# Provably secure grouping-proofs for RFID tags*

Mike Burmester[1], Breno de Medeiros[2], and Rossana Motta[1]

[1] Florida State University, Tallahassee FL 32306, USA
{burmester,motta}@cs.fsu.edu
[2] Google Inc., 1600 Amphitheatre Pkwy Mountain View, CA 94043
breno@google.com

**Abstract.** We investigate an application of RFIDs referred to in the literature as *group scanning*, in which several tags are "simultaneously" scanned by a reader device. Our goal is to study the group scanning problem in strong adversarial models. We present a security model for this application and give a formal description of the attending security requirements, focusing on the privacy (anonymity) of the grouped tags, and/ or forward-security properties. Our model is based on the Universal Composability framework and supports re-usability (through modularity of security guarantees). We introduce novel protocols that realize the security models, focusing on efficient solutions based on off-the-shelf components, such as highly optimized pseudo-random function designs that require fewer than 2000 Gate-Equivalents.

## 1 Introduction and previous work

Radio Frequency Identification (RFID) tags were initially developed as small electronic hardware components whose main function is to broadcast a unique identifying number upon request. The simplest type of RFID tags are *passive* devices—i.e., without an internal power source of their own, relying on an antenna coil to capture RF power broadcast by an RFID reader. In this paper, we focus on tags that additionally feature a basic integrated circuit and memory. This IC can be used to process challenges issued by the RFID reader and to generate an appropriate response. For details on these tags, and more generally on the standards for RFID systems, the reader is referred to the Electronic Protocol Code [10] and the ISO 18000 standard [11].

The low cost and high convenience value of RFID tags gives them the potential for massive deployment. Accordingly, they have found increased adoption in manufacturing (assembly-line oversight), supply chain management, inventory control, business automation applications, and in counterfeit prevention. Initial designs of RFID identification protocols focused on performance issues with lesser attention paid to resilience and security. As the technology has matured and found application into high-security and/or high-integrity settings, the need for support of stronger security features has been recognized. Many works have looked into the issue of secure identification and authentication, including [1–3, 6–9, 13, 14, 18–24].

* Part of this material is based on work supported by the U.S. Army Research Laboratory, and the U.S. Research Office under grant number DAAD 19-02-1-0235.

Ari Juels introduced the security context of a new RFID application—which he called a yoking-proof [12], that involves generating evidence of simultaneous presence of two tags in the range of an RFID reader. As noted in [12], interesting security engineering challenges arise in regards to yoking-proofs when the trusted server (or Verifier) is not online during the scan activity. The first proposed protocol introduced in [12] was later found to be insecure [17, 5]. Yoking-proofs have been extended to *grouping-proofs* in which groups of tags prove simultaneous presence in the range of an RFID reader—see e.g. [17, 16, 5]. In this paper, we examine the latter solutions and identify similar weaknesses in their design.

Our main contribution in this paper is to present a comprehensive security framework for RFID grouping-proofs, including a formal description of the attending security requirements. In previous work, the group scanning application has only been described at relatively informal levels, making it difficult to provide side-to-side comparisons between alternative proposals. We then construct practical solutions guided by the security requirements and constraints of this novel model.

As Juels already pointed out, there are several practical scenarios where grouping-proofs could substantially expand the capabilities of RFID-based systems. For example, some products may need to be shipped together in groups and one may want to monitor their progress through the supply chain—e.g., of hardware components or kits. Other situations include environments that require a high level of security, such as airports. In this case, it may be necessary to couple an identifier, such as an electronic passport, with a physical person or with any of his/her belongings, such as their bags. In battlefield contexts, weaponry or equipment may have to be linked to specific personnel, so that it may only be used or operated by the intended users.

In some of the above scenarios, the RFID reader may not enjoy continuous connectivity with the trusted Verifier, and delayed confirmation may be acceptable. For instance, this may be the case with supply chain applications, due to the increased fragmentation and outsourcing of manufacturing functions. A supplier of partially assembled kits may perform scanning activities that will be verified later when the kits are assembled at a different site. Therefore, efficient and optimized realizations of this primitive that achieve strong security guarantees—such as we describe in this paper—are practically relevant contributions in the design space of RFID protocols.

## 2 RFID deployments and threat model

A typical deployment of an RFID system involves three types of legitimate entities: *tags*, *readers* and a *Verifier*. The tags are attached to, or embedded in, objects to be identified. In this paper we focus on passive RFID tags that have no power of their own but have a small footprint CMOS integrated circuit, ROM, RAM and non-volatile EEPROM. The RFID readers typically contain a *transceiver*, a *control unit* and a *coupling element*, to interrogate tags. They implement a radio interface to the tags and a high level interface to the Verifier that processes captured data.

The Verifier (a back-end server) is a trusted entity that maintains a database containing the information needed to identify tags, including their identification numbers. In our protocols, since the integrity of the whole RFID system is entirely dependent on

the proper behavior of the Verifier, we assume that the Verifier is physically secure and not attackable.

Grouping-proofs involve several tags being scanned by an RFID reader in the same session. The reader establishes a communication channel that links the tags of a group and enables the tags to generate a proof of "simultaneous presence" within its broadcast range. The proof should be verifiable by the Verifier. Throughout this paper, we assume the following about the environment characterizing group scanning applications:

- *The tags are passive,* i.e., have no power of their own , and have very limited computation and communication capabilities. However, we assume that they are able to perform basic cryptographic operations such as generating pseudo-random numbers and evaluating pseudo-random functions.
- *RFID tags do not maintain clocks or keep time.* However, the activity time span of a tag during a single session can be limited using techniques such as measuring the discharge rate of capacitors, as described in [12].
- *RFID readers establish communication channels* that link the tags of a group. This takes place at the data link layer of the RFID network: after tags that claim to belong to a group are "identified" (tags may use pseudonyms) a common (wireless) channel linking the tags via the reader is established.
- *RFID readers are trusted to manage the interrogation of tags.* They enable the tags of a group to generate a grouping proof during an interrogation session, and keep a record of such proofs for each session. These records cannot be manipulated by the adversary. In the offline case readers must also store private information regarding interrogation challenges obtained from the Verifier.
- *The Verifier is a trusted entity,* that may share some secret information with the tags such as cryptographic keys. The Verifier has a secure channel (private and authenticated) that links it to the (authenticated) RFID readers.
- *Grouping proofs are only valid if they are generated according to their protocol in the presence of an authorized RFID reader.* In particular if the flows of the protocol are ordered, the ordering cannot be violated. Also, proofs generated during different sessions are not valid (even if correct).

The Verifier can be online or offline and different solutions are required in each case. We further distinguish between online *fully-interactive* mode and online *batch* mode. In fully-interactive mode the Verifier can receive and send messages to specifics tags throughout the protocol execution. In contrast, the interaction of the Verifier in batch mode is restricted to broadcasting a challenge that is valid for a (short) time span, collecting responses from the tags (via RFID reader intermediates), and checking for legitimate group interactions—the Verifier in batch mode never unicasts messages to particular groups of tags.

It is straightforward to design solutions for the fully-interactive mode of the grouping-proof problem—indeed, it is sufficient for individual tags to authenticate themselves to the Verifier, which will then decide on the success of the grouping-proof by using auxiliary data, e.g., the tag identifiers of the groups. Therefore, research on grouping-proofs has focused on the offline case, with some results also targeted at the online batch modality. Accordingly, in this paper, we focus on offline solutions, except for the forward-secure protocