easier to use the following definition, that we call m-Fixed Trace Privacy, that considers equivalence formulas between executions of the protocol with a fixed sequence of actions chosen by the adversary. Basically we split a big equivalence into an exponential number (in m) of smaller formulas.

Definition 4 (m-Fixed Trace Privacy)**.** Given an interpretation I of function symbols in $\mathcal{F}$, a protocol satisfies m-Fixed Trace Privacy if for all p, q such that $p+q = m$, for all $(\alpha_i)_{1 \le i \le m} \in (\Gamma_n)^p \times (\Gamma_{n-1} \setminus \text{SetKey}_{n-1})^q$, for all computational models $\mathcal{M}_c$ that extend I, we have:

$$\mathcal{M}_c \models (t^{\sigma_i}_{\alpha_i}(\phi_i))_{i \le m}, g_{\text{guess}}(\phi_{m+1}) \sim (t^{\tilde{\sigma}_i}_{\alpha_i}(\tilde{\phi}_i))_{i \le m}, g_{\text{guess}}(\tilde{\phi}_{m+1})$$

where $\phi_1 = \epsilon$, $\sigma_1 = \sigma_{\text{init}}$ and for all $1 < i \le m$:

- $\phi_{i+1} = \phi_i, t^{\sigma_i}_{\alpha_i}(\phi_i)$
- $\tilde{\phi}_{i+1} = \tilde{\phi}_i, t^{\tilde{\sigma}_i}_{\alpha_i}(\tilde{\phi}_i)$
- $\sigma_{i+1} = \begin{cases} \sigma_i\, \theta^{\sigma_i}_{\alpha_i}(\phi_i) \text{ if } i \ne p+1 \\ \sigma_i \{x_{\mathsf{k}_{n-1}} \mapsto \sigma_i(x_{\mathsf{k}_{n-1}})\}\, \theta^{\sigma_i}_{\alpha_i}(\phi_i) \text{ if } i = p+1 \end{cases}$

- $\tilde{\sigma}_{i+1} = \begin{cases} \tilde{\sigma}_i\, \theta^{\tilde{\sigma}_i}_{\alpha_i}(\tilde{\phi}_i) \text{ if } i \neq p+1 \\ \tilde{\sigma}_i\, \{x_{\mathsf{k}_{n-1}} \mapsto \tilde{\sigma}_i(x_{\mathsf{k}_n})\}\, \theta^{\tilde{\sigma}_i}_{\alpha_i}(\tilde{\phi}_i) \text{ if } i = p+1 \end{cases}$

Proposition 4. *m-Fixed Trace Privacy is equivalent to m-Bounded Session Privacy.*

Proof: We start by showing that m-Bounded Session Privacy implies m-Fixed Trace Privacy by contraposition: let p, q be two integers such that $p+q = m$, $(\alpha_i)_{1\leq i\leq m} \in (\Gamma_n)^p \times (\Gamma_{n-1} \backslash \textsc{SetKey}_{n-1})^q$ and $\mathcal{M}_c$ be a computational model such that:

$$\mathcal{M}_c \models (t^{\sigma_i}_{\alpha_i}(\phi_i))_{i\leq m}, g_{\text{guess}}(\phi_{m+1}) \not\sim (t^{\tilde{\sigma}_i}_{\alpha_i}(\tilde{\phi}_i))_{i\leq m}, g_{\text{guess}}(\tilde{\phi}_{m+1})$$

Let $\mathcal{A}$ be an adversary with a non negligible probability of distinguishing between the two distributions (interpreted in $\mathcal{M}_c$). We define the model $\mathcal{M}'_c$ to be $\mathcal{M}_c$ with a signature extended by the function symbols $\{to\} \cup \Gamma_n$: for all $\alpha \in \Gamma_n$, $[\![\alpha]\!]_{\mathcal{M}'_c}$ is interpreted by a machine returning α on all input, and the function symbol to is interpreted by a machine satisfying:

$$[\![to(\phi_i)]\!]_{\mathcal{M}'_c} = [\![to(\tilde{\phi}_i)]\!]_{\mathcal{M}'_c} = [\![\alpha_i]\!]_{\mathcal{M}'_c}$$

This only depends on the size of ϕ_i and $\tilde{\phi}_i$, and is therefore possible to do with a polynomial time probabilistic Turing Machine, hence $\mathcal{M}'_c$ is a computational model. Moreover by construction $\mathcal{A}$ has a non negligible probability of breaking m-Bounded Session Privacy in $\mathcal{M}'_c$, which concludes the proof.

Conversely we show that m-Fixed Trace Privacy implies m-Bounded Session Privacy by contraposition: using notation of Definition 3, let p, q be such that $p+q = m$ and assume a model $\mathcal{M}_c$ and an adversary $\mathcal{A}$ with a non negligible probability of distinguishing between $[\![\phi_{m+1}]\!]$ and $[\![\tilde{\phi}_{m+1}]\!]$ (we remove $[\![g_{guess}(\phi_{m+1})]\!]$) since $\mathcal{A}$ can compute it himself). We let $S = (\Gamma_n)^p \times (\Gamma_{n-1} \backslash \textsc{SetKey}_{n-1})^q$ and for all $\vec{\alpha} \in S$, $E_{\vec{\alpha}}$ is the event $\bigwedge_i [\![to(\phi_i)]\!]_\eta = \alpha_i$. We then have:

$$\begin{aligned} &\mathbf{Pr}_\rho(\mathcal{A}([\![\phi_{m+1}]\!]_{\eta,\rho}) = 1) \\ &= \sum_{\vec{\alpha}\in S} \mathbf{Pr}_\rho(\mathcal{A}([\![\phi_{m+1}]\!]_{\eta,\rho}) = 1 \mid E_{\vec{\alpha}}) \times \mathbf{Pr}_\rho(E_{\vec{\alpha}}) \\ &= \sum_{\vec{\alpha}\in S} \mathbf{Pr}_\rho(\mathcal{A}([\![(t^{\sigma_i}_{\alpha_i}(\psi_i^{\vec{\alpha}}))_{i\leq m}]\!]_{\eta,\rho}) = 1) \times \mathbf{Pr}_\rho(E_{\vec{\alpha}}) \end{aligned}$$

where $\psi_i^{\vec{\alpha}}$ denotes the i-th frame obtained for the fixed trace $\vec{\alpha}$ (it is the ϕ_i from Definition 4 renamed to avoid notations clash). Similarly we let $E'_{\vec{\alpha}}$ is the event $\bigwedge_i [\![to(\tilde{\phi}_i)]\!]_\eta = \alpha_i$ and we have:

$$\begin{aligned} &\mathbf{Pr}_\rho(\mathcal{A}([\![\tilde{\phi}_{m+1}]\!]_{\eta,\rho}) = 1) \\ &= \sum_{\vec{\alpha}\in S} \mathbf{Pr}_\rho(\mathcal{A}([\![(t^{\tilde{\sigma}_i}_{\alpha_i}(\tilde{\psi}_i^{\vec{\alpha}}))_{i\leq m}]\!]_{\eta,\rho}) = 1) \times \mathbf{Pr}_\rho(E'_{\vec{\alpha}}) \end{aligned}$$

Since $\mathbf{Pr}_\rho(E_{\vec{\alpha}}) \leq 1$, $\mathbf{Pr}_\rho(E'_{\vec{\alpha}}) \leq 1$ and bounding the absolute value of the sum by the sum of the absolute values we get that:

$$\left|\mathbf{Pr}_\rho(\mathcal{A}([\![\phi_{m+1}]\!]_{\eta,\rho}) = 1) - \mathbf{Pr}_\rho(\mathcal{A}([\![\tilde{\phi}_{m+1}]\!]_{\eta,\rho}) = 1)\right|$$

is upper bounded by:

$$\sum_{\vec{\alpha}\in S} \left|\mathbf{Pr}_\rho(\mathcal{A}([\![(t^{\sigma_i}_{\alpha_i}(\psi_i^{\vec{\alpha}}))_{i\leq m}]\!]_{\eta,\rho}) = 1) - \mathbf{Pr}_\rho(\mathcal{A}([\![(t^{\tilde{\sigma}_i}_{\alpha_i}(\tilde{\psi}_i^{\vec{\alpha}}))_{i\leq m}]\!]_{\eta,\rho}) = 1)\right|$$

Since $\mathcal{A}$ distinguishes $[\![\phi_{m+1}]\!]$ and $[\![\tilde{\phi}_{m+1}]\!]$ with non negligible probability, and since S is finite we know that there exists $\vec{\alpha}$ such that:

$$\left|\mathbf{Pr}_\rho(\mathcal{A}([\![(t^{\sigma_i}_{\alpha_i}(\psi_i^{\vec{\alpha}}))_{i\leq m}]\!]_{\eta,\rho}) = 1) - \mathbf{Pr}_\rho(\mathcal{A}([\![(t^{\tilde{\sigma}_i}_{\alpha_i}(\tilde{\psi}_i^{\vec{\alpha}}))_{i\leq m}]\!]_{\eta,\rho}) = 1)\right|$$

is non negligible. Therefore m-Fixed Trace Privacy does not hold, which concludes this proof. ∎

As explained in [4], indistinguishability properties can be expressed in the logic for arbitrary *determinate* protocols. For such protocols, observational equivalence and trace equivalence coincide [14]. The above proposition is a similar result in the computational model. It can be extended to other equivalence properties, as long as m does not depend on the security parameter.

IV. Two RFID protocols

We are now going to describe the LAK and KCL RFID protocols, as well, attacks, patches and formal (computational) security proofs of the fixed versions.

We first consider that names are randomly generated numbers, even though, because of the limited computing capabilities of the tags, they have to be implemented using a Cryptographic Pseudo Random Number Generator (PRNG). This issue will be discussed in the section V: we will show that we can always safely abstract the pseudo random numbers as random numbers, provided that a PRNG is used and the random seed is never used for other purposes.

A. A known attack on KCL

Let us return to the simple example described in Figure 2:

$$\begin{aligned} R &: \mathsf{n}_R \xleftarrow{\$} \\ T_{\mathsf{A}} &: \mathsf{n}_T \xleftarrow{\$} \\ \\ 1 : R \longrightarrow T_{\mathsf{A}} &: \mathsf{n}_R \\ 2 : T_{\mathsf{A}} \longrightarrow R &: \langle \mathsf{A} \oplus \mathsf{n}_T\, ,\, \mathsf{n}_T \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k}_{\mathsf{A}}) \rangle \end{aligned}$$

As reported in [28], there is a simple attack that we depict in Figure 4. In this attack the tag is challenged twice with the same name: observing the exchanges between the tag and the reader, the adversary can replay the name. Finally the adversary checks if he is talking with the same tag by xoring the two components of the message sent by the second tag, and verifies whether the result is the same as what he obtained with the same operation in the first session.

In the left execution, the xor of the two part of the tag answers will be the same:

$$\begin{aligned} \mathsf{T}_{\mathsf{A}} \oplus \mathsf{n}_T \oplus \mathsf{n}_T \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k}_{\mathsf{A}}) &= \mathsf{T}_{\mathsf{A}} \oplus \mathsf{n}'_T \oplus \mathsf{n}'_T \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k}_{\mathsf{A}}) \\ &= \mathsf{T}_{\mathsf{A}} \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k}_{\mathsf{A}}) \end{aligned}$$

$$
\begin{array}{lcl|lcl}
1 : R \longrightarrow T_{\mathsf{A}} & : & \mathsf{n}_R & R \longrightarrow T_{\mathsf{A}} & : & \mathsf{n}_R \\
2 : T_{\mathsf{A}} \longrightarrow R & : & \langle \mathsf{T}_{\mathsf{A}} \oplus \mathsf{n}_T\,,\, \mathsf{n}_T \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k}_{\mathsf{A}}) \rangle & T_{\mathsf{A}} \longrightarrow R & : & \langle \mathsf{T}_{\mathsf{A}} \oplus \mathsf{n}_T\,,\, \mathsf{n}_T \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k}_{\mathsf{A}}) \rangle \\
 & & & & & \\
3 : E \longrightarrow T_{\mathsf{A}} & : & \mathsf{n}_R & E \longrightarrow T_{\mathsf{B}} & : & \mathsf{n}_R \\
4 : T_{\mathsf{A}} \longrightarrow R & : & \langle \mathsf{T}_{\mathsf{A}} \oplus \mathsf{n}'_T\,,\, \mathsf{n}'_T \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k}_{\mathsf{A}}) \rangle & T_{\mathsf{B}} \longrightarrow R & : & \langle \mathsf{T}_{\mathsf{B}} \oplus \mathsf{n}'_T\,,\, \mathsf{n}'_T \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k}_{\mathsf{B}}) \rangle
\end{array}
$$

Fig. 4. Attack against the original KCL protocol

Whereas in the right execution we will obtain two values $\mathsf{T}_{\mathsf{A}} \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k}_{\mathsf{A}})$ and $\mathsf{T}_{\mathsf{B}} \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k}_{\mathsf{B}})$ which will be different with high probability.

B. KCL$^+$, a revised version of KCL

We propose a simple correction to the KCL protocol: we replace the first occurrence of the name n_T with its hash, breaking the algebraic property that was used in the attack. This protocol is depicted in Figure 5, and to our knowledge there exists no formal study of this revised version.

$$
\begin{array}{lcl}
R & : & \mathsf{n}_R \xleftarrow{\$} \\
T & : & \mathsf{n}_T \xleftarrow{\$} \\
 & & \\
1 : R \longrightarrow T_{\mathsf{A}} & : & \mathsf{n}_R \\
2 : T_{\mathsf{A}} \longrightarrow R & : & \langle \mathsf{A} \oplus \mathsf{H}(\mathsf{n}_T, \mathsf{k}_{\mathsf{A}})\,,\, \mathsf{n}_T \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k}_{\mathsf{A}}) \rangle
\end{array}
$$

Fig. 5. The KCL$^+$ protocol

We are now going to illustrate our method by showing that the KCL$^+$ protocol verifies m-Fixed Trace Privacy with two tags A and B. Assuming collision resistance only, there is actually an attack on the protocol KCL$^+$ (exactly the same attack as the one described in Section IV-D). We therefore assume the PRF property.

Theorem 2 (Unlinkability for an arbitrary number of rounds)**.** *Assuming PRF for the keyed hash function, the* KCL$^+$ *protocol verifies m-Fixed Trace Privacy for two agents and all m.*

In the proof below, the primed version of a term t is the term t, in which the names $\mathsf{n}_1, \ldots, \mathsf{n}_l$ appearing in t have been replaced by $\mathsf{n}'_1, \ldots, \mathsf{n}'_l$. We will use t^{Id}_{ϕ} (where $\mathsf{Id} = \mathsf{A}$ or B) to denote the response of the tag T_{Id} to a challenge:

$$t^{\mathsf{Id}}_{\phi} = \langle \mathsf{Id} \oplus \mathsf{H}(\mathsf{n}_T, \mathsf{k}_{\mathsf{Id}})\,,\, \mathsf{n}_T \oplus \mathsf{H}(g(\phi), \mathsf{k}_{\mathsf{Id}}) \rangle$$

Since the axioms are Horn clauses, we can view them as inference rules, in which the premises of the inferences are the negative literals of the clause. This is easier to display and read.

Proof: We prove this by induction on m (induction is outside our logic). Let $\phi, \tilde{\phi}$ be two sequences of terms from the m-Fixed Trace Privacy definition. By induction hypothesis, we assume that we have a derivation of $\phi \sim \tilde{\phi}$ (in the base case, this is the reflexivity of $\sim$). We consider two cases.

If the adversary decides to start a new session with the reader, we need to show that $\phi, \mathsf{n}_R \sim \tilde{\phi}, \mathsf{n}_R$ where n_R is fresh in $\phi, \tilde{\phi}$. This case is easy, we only need to apply the $FreshNonce$ axiom:

$$\frac{\phi \sim \tilde{\phi}}{\phi, \mathsf{n}_R \sim \tilde{\phi}, \mathsf{n}_R}\, FreshNonce$$

Otherwise, the adversary decides to interact with the tags, e.g. A on the left and B on the right (the other cases are identical). We want to show that $\phi, t^{\mathsf{A}}_{\phi} \sim \tilde{\phi}, \tilde{t}^{\mathsf{B}}_{\tilde{\phi}}$. We let n be a fresh name and $\psi, \tilde{\psi}$ be defined by:

$$\psi \equiv \phi, \mathsf{n}_T \oplus \mathsf{H}(g(\phi), \mathsf{k}_{\mathsf{A}})$$
$$\tilde{\psi} \equiv \tilde{\phi}, \mathsf{n}_T \oplus \mathsf{H}(g(\tilde{\phi}), \mathsf{k}_{\mathsf{B}})$$

We start (from the root) our proof by applying the FA axiom (breaking the pair) and then to introduce an intermediate term $\mathsf{A} \oplus \mathsf{n}$ since, intuitively, $\mathsf{H}(\mathsf{n}_T, \mathsf{k}_{\mathsf{A}})$ (resp. $\mathsf{H}(\mathsf{n}_T, \mathsf{k}_{\mathsf{B}})$) should be indistinguishable from a random number.

$$\dfrac{\dfrac{\dfrac{\dfrac{\psi, \mathsf{H}(\mathsf{n}_T, \mathsf{k}_{\mathsf{A}}) \sim \psi, \mathsf{n}}{\psi, \mathsf{A}, \mathsf{H}(\mathsf{n}_T, \mathsf{k}_{\mathsf{A}}) \sim \psi, \mathsf{A}, \mathsf{n}}\, FA}{\psi, \mathsf{A} \oplus \mathsf{H}(\mathsf{n}_T, \mathsf{k}_{\mathsf{A}}) \sim \psi, \mathsf{A} \oplus \mathsf{n}}\, FA \qquad P_1}{\psi, \mathsf{A} \oplus \mathsf{H}(\mathsf{n}_T, \mathsf{k}_{\mathsf{A}}) \sim \tilde{\psi}, \mathsf{B} \oplus \mathsf{H}(\mathsf{n}_T, \mathsf{k}_{\mathsf{B}})}\, Trans}{\phi, t^A_{\phi} \sim \tilde{\phi}, t^B_{\tilde{\phi}}}\, FA$$

where P_1 is a derivation of $\psi, \mathsf{A} \oplus \mathsf{n} \sim \tilde{\psi}, \mathsf{B} \oplus \mathsf{H}(\mathsf{n}_T, \mathsf{k}_{\mathsf{B}})$.

a) Left Derivation: We have to find first a a derivation of $\psi, \mathsf{H}(\mathsf{n}_T, \mathsf{k}_{\mathsf{A}}) \sim \psi, \mathsf{n}$. The ultimate goal is to apply the PRF axioms. For that, we need to introduce, on both sides of the $\sim$ predicate, equality tests between the last message hashed under key k_{A} (i.e. n_T), and all the previous hashed messages under key k_{A}. We let $m_1, \ldots, m_l$ be the set of messages hashed with k_{A} in ϕ. We know that these messages are either names n'_T, or of the form $g(\phi')$ where ϕ' is a strict prefix of ϕ.

Let $\alpha = \mathsf{H}(\mathsf{n}_T, \mathsf{k}_{\mathsf{A}})$, $\beta = \mathsf{n}$. For all $1 \leq i \leq l$ we let $e_i \equiv \mathsf{EQ}(\mathsf{n}_T; m_i)$, and s^x be the term:

$$\begin{array}{c} \mathsf{if}\ e_1\ \mathsf{then}\ x\ \mathsf{else} \\ \mathsf{if}\ e_2\ \mathsf{then}\ x\ \mathsf{else} \\ \vdots \\ \mathsf{if}\ e_l\ \mathsf{then}\ x\ \mathsf{else}\ x \end{array}$$

We observe that, for every term u, $u = s^u$ is derivable from the equality axioms. We are now going to use the CS axiom to split the proof. To do so we introduce for every $1 \leq i \leq l$ the term u^x_i:

$$\begin{array}{c} \mathsf{if}\ e_1\ \mathsf{then}\ 0\ \mathsf{else} \\ \vdots \\ \mathsf{if}\ e_{i-1}\ \mathsf{then}\ 0\ \mathsf{else} \\ \mathsf{if}\ e_i\ \mathsf{then}\ x\ \mathsf{else}\ 0 \end{array}$$

And the term u^x_{l+1}:

$$\mathsf{if}\ e_1\ \mathsf{then}\ 0\ \mathsf{else}\ \ldots \mathsf{if}\ e_l\ \mathsf{then}\ 0\ \mathsf{else}\ x$$

By repeatedly applying the CS axiom we obtain:

$$\dfrac{\dfrac{\forall i \in \{1,\ldots,l+1\},\ \psi, e_1, \ldots, e_l, u_i^\alpha \sim \tilde{\psi}, e_1, \ldots, e_l, u_i^\beta}{\psi, s^{\mathsf{H}(\mathsf{n}_T,\mathsf{k}_\mathsf{A})} \sim \psi, s^{\mathsf{n}}}\, CS}{\psi, \mathsf{H}(\mathsf{n}_T, \mathsf{k}_\mathsf{A}) \sim \psi, \mathsf{n}}\, Congr$$

First note that, using the $EqIndep$ axiom, we derive, for every $1 \leq i \leq l$, $e_i = \mathsf{false}$. This allows us to deal with cases 1 to l, since this implies that $u_i^\alpha = u_i^\alpha = 0$ is derivable. Therefore we have for all $i \in \{1,\ldots,l\}$:

$$\dfrac{\dfrac{}{\psi, \mathsf{false}, \ldots, \mathsf{false}, 0 \sim \psi, \mathsf{false}, \ldots, \mathsf{false}, 0}\, Refl}{\psi, e_1, \ldots, e_l, u_i^\alpha \sim \psi, e_1, \ldots, e_l, u_i^\beta}\, Congr$$

Consider now the case $i = l+1$. The conditions on the occurrences of H and k_A are satisfied, thanks to the choice of $e_1, \ldots, e_l$. We may use the PRF_l axiom:

$$\dfrac{\dfrac{\dfrac{}{\psi, u_1^\alpha \sim \psi, u_1^\beta}\, PRF}{\psi, \mathsf{false}, \ldots, \mathsf{false}, u_{l+1}^\alpha \sim \psi, \mathsf{false}, \ldots, \mathsf{false}, u_{l+1}^\beta}\, FA^{(l)}}{\psi, e_1, \ldots, e_l, u_1^\alpha \sim \psi, e_1, \ldots, e_l, u_1^\beta}\, Congr$$

b) Right Derivation (P_1): Now, we have to derive $\psi, \mathsf{A} \oplus \mathsf{n} \sim \tilde{\psi}, \mathsf{B} \oplus \mathsf{H}(\mathsf{n}_T, \mathsf{k}_\mathsf{B})$. We start by replacing A with B, splitting again the proof in two subgoals:

$$\dfrac{\psi, \mathsf{A} \oplus \mathsf{n} \sim \tilde{\psi}, \mathsf{B} \oplus \mathsf{n} \qquad \tilde{\psi}, \mathsf{B} \oplus \mathsf{n} \sim \tilde{\psi}, \mathsf{B} \oplus \mathsf{H}(\mathsf{n}_T, \mathsf{k}_\mathsf{B})}{\psi, \mathsf{A} \oplus \mathsf{n} \sim \tilde{\psi}, \mathsf{B} \oplus \mathsf{H}(\mathsf{n}_T, \mathsf{k}_\mathsf{B})}\, Trans$$

For the right part, we first decompose the goal:

$$\dfrac{\dfrac{\dfrac{\tilde{\psi}, \mathsf{H}(\mathsf{n}_T, \mathsf{k}_\mathsf{B}) \sim \tilde{\psi}, \mathsf{n}}{\tilde{\psi}, \mathsf{n} \sim \tilde{\psi}, \mathsf{H}(\mathsf{n}_T, \mathsf{k}_\mathsf{B})}\, Sym}{\tilde{\psi}, \mathsf{B}, \mathsf{n} \sim \tilde{\psi}, \mathsf{B}, \mathsf{H}(\mathsf{n}_T, \mathsf{k}_\mathsf{B})}\, FA}{\tilde{\psi}, \mathsf{B} \oplus \mathsf{n} \sim \tilde{\psi}, \mathsf{B} \oplus \mathsf{H}(\mathsf{n}_T, \mathsf{k}_\mathsf{B})}\, FA$$

Then, the derivation of $\tilde{\psi}, \mathsf{H}(\mathsf{n}_T, \mathsf{k}_\mathsf{B}) \sim \tilde{\psi}, \mathsf{n}$ is similar to the derivation of $\psi, \mathsf{H}(\mathsf{n}_T, \mathsf{k}_\mathsf{A}) \sim \psi, \mathsf{n}$.

For the left part, we start by using the axiom on $\oplus$:

$$\dfrac{\psi \sim \tilde{\psi}}{\psi, \mathsf{A} \oplus \mathsf{n} \sim \tilde{\psi}, \mathsf{B} \oplus \mathsf{n}}\, Indep$$

It only remains to show that $\psi \sim \tilde{\psi}$. We do this using the $Trans$ and $Indep$ axioms:

$$\overbrace{\phi, \mathsf{n}_T \oplus \mathsf{H}(g(\phi), \mathsf{k}_\mathsf{A}) \sim \underbrace{\phi, \mathsf{n}_T}_{\mathsf{MSim}}}^{\mathsf{LSim}} \sim \overbrace{\tilde{\phi}, \mathsf{n}_T}^{\mathsf{RSim}} \sim \phi, \mathsf{n}_T \oplus \mathsf{H}(g(\phi), \mathsf{k}_\mathsf{B})$$

$$\dfrac{\dfrac{\dfrac{}{\phi \sim \phi}\, Refl}{\mathsf{LSim}}\, Indep \quad \dfrac{\phi \sim \tilde{\phi}}{\mathsf{MSim}}\, Indep \quad \dfrac{\dfrac{}{\phi \sim \phi}\, Refl}{\mathsf{RSim}}\, Indep}{\phi, \mathsf{n}_T \oplus \mathsf{H}(g(\phi), \mathsf{k}_\mathsf{A}) \sim \phi, \mathsf{n}_T \oplus \mathsf{H}(g(\phi), \mathsf{k}_\mathsf{B})}\, Trans$$

Which concludes the inductive step proof. ■

In this result we consider only two tags, for simplicity reasons. In particular they cannot be corrupted tags. Our framework is expressive enough for multiple tags, including corrupted one, though we did not complete the proof in that case.

$$\begin{array}{lcl} R & : & \mathsf{n}_R \xleftarrow{\$} \\ T_\mathsf{A} & : & \mathsf{n}_T \xleftarrow{\$} \end{array}$$

$$\begin{array}{lcl} 1 : R \longrightarrow T_\mathsf{A} & : & \mathsf{n}_R \\ 2 : T_\mathsf{A} \longrightarrow R & : & \langle \mathsf{n}_T\, ,\, \mathsf{h}(\mathsf{n}_R \oplus \mathsf{n}_T \oplus \mathsf{k}_\mathsf{A}) \rangle \\ 3 : R \longrightarrow T_\mathsf{A} & : & \mathsf{h}\big(\big(\mathsf{h}(\mathsf{n}_R \oplus \mathsf{n}_T \oplus \mathsf{k}_\mathsf{A}) \oplus \mathsf{n}_R \oplus \mathsf{k}_\mathsf{A}\big) \end{array}$$

$$\begin{array}{lcl} R & : & \mathsf{k}_\mathsf{A} = \mathsf{h}(\mathsf{k}_\mathsf{A})\ ,\ \mathsf{k}^0_\mathsf{A} = \mathsf{k}_\mathsf{A} \\ T_\mathsf{A} & : & \mathsf{k}_\mathsf{A} = \mathsf{h}(\mathsf{k}_\mathsf{A}) \end{array}$$

Fig. 6. The LAK protocol

C. The LAK protocol

a) Description: We describe in Figure 6 the original protocol from [22]. As we mentioned before, the protocol we consider is a simplified version of the LAK protocol, without the key server. In the LAK protocol, the reader shares a private key k_A with each of its tags T_A. h is an hash function. This is a stateful protocol: the key is updated after each successful completion of the protocol, and the reader keeps in k^0_A the previous value of the key. This value is used as a backup in case T_A has not completed the protocol (for example because the last message was lost) and therefore not updated its version of the key. The protocol allows to recover from such a desynchronization: the reader R can use the previous version of k_A at the next round (which is the version used by T_A) and finish the protocol.

The protocol is supposed to achieve mutual authentication and unlinkability. Even though such properties can be defined in various ways, we recall below a known attack against the LAK protocol, which will force us to modify it.

b) An attack on LAK: An attack on authentication is described in [28] and sketched below:

$$\begin{array}{ll} 1: & R \longrightarrow T_\mathsf{A} : \mathsf{n}_R \\ 2: & T_\mathsf{A} \longrightarrow E : \langle \mathsf{n}_T\, ,\, \mathsf{h}(\mathsf{n}_R \oplus \mathsf{n}_T \oplus \mathsf{k}_\mathsf{A}) \rangle \\ & \\ 3: & R \longrightarrow E : \mathsf{n}'_R \\ 4: & E \longrightarrow R : \langle \mathsf{n}'_R \oplus \mathsf{n}_R \oplus \mathsf{n}_T\, ,\, \mathsf{h}(\mathsf{n}_R \oplus \mathsf{n}_T \oplus \mathsf{k}_\mathsf{A}) \rangle \\ 5: & R \longrightarrow E : \mathsf{h}\big(\big(\mathsf{h}(\mathsf{n}_R \oplus \mathsf{n}_T \oplus \mathsf{k}_\mathsf{A}) \oplus \mathsf{n}'_R \oplus \mathsf{k}_\mathsf{A}\big) \end{array}$$

In this attack, the adversary simply observes the beginning of an honest execution of the protocol (without completing the protocol, so that the reader and the tag do not update the key) between a tag A and the reader. The adversary obtains $\mathsf{h}(\mathsf{n}_R \oplus \mathsf{n}_T \oplus \mathsf{k}_\mathsf{A})$ and the names $\mathsf{n}_R, \mathsf{n}_T$. He then interacts with the reader to get a new name n'_R and impersonates the tag A by choosing the returned tag n'_T such that $\mathsf{n}'_R \oplus \mathsf{n}'_T = \mathsf{n}_R \oplus \mathsf{n}_T$.

D. A stateless revised version of LAK

In [18], the authors consider a corrected (and stateless) version of the protocol, which they proved secure. This version

of the protocol is described below:

$$\begin{array}{lcl} R & : & \mathsf{n}_R \xleftarrow{\$} \\ T_\mathsf{A} & : & \mathsf{n}_T \xleftarrow{\$} \end{array}$$

$$\begin{array}{lcl} 1 : R \longrightarrow T_\mathsf{A} & : & \mathsf{n}_R \\ 2 : T_\mathsf{A} \longrightarrow R & : & \langle \mathsf{n}_T \, , \, \mathsf{h}(\langle \mathsf{n}_R, \mathsf{n}_T, \mathsf{k}_\mathsf{A} \rangle) \rangle \\ 3 : R \longrightarrow T_\mathsf{A} & : & \mathsf{h}\big(\langle (\mathsf{h}(\langle \mathsf{n}_R, \mathsf{n}_T, \mathsf{k}_\mathsf{A} \rangle), \mathsf{n}_R, \mathsf{k}_\mathsf{A} \rangle\big) \end{array}$$

This new version avoids the previous attack, which relied on the algebraic properties of exclusive-or. Formally, the protocol is described in the applied pi-calculus in [18], in which they prove the strong unlinkability property of [1] in the Dolev-Yao model for an unbounded number of session.

a) Attack against stateless LAK*:* Since the stateless version of LAK was proved in the symbolic model, no computational security assumptions were made on h. We show in Figure 7 that choosing h to be a one-way cryptographic hash function (OW-CPA and Strongly Collision Resistant for example) is not enough to guarantee unlinkability.

The attack is quite simple: it suffices that the hash function h leaks a few bits of the hashed message (which is possible for an one-way hash function). This means that, when hashing a message of the form $\langle \mathsf{n}_R, \mathsf{n}_T, \mathsf{k} \rangle$, the hash function h will leak some bits of the agent key k. Since the keys are drawn uniformly at random, there is a non negligible probability for the leaked bits to be different when hashing messages with different keys. In particular an adversary will be be able to distinguish $\mathsf{h}(\langle \mathsf{n}_R, \mathsf{n}_T, \mathsf{k}_\mathsf{A} \rangle), \mathsf{h}(\langle \mathsf{n}'_R, \mathsf{n}'_T, \mathsf{k}_\mathsf{A} \rangle)$ from $\mathsf{h}(\langle \mathsf{n}_R, \mathsf{n}_T, \mathsf{k}_\mathsf{A} \rangle), \mathsf{h}(\langle \mathsf{n}'_R, \mathsf{n}'_T, \mathsf{k}_\mathsf{B} \rangle)$ with high probability.

Observe that this attack would still work if we modified the protocol to update the keys after a successful execution of the protocol (in other word, if we consider the original LAK protocol with concatenation instead of xor), because the attacker could start executions of the protocol without finishing them, preventing the keys from being updated.

Remark 3. In the original paper introducing LAK [22], the hash function is described as a one-way cryptographic hash function, which a priori does not prevent the attack described above. However, in the security analysis section, the authors assume the function to be indistinguishable from a random oracle, which prevents the attack. It is actually sufficient to assume PRF, for which there are effective constructions (subject to hardness assumptions).

E. The LAK$^+$ *protocol*

We describe here a stateless version of the LAK protocol, that we call LAK$^+$. As in the LAK protocol, the reader shares with each tag a secret key k. We use a keyed-hash function that is assumed to be PRF to prevent the attack depicted in Section IV-D. This protocol uses a function c that combines the names. It could be a priori a xor, as in the original protocol, or a pairing, as in the revised version of [18] or something else. We will look for sufficient conditions on this function c, such that the protocol is secure.

$$\begin{array}{lcl} R & : & \mathsf{n}_R \xleftarrow{\$} \\ T & : & \mathsf{n}_T \xleftarrow{\$} \end{array}$$

$$\begin{array}{lcl} 1 : R \longrightarrow T & : & \mathsf{n}_R \\ 2 : T \longrightarrow R & : & \langle \mathsf{n}_T \, , \, \mathsf{H}(\mathsf{c}(\mathsf{n}_R, \mathsf{n}_T), \mathsf{k}_\mathsf{A}) \rangle \\ 3 : R \longrightarrow T & : & \mathsf{H}\big(\mathsf{c}(\mathsf{H}(\mathsf{c}(\mathsf{n}_R, \mathsf{n}_T), \mathsf{k}_\mathsf{A}), \mathsf{n}_R), \mathsf{k}_\mathsf{A}\big) \end{array}$$

We start by describing two different attacks that rely on some properties of the function c. In each case, we give a sufficient condition on c that prevents the attack. Next, we show that these two conditions are sufficient to prove that the LAK$^+$ protocol verifies the Bounded Session Privacy property.

a) First Attack:: The attack depicted below is a generalization of the attack from [28]. It works when there exists a function s (computable in probabilistic polynomial time) such that the quantity below is not negligible:

$$\mathbf{Pr}\big(\mathsf{n}_R, \mathsf{n}_T, \mathsf{n}'_R : \mathsf{EQ}(\mathsf{c}(\mathsf{n}_R, \mathsf{n}_T); \mathsf{c}(\mathsf{n}'_R, s(\mathsf{n}_R, \mathsf{n}_T, \mathsf{n}'_R)))\big) \quad (1)$$

This condition is satisfied if c is the xor operation (choose $s(\mathsf{n}_R, \mathsf{n}_T, \mathsf{n}'_R) = \mathsf{n}'_R \oplus \mathsf{n}_T \oplus \mathsf{n}'_R$).

$$\begin{array}{lcl} 1 : & E \longrightarrow T_A : \mathsf{n}_R \\ 2 : & T_A \longrightarrow E : \langle \mathsf{n}_T \, , \, \mathsf{H}(\mathsf{c}(\mathsf{n}_R, \mathsf{n}_T), \mathsf{k}_\mathsf{A}) \rangle \\ \\ 3 : & R \longrightarrow E : \mathsf{n}'_R \\ 4 : & E \longrightarrow R : \langle s(\mathsf{n}_R, \mathsf{n}_T, \mathsf{n}'_R) \, , \, \mathsf{H}(\mathsf{c}(\mathsf{n}_R, \mathsf{n}_T), \mathsf{k}_\mathsf{A}) \rangle \\ 5 : & R \longrightarrow E : \mathsf{H}\big(\mathsf{c}(\mathsf{H}(\mathsf{c}(\mathsf{n}_R, \mathsf{n}_T), \mathsf{k}_\mathsf{A}), \mathsf{n}_R), \mathsf{k}_\mathsf{A}\big) \end{array}$$

The attacker starts by sending a name n_R to the tag, and gets the name n_T chosen by the tag as well as the hash $\mathsf{H}(\mathsf{c}(\mathsf{n}_R, \mathsf{n}_T), \mathsf{k}_\mathsf{A})$. Then the attacker initiates a second round of the protocol with the reader. The reader sends first a name n'_R. The attacker is then able to answer, re-using the hash $\mathsf{H}(\mathsf{c}(\mathsf{n}_R, \mathsf{n}_T), \mathsf{k}_\mathsf{A})$ sent by the tag in the first round, choosing $s(\mathsf{n}_R, \mathsf{n}_T, \mathsf{n}'_R)$ as a replacement of the name n'_T. Using Equation (1), there is a non negligible probability for the reader to accept the forged message as genuine.

This attack can be prevented by requiring c to be injective on its first argument:

$$\forall a, b, x, y.\ \mathsf{EQ}(\mathsf{c}(a, b); \mathsf{c}(x, y)) \Rightarrow \mathsf{EQ}(a; x)$$

b) Second Attack:: We have an unlinkability attack if we can distinguish between the answers of the tags, even though the hash function is assumed to be a PRF. This is possible if there exists a constant g_1 and a function s such that:

$$\mathbf{Pr}\big(x, y : \mathsf{c}(g_1, x) = \mathsf{c}(s(x), y)\big) \text{ is not negligible} \quad (2)$$

If this is the case, then the unlinkability attack described below has a non negligible probability of success in distinguishing two consecutive rounds with the same tag A from one round with the tag A and one round with the tag B.

The attack works as follows: it starts by impersonating the reader, sends g_1 to the tag and gets the response $\langle \mathsf{n}_T \, , \, \mathsf{H}(\mathsf{c}(g_1, \mathsf{n}_T), \mathsf{k}_\mathsf{A}) \rangle$. Then the attacker initiates a new round of the protocol by sending $s(\mathsf{n}_T)$ to the second tag. Using Equation 2, there is a non negligible probability that

$$\begin{array}{lcl|lcl} 1 : E \longrightarrow T_{\mathsf{A}} & : & \mathsf{n}_R & E \longrightarrow T_{\mathsf{A}} & : & \mathsf{n}_R \\ 2 : T_{\mathsf{A}} \longrightarrow E & : & \langle \mathsf{n}_T\,,\, \mathsf{h}(\langle \mathsf{n}_R, \mathsf{n}_T, \mathsf{k}_{\mathsf{A}}\rangle)\rangle & T_{\mathsf{A}} \longrightarrow E & : & \langle \mathsf{n}_T\,,\, \mathsf{h}(\langle \mathsf{n}_R, \mathsf{n}_T, \mathsf{k}_{\mathsf{A}}\rangle)\rangle \\ & & & & & \\ 3 : E \longrightarrow T_{\mathsf{A}} & : & \mathsf{n}'_R & E \longrightarrow T_{\mathsf{B}} & : & \mathsf{n}'_R \\ 4 : T_{\mathsf{A}} \longrightarrow E & : & \langle \mathsf{n}'_T\,,\, \mathsf{h}(\langle \mathsf{n}'_R, \mathsf{n}'_T, \mathsf{k}_{\mathsf{A}}\rangle)\rangle & T_{\mathsf{B}} \longrightarrow E & : & \langle \mathsf{n}'_T\,,\, \mathsf{h}(\langle \mathsf{n}'_R, \mathsf{n}'_T, \mathsf{k}_{\mathsf{B}}\rangle)\rangle \end{array}$$

Fig. 7. Unlinkability attack in two rounds against the stateless LAK protocol

the hash in the response from the tag A in the second round of the protocol is the same as in the first round, whereas this will not be the case if the second round is initiated with B.

This attack can be prevented by asking c to be right injective on its second argument:

$$\forall a, b, x, y.\ \mathsf{EQ}(\mathsf{c}(a, b); \mathsf{c}(x, y)) \Rightarrow \mathsf{EQ}(b; y)$$

c) Unlinkability of the LAK$^+$ *protocol:* To prevent all the attacks against LAK$^+$ described above, we are going to require c to be right and left injective. This can easily be expressed in the logic using the two axioms in Figure 9, which are satisfied, for instance, when c is a pairing.

$$\begin{array}{l} \mathsf{if}\ \mathsf{EQ}(u; u')\ \mathsf{then}\ \mathsf{false}\ \mathsf{else}\ \mathsf{EQ}(\mathsf{c}(u, v); \mathsf{c}(u', v')) = \mathsf{false} \\ \mathsf{if}\ \mathsf{EQ}(v; v')\ \mathsf{then}\ \mathsf{false}\ \mathsf{else}\ \mathsf{EQ}(\mathsf{c}(u, v); \mathsf{c}(u', v')) = \mathsf{false} \end{array}$$

Fig. 9. Injectivity axioms on the combination function c

Three messages are sent in a complete session of the LAK$^+$ protocol: two by the reader and one by the tag. Therefore, if we want to show interesting properties of the LAK$^+$ protocol, we need to consider at least 6 terms in the trace (two full sessions, e.g. twice with the same tag T_{A} or with the tag T_{A} and the tag T_{B}). This leads us to consider the 6-Fixed Trace Privacy of the LAK$^+$ protocol.

Theorem 3. *The* LAK$^+$ *protocol verifies* 6*-Fixed Trace Privacy. In particular, the following formula is derivable using the axioms from Section II and the axioms in Figure 9:*

$$\mathsf{n}_R, s^{\mathsf{A}}_{\phi_0}, t^{\mathsf{A}}_{\phi_1}, \mathsf{n}'_R, s'^{\mathsf{A}}_{\phi_2}, t'^{\mathsf{A}}_{\phi_3} \sim \mathsf{n}_R, s^{\mathsf{A}}_{\phi_0}, t^{\mathsf{A}}_{\phi_1}, \mathsf{n}'_R, s'^{\mathsf{B}}_{\phi_2}, t'^{\mathsf{B}}_{\tilde{\phi}_3}$$

Where:

$$\begin{array}{l} s^{\mathsf{Id}}_{\phi} = \langle \mathsf{n}_T\,,\, \mathsf{H}(\mathsf{c}(g(\phi), \mathsf{n}_T), \mathsf{k}_{\mathsf{Id}})\rangle \\ t^{\mathsf{Id}}_{\phi} = \ \mathsf{if}\ \mathsf{EQ}\big(\mathsf{H}\big(\mathsf{c}(\mathsf{n}_R, \pi_1(g(\phi))), \mathsf{k}_{\mathsf{Id}}\big); \pi_2(g(\phi))\big) \\ \qquad\quad \mathsf{then}\ \mathsf{H}\big(\mathsf{c}(\pi_2(g(\phi)), \mathsf{n}_R), \mathsf{k}_{\mathsf{Id}}\big) \\ \phi_0 = \mathsf{n}_R \qquad \phi_1 = \mathsf{n}_R, s^{\mathsf{A}}_{\phi_0} \qquad \phi_2 = \mathsf{n}_R, s^{\mathsf{A}}_{\phi_0}, t^{\mathsf{A}}_{\phi_1} \\ \phi_3 = \mathsf{n}_R, s^{\mathsf{A}}_{\phi_0}, t^{\mathsf{A}}_{\phi_1}, \mathsf{n}'_R, s'^{\mathsf{A}}_{\phi_2} \qquad \tilde{\phi}_3 = \mathsf{n}_R, s^{\mathsf{A}}_{\phi_0}, t^{\mathsf{A}}_{\phi_1}, \mathsf{n}'_R, s'^{\mathsf{B}}_{\phi_2} \end{array}$$

The proof of this theorem is given in Appendix A. As for the KCL$^+$ protocol, by induction on m, it should be possible to generalize the result to an arbitrary m-Fixed Trace Privacy (if m is independent of the security parameter), although we did not complete this generalization formally.

V. Pseudo Random Number Generator

A PRNG uses an internal state, which is updated at each call, and outputs a pseudo random number. This can be modeled by a function G taking the internal state as input, and outputing a pair with the new internal state and the generated pseudo random number (retrieved using the projections π_S and π_o). Besides, a function init_S is used to initialized the internal state with a random seed (which can be hard-coded in the tag).

Definition 5. A PRNG is a tuple of polynomial functions (G, init_S) such that for every PPT adversary $\mathcal{A}$ and for every n, the following quantity is negligible in η:

$$\big|\mathbf{Pr}\left(r \in \{0,1\}^\eta :\ \mathcal{A}(\pi_o(s_0), \ldots, \pi_o(s_n)) = 1\right) - \mathbf{Pr}\left(r_0, \ldots, r_n \in \{0,1\}^\eta :\ \mathcal{A}(r_0, \ldots, r_n) = 1\right)\big|$$

where $s_0 = G(\mathsf{init}_S(r, 1^\eta))$ and for all $0 \leq i < n,\ s_{i+1} = G(\pi_S(s_i))$.

This can be translated in the logic by the $PRNG_n$ axioms:

$$\pi_o(s_0), \ldots, \pi_o(s_n) \sim \mathsf{n}_0, \ldots, \mathsf{n}_n$$
where $s_0 \equiv G(\mathsf{init}_S(\mathsf{n}))$ and $\forall 0 \leq i < n,\ s_{i+1} \equiv G(\pi_S(s_i))$.

The soundness of these axioms is an immediate consequence of Definition 5:

Proposition 5. *The* $(PRNG_n)_n$ *axioms are sound in any computational model* $\mathcal{M}_c$ *where* (G, init_S) *is interpreted as a* PRNG*.*

For each protocol where a strict separation exists between the cryptographic material used for random number generation and the other primitives (e.g. encryption keys), pseudo random numbers generated using a PRNG can be abstracted as random numbers using the following proposition (the proof can be found in [13]):

Proposition 6. *For every names* $\mathsf{n}, (\mathsf{n}_i)_{i \leq n}$ *and contexts* $u_0, \ldots, u_n$ *that do not contain these names, the following formula is derivable using the axioms in Figure 1 and* $PRNG_n$*:*

$$u_0[\pi_o(s_0)], \ldots, u_n[\pi_o(s_n)] \sim u_0[\mathsf{n}_0], \ldots, u_n[\mathsf{n}_n]$$

where $s_0 \equiv G(\mathsf{init}_S(\mathsf{n}))$ *and* $\forall 0 \leq i < n,\ s_{i+1} \equiv G(\pi_S(s_i))$.

Remark 4 (Forward Secrecy). We did not study forward secrecy of RFID protocols, but this could easily be done. The standard forward secrecy assumption on a PRNG states that leaking the internal state $\pi_S(s_n)$ of the PRNG (e.g. with a physical attack on the RFID chip) does not allow the adversary to gain any information about the previously generated names $(\pi_o(s_n))_{i \leq n}$. This could be expressed in the logic using, for example, the following formula:

$$\pi_o(s_0), \ldots, \pi_o(s_n), \pi_S(s_n) \sim \mathsf{n}_0, \ldots, \mathsf{n}_n, \pi_S(s_n)$$
where $s_0 \equiv G(\mathsf{init}_S(\mathsf{n}))$ and $\forall 0 \leq i < n,\ s_{i+1} \equiv G(\pi_S(s_i))$.

$1 : E \longrightarrow T_{\mathsf{A}} \;:\; g_1$	$E \longrightarrow T_{\mathsf{A}} \;:\; g_1$
$2 : T_{\mathsf{A}} \longrightarrow E \;:\; \langle \mathsf{n}_T\,,\, \mathsf{H}(\mathsf{c}(g_1, \mathsf{n}_T), \mathsf{k}_{\mathsf{A}})\rangle$	$T_{\mathsf{B}} \longrightarrow E \;:\; \langle \mathsf{n}_T\,,\, \mathsf{H}(\mathsf{c}(g_1, \mathsf{n}_T), \mathsf{k}_{\mathsf{A}})\rangle$
$3 : E \longrightarrow T_{\mathsf{A}} \;:\; s(\mathsf{n}_T)$	$E \longrightarrow T_{\mathsf{B}} \;:\; s(\mathsf{n}_T)$
$4 : T_{\mathsf{A}} \longrightarrow E \;:\; \langle \mathsf{n}'_T\,,\, \mathsf{H}(\mathsf{c}(s(\mathsf{n}_T), \mathsf{n}'_T), \mathsf{k}_{\mathsf{A}})\rangle$	$T_{\mathsf{B}} \longrightarrow E \;:\; \langle \mathsf{n}'_T\,,\, \mathsf{H}(\mathsf{c}(s(\mathsf{n}_T), \mathsf{n}'_T), \mathsf{k}_{\mathsf{B}})\rangle$

Fig. 8. Unlinkability attack against LAK$^+$

VI. Conclusion

We gave a framework for formally proving the security of RFID protocols in the computational model: we expressed cryptographic assumptions on hash functions and an unlinkability property as formulas of the Complete Symbolic Attacker logic. We then illustrated this method on two examples, providing formal security proofs. We also showed that the security assumptions used in the proofs of these two protocols cannot be weakened (at least not in an obvious way).

What our framework is missing is an automatic tool for the logic, since the formal proofs are already heavy for simple protocols. Building such a tool would help streamline the design of formally verified protocols, and is the goal of our future research.

Compiling the process equivalence in our logic has already been explained in [4]. In principle, we could use any automatic first order theorem prover to complete the proofs. However, the search space may be too large on our examples. This is why the focus of our current research is on the design of appropriate strategies.

References

[1] M. Arapinis, T. Chothia, E. Ritter, and M. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proceedings of IEEE Conf. Computer Security Foundations*, 2010.

[2] Gergei Bana and Rohit Chadha. Verification methods for the computationally complete symbolic attacker based on indistinguishability. *IACR Cryptology ePrint Archive*, 2016:69, 2016.

[3] Gergei Bana and Hubert Comon-Lundh. Towards unconditional soundness: Computationally complete symbolic attacker. In Pierpaolo Degano and Joshua D. Guttman, editors, *Proceedings of the 1st International Conference on Principles of Security and Trust (POST'12)*, volume 7215 of *Lecture Notes in Computer Science*, pages 189–208. Springer, March 2012.

[4] Gergei Bana and Hubert Comon-Lundh. A computationally complete symbolic attacker for equivalence properties. In *Proc. ACM Conference on Computers and Communications Security*, 2014.

[5] Gilles Barthe, Cédric Fournet, Benjamin Grégoire, Pierre-Yves Strub, Nikhil Swamy, and Santiago Zanella-Béguelin. Probabilistic relational verification for cryptographic implementations. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '14, pages 193–205, New York, NY, USA, 2014. ACM.

[6] Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin. *Computer-Aided Security Proofs for the Working Cryptographer*, pages 71–90. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[7] K. Bhargavan, C. Fournet, M. Kohlweiss, A. Pironti, and P. Y. Strub. Implementing tls with verified cryptographic security. In *2013 IEEE Symposium on Security and Privacy*, pages 445–459, May 2013.

[8] Bruno Blanchet. ProVerif*: Cryptographic protocols verifier in the formal model*. available at http://proseccco.gforge..inria.fr/personal/bblanchet/proverif/.

[9] Bruno Blanchet. A computationally sound mechanized prover for security protocols. In *2006 IEEE Symposium on Security and Privacy (S&P 2006), 21-24 May 2006, Berkeley, California, USA*, pages 140–154. IEEE Computer Society, 2006.

[10] Bruno Blanchet, Aaron D. Jaggard, Andre Scedrov, and Joe-Kai Tsay. Computationally sound mechanized proofs for basic and public-key Kerberos. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS'08)*, pages 87–99, Tokyo, Japan, March 2008. ACM.

[11] Vincent Cheval. *APTE: An Algorithm for Proving Trace Equivalence*, pages 587–592. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[12] Hung-Yu Chien. Sasi: A new ultralightweight rfid authentication protocol providing strong authentication and strong integrity. *IEEE Trans. Dependable Secur. Comput.*, 4(4):337–340, October 2007.

[13] Hubert Comon and Adrien Koutsos. Formal computational unlinkability proofs of rfid protocols, 2017. arXiv:1705.02296.

[14] Véronique Cortier and Stéphanie Delaune. A method for proving observational equivalence. In *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF'09)*, pages 266–276, Port Jefferson, NY, USA, July 2009. IEEE Computer Society Press.

[15] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[16] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[17] Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography, 2001.

[18] Lucca Hirschi, David Baelde, and Stéphanie Delaune. A method for verifying privacy-type properties: the unbounded case. In Michael Locasto, Vitaly Shmatikov, and Úlfar Erlingsson, editors, *Proceedings of the 37th IEEE Symposium on Security and Privacy (S&P'16)*, pages 564–581, San Jose, California, USA, May 2016. IEEECSP.

[19] Ari Juels and Stephen A. Weis. Defining strong privacy for rfid. *ACM Trans. Inf. Syst. Secur.*, 13(1):7:1–7:23, November 2009.

[20] I. J. Kim, E. Y. Choi, and D. H. Lee. Secure mobile rfid system against privacy and security problems. In *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2007. SECPerU 2007. Third International Workshop on*, pages 67–72, July 2007.

[21] Tri Van Le, Mike Burmester, and Breno de Medeiros. Universally composable and forward-secure RFID authentication and authenticated key exchange. In Feng Bao and Steven Miller, editors, *Proceedings of the 2007 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2007, Singapore, March 20-22, 2007*, pages 242–252. ACM, 2007.

[22] Sangshin Lee, Tomoyuki Asano, and Kwangjo Kim. Rfid mutual authentication scheme based on synchronized secret information. In *Symposium on cryptography and information security*, 2006.

[23] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The tamarin prover for the symbolic analysis of security protocols. In *Proceedings of the 25th International Conference on Computer Aided Verification*, CAV'13, pages 696–701, Berlin, Heidelberg, 2013. Springer-Verlag.

[24] Khaled Ouafi and Raphael C.-W. Phan. Privacy of recent RFID authentication protocols. In Liqun Chen, Yi Mu, and Willy Susilo, editors, *Information Security Practice and Experience, 4th International Conference, ISPEC 2008, Sydney, Australia, April 21-23, 2008, Proceedings*, volume 4991 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2008.

[25] Pedro Peris-Lopez, Julio César Hernández Castro, Juan M. Estévez-Tapiador, and Arturo Ribagorda. Advances in ultralightweight cryptography for low-cost RFID tags: Gossamer protocol. In Kyo-Il Chung, Kiwook Sohn, and Moti Yung, editors, *Information Security Applications, 9th International Workshop, WISA 2008, Jeju Island, Korea, September 23-25, 2008, Revised Selected Papers*, volume 5379 of *Lecture Notes in Computer Science*, pages 56–68. Springer, 2008.

[26] Guillaume Scerri. *Proofs of security protocols revisited*. PhD thesis, École Normale Supérieure de Cachan, 2015.

[27] Guillaume Scerri and Ryan Stanley-Oakes. Analysis of key wrapping apis: Generic policies, computational security. In *IEEE 29th Computer*

Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016, pages 281–295. IEEE Computer Society, 2016.
[28] Ton Van Deursen and Sasa Radomirovic. Attacks on rfid protocols. *IACR Cryptology ePrint Archive*, 2008:310, 2008.
[29] Serge Vaudenay. *On Privacy Models for RFID*, pages 68–87. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

APPENDIX A
PROOF OF THEOREM 3

Unsurprisingly, it turns out that left and right injectivity of c implies the injectivity of c:

Proposition 7. *The following formula is derivable using the axioms from Section II and the axioms in Figure 9:*

$$\mathsf{EQ}\big(\mathsf{c}(u,v);\mathsf{c}(u',v')\big) = \textit{if } \mathsf{EQ}(u;u') \textit{ then } \big(\textit{if } \mathsf{EQ}(v;v') \textit{ then true else false}\big) \textit{ else false}$$

The proof is straightforward rewriting using left and right injectivity and the if then else axioms.

We are now ready to give the proof of Theorem 3. Most of the formulas are easy to prove, so we are going to focus on the formula explicitly given in the theorem statement, which is in our opinion the hardest case.

Before starting, we define several new terms in Figure 10.

We have similar definition for the tilded versions $\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}, \ldots$. We start by applying the FA axiom several times:

$$\frac{\phi_2, \alpha, \beta, \gamma \sim \phi_2, \tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}}{\phi_3, t'^{A}_{\phi_3} \sim \tilde{\phi}_3, t'^{B}_{\tilde{\phi}_3}}\, FA^*$$

We are now going to use the CS axiom on the conditional e_4, e_5 to split the proof. To do so we introduce the term:

$$u^x \equiv \mathsf{if}\ e_4\ \mathsf{then}\ \big(\mathsf{if}\ e_5\ \mathsf{then}\ x\ \mathsf{else}\ x\big)\ \mathsf{else}\ \big(\mathsf{if}\ e_5\ \mathsf{then}\ x\ \mathsf{else}\ x\big)$$

and the terms:

$$u^x_1 \equiv \mathsf{if}\ e_4\ \mathsf{then}\ 0\ \mathsf{else}\ \big(\mathsf{if}\ e_5\ \mathsf{then}\ 0\ \mathsf{else}\ x\big)$$
$$u^x_2 \equiv \mathsf{if}\ e_4\ \mathsf{then}\ 0\ \mathsf{else}\ \big(\mathsf{if}\ e_5\ \mathsf{then}\ x\ \mathsf{else}\ 0\big)$$
$$u^x_3 \equiv \mathsf{if}\ e_4\ \mathsf{then}\ \big(\mathsf{if}\ e_5\ \mathsf{then}\ 0\ \mathsf{else}\ x\big)\ \mathsf{else}\ 0$$
$$u^x_4 \equiv \mathsf{if}\ e_4\ \mathsf{then}\ \big(\mathsf{if}\ e_5\ \mathsf{then}\ x\ \mathsf{else}\ 0\big)\ \mathsf{else}\ 0$$

Similarly we introduced the tilded versions of these terms. We observe that for all term s we have $s = u^s$ and $s = \tilde{u}^s$. Therefore we can apply the CS axiom, which gives us:

$$\frac{\forall i \in \{1,\ldots,4\},\quad \phi_2, e_4, e_5, u^\alpha_i, u^\beta_i, u^\gamma_i \sim \phi_2, \tilde{e}_4, \tilde{e}_5, \tilde{u}^{\tilde{\alpha}}_i, \tilde{u}^{\tilde{\beta}}_i, \tilde{u}^{\tilde{\gamma}}_i}{\phi_2, \alpha, \beta, \gamma \sim \phi_2, \tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}}\, CS^*$$

We let $\phi = \phi_2, e_4, e_5$ and $\tilde{\phi} = \phi_2, \tilde{e}_4, \tilde{e}_5$.

- **Case** $i = 1$**:** Let n be a fresh name, we start by the derivation P_1 displayed in Figure 11. Using $EqIndep$ we know that $\epsilon_1 = \epsilon_2 = \epsilon_3 = \mathsf{false}$, and using the left injectivity of c this shows that $e_1 = e_2 = e_3 = \mathsf{false}$. Therefore we know that:

$$u^\beta_1 = v^\beta \equiv \mathsf{if}\ e_1\ \mathsf{then}\ 0\ \mathsf{else}\ \mathsf{if}\ e_2\ \mathsf{then}\ 0\ \mathsf{else}\ \big(\mathsf{if}\ e_3\ \mathsf{then}\ 0\ \mathsf{else}\ \big(u^\beta_1\big)\big)$$

$$u^{\mathsf{n}}_1 = v^{\mathsf{n}} \equiv \mathsf{if}\ e_1\ \mathsf{then}\ 0\ \mathsf{else}\ \mathsf{if}\ e_2\ \mathsf{then}\ 0\ \mathsf{else}\ \big(\mathsf{if}\ e_3\ \mathsf{then}\ 0\ \mathsf{else}\ \big(u^{\mathsf{n}}_1\big)\big)$$

Hence we can apply the PRF axiom, which shows that:

$$\frac{\dfrac{}{\phi, u^\alpha_1, v^\beta, u^\gamma_1 \sim \phi, u^\alpha_1, v^{\mathsf{n}}, u^\gamma_1}\, PRF}{\phi, u^\alpha_1, u^\beta_1, u^\gamma_1 \sim \phi, u^\alpha_1, u^{\mathsf{n}}_1, u^\gamma_1}\, Congr$$

Similarly we show that:

$$\frac{\dfrac{}{\tilde{\phi}, \tilde{u}^{\tilde{\alpha}}_1, \tilde{v}^{\mathsf{n}}, \tilde{u}^{\tilde{\gamma}}_1 \sim \tilde{\phi}, \tilde{u}^{\tilde{\alpha}}_1, \tilde{v}^{\tilde{\beta}}, \tilde{u}^{\tilde{\gamma}}_1}\, PRF}{\tilde{\phi}, \tilde{u}^{\tilde{\alpha}}_1, \tilde{u}^{\mathsf{n}}_1, \tilde{u}^{\tilde{\gamma}}_1 \sim \tilde{\phi}, \tilde{u}^{\tilde{\alpha}}_1, \tilde{u}^{\tilde{\beta}}_1, \tilde{u}^{\tilde{\gamma}}_1}\, Congr$$

It remains to show that $\phi, u^\alpha_1, u^\gamma_1 \sim \tilde{\phi}, \tilde{u}^{\tilde{\alpha}}_1, \tilde{u}^{\tilde{\gamma}}_1$. We do this exactly like we did to get rid of the u^β_1 and $\tilde{u}^{\tilde{\beta}}_1$. First we use FA, $Trans$ and $FreshNonce$ to get the derivation P_2 displayed in Figure 11.

The formulas $\phi, u^\alpha_1, u^\gamma_1 \sim \phi, u^\alpha_1, u^{\mathsf{n}}_1$ and $\tilde{\phi}, \tilde{u}^{\tilde{\alpha}}_1, \tilde{u}^{\mathsf{n}}_1 \sim \tilde{\phi}, \tilde{u}^{\tilde{\alpha}}_1, \tilde{u}^{\tilde{\gamma}}_1$ are dealt with exactly like we did for $\phi, u^\alpha_1, u^\beta_1, u^\gamma_1 \sim \phi, u^\alpha_1, u^{\mathsf{n}}_1, u^\gamma_1$, introducing the corresponding conditional tests. We do not detail these two cases, but notice that the right injectivity of c is needed for them.

We now need to show that $\phi, u^\alpha_1 \sim \tilde{\phi}, \tilde{u}^{\tilde{\alpha}}_1$, which is done by applying the FA axiom several time:

$$\frac{\dfrac{\phi_2, \mathsf{n}'_R, \mathsf{n}'_T, \mathsf{H}(\mathsf{c}(g(\phi_2), \mathsf{n}'_T), \mathsf{k_A}) \sim \phi_2, \mathsf{n}'_R, \mathsf{n}'_T, \mathsf{H}(\mathsf{c}(g(\phi_2), \mathsf{n}'_T), \mathsf{k_B})}{\phi_3 \sim \tilde{\phi}_3}\, FA^*}{\phi, u^\alpha_1 \sim \tilde{\phi}, \tilde{u}^{\tilde{\alpha}}_1}\, FA^*$$

Let $\psi \equiv \phi_2, \mathsf{n}'_R, \mathsf{n}'_T$, it is then easy to show that $\psi, \mathsf{H}(\mathsf{c}(g(\phi_2), \mathsf{n}'_T), \mathsf{k_A}) \sim \psi, \mathsf{H}(\mathsf{c}(g(\phi_2), \mathsf{n}'_T), \mathsf{k_A})$ is derivable using the fact that n'_T is fresh in ψ, the right injectivity of c and the PRF axiom.

- **Case** $i = 2$ **and** $i = 3$**:** These case are very similar to the case $i = 1$, except that we need to use the Dup axiom at some point to get rid of the double occurrence of γ (in case $i = 2$) or α (in case $i = 3$).
- **Case** $i = 4$**:** Using Proposition 7 we know that

$$e_4 = \mathsf{if}\ \epsilon_4\ \mathsf{then}\ \big(\mathsf{if}\ \epsilon'_4\ \mathsf{then\ true\ else\ false}\big)\ \mathsf{else\ false}$$
$$e_5 = \mathsf{if}\ \epsilon_5\ \mathsf{then}\ \big(\mathsf{if}\ \epsilon'_5\ \mathsf{then\ true\ else\ false}\big)\ \mathsf{else\ false}$$

Since booleans $\epsilon'_4 \equiv \mathsf{EQ}\big(\pi_1(g(\phi_3)); \mathsf{n}'_T\big)$ and $\epsilon_5 \equiv \mathsf{EQ}\big(\mathsf{n}'_R; \pi_1(g(\phi_3))\big)$ we have:

$$\begin{aligned}
&\mathsf{if}\ \epsilon'_4\ \mathsf{then}\ \big(\mathsf{if}\ \epsilon_5\ \mathsf{then\ true\ else\ false}\big)\ \mathsf{else\ false}\\
&= \mathsf{if}\ \epsilon'_4\ \mathsf{then}\ \begin{pmatrix}\mathsf{if}\ \mathsf{EQ}\big(\mathsf{n}'_R; \mathsf{n}'_T\big)\ & \mathsf{then\ true}\\ & \mathsf{else\ false}\end{pmatrix}\ \mathsf{else\ false}\\
&= \mathsf{if}\ \epsilon'_4\ \mathsf{then}\ \big(\mathsf{if\ false\ then\ true\ else\ false}\big)\ \mathsf{else\ false}\\
&= \mathsf{false}
\end{aligned}$$

$$\alpha \equiv \langle \mathsf{n}'_T\,,\, \mathsf{H}(\mathsf{c}(g(\phi_2), \mathsf{n}'_T), \mathsf{k_A})\rangle$$
$$\beta \equiv \mathsf{H}\big(\mathsf{c}(\mathsf{n}'_R, \pi_1(g(\phi_3))), \mathsf{k_A}\big)$$
$$\gamma \equiv \mathsf{H}\big(\mathsf{c}(\pi_2(g(\phi_3)), \mathsf{n}'_R), \mathsf{k_A}\big)$$

$\epsilon_1 \equiv \mathsf{EQ}\big(\mathsf{n}'_R; g(\phi_0)\big)$	$e_1 \equiv \mathsf{EQ}\big(\mathsf{c}(\mathsf{n}'_R, \pi_1(g(\phi_3))); \mathsf{c}(g(\phi_0), \mathsf{n}_T)\big)$	(in term $s^A_{\phi_0}$)
$\epsilon_2 \equiv \mathsf{EQ}\big(\mathsf{n}'_R; \mathsf{n}_R\big)$	$e_2 \equiv \mathsf{EQ}\big(\mathsf{c}(\mathsf{n}'_R, \pi_1(g(\phi_3))); \mathsf{c}(\mathsf{n}_R, \pi_1(g(\phi_1)))\big)$	(in term $t^A_{\phi_1}$)
$\epsilon_3 \equiv \mathsf{EQ}\big(\mathsf{n}'_R; \pi_2(g(\phi_1))\big)$	$e_3 \equiv \mathsf{EQ}\big(\mathsf{c}(\mathsf{n}'_R, \pi_1(g(\phi_3))); \mathsf{c}(\pi_2(g(\phi_1)), \mathsf{n}_R)\big)$	(in term $t^A_{\phi_1}$)
$\epsilon_4 \equiv \mathsf{EQ}\big(\mathsf{n}'_R; g(\phi_2)\big)$ $\epsilon'_4 \equiv \mathsf{EQ}\big(\pi_1(g(\phi_3)); \mathsf{n}'_T\big)$	$e_4 \equiv \mathsf{EQ}\big(\mathsf{c}(\mathsf{n}'_R, \pi_1(g(\phi_3))); \mathsf{c}(g(\phi_2), \mathsf{n}'_T)\big)$	(in term $s^A_{\phi_2}$)
$\epsilon_5 \equiv \mathsf{EQ}\big(\mathsf{n}'_R; \pi_1(g(\phi_3))\big)$ $\epsilon'_5 \equiv \mathsf{EQ}\big(\mathsf{n}'_R; \pi_2(g(\phi_3))\big)$	$e_5 \equiv \mathsf{EQ}\big(\mathsf{c}(\mathsf{n}'_R, \pi_1(g(\phi_3))); \mathsf{c}(\pi_2(g(\phi_3)), \mathsf{n}'_R)\big)$	(in term $t^A_{\phi_3}$)

Fig. 10. Term Definitions for the LAK$^+$ Unlinkability Proof

Proof Tree P_1:

$$\dfrac{\dfrac{\phi, u_1^\alpha, u_1^\gamma \sim \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}, \tilde{u}_1^{\tilde{\gamma}}}{\phi, u_1^\alpha, \mathsf{n}, u_1^\gamma \sim \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}, \mathsf{n}, \tilde{u}_1^{\tilde{\gamma}}}\, FreshNonce}{\phi, u_1^\alpha, u_1^{\mathsf{n}}, u_1^\gamma \sim \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}, \tilde{u}_1^{\mathsf{n}}, \tilde{u}_1^{\tilde{\gamma}}}\, FA^* \qquad \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}, \tilde{u}_1^{\mathsf{n}}, \tilde{u}_1^{\tilde{\gamma}} \sim \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}, \tilde{u}_1^{\tilde{\beta}}, \tilde{u}_1^{\tilde{\gamma}}}{\phi, u_1^\alpha, u_1^\beta, u_1^\gamma \sim \phi, u_1^\alpha, u_1^{\mathsf{n}}, u_1^\gamma \qquad \phi, u_1^\alpha, u_1^{\mathsf{n}}, u_1^\gamma \sim \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}, \tilde{u}_1^{\tilde{\beta}}, \tilde{u}_1^{\tilde{\gamma}}}{\phi, u_1^\alpha, u_1^\beta, u_1^\gamma \sim \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}, \tilde{u}_1^{\tilde{\beta}}, \tilde{u}_1^{\tilde{\gamma}}}\, Trans}\, Trans$$

Proof Tree P_2:

$$\dfrac{\dfrac{\dfrac{\phi, u_1^\alpha \sim \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}}{\phi, u_1^\alpha, \mathsf{n} \sim \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}, \mathsf{n}}\, FreshNonce}{\phi, u_1^\alpha, u_1^{\mathsf{n}} \sim \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}, \tilde{u}_1^{\mathsf{n}}}\, FA^* \qquad \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}, \tilde{u}_1^{\mathsf{n}} \sim \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}, \tilde{u}_1^{\tilde{\gamma}}}{\phi, u_1^\alpha, u_1^\gamma \sim \phi, u_1^\alpha, u_1^{\mathsf{n}} \qquad \phi, u_1^\alpha, u_1^{\mathsf{n}} \sim \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}, \tilde{u}_1^{\tilde{\gamma}}}{\phi, u_1^\alpha, u_1^\gamma \sim \tilde{\phi}, \tilde{u}_1^{\tilde{\alpha}}, \tilde{u}_1^{\tilde{\gamma}}}\, Trans}\, Trans$$

Fig. 11. Derivations P_1 and P_2

And therefore, for all term v we have $u_4^v = 0$. Similarly we have $\tilde{u}_4^v = 0$ This means that we have:

$$\dfrac{\dfrac{\phi_3 \sim \tilde{\phi}_3}{\phi, 0, 0, 0 \sim \tilde{\phi}, 0, 0, 0}\, FA}{\phi, u_4^\alpha, u_4^\beta, u_4^\gamma \sim \tilde{\phi}, u_4^{\tilde{\alpha}}, u_4^{\tilde{\beta}}, u_4^{\tilde{\gamma}}}\, Congr$$

We already showed in the case $i = 1$ that $\phi_3 \sim \tilde{\phi}_3$ is derivable.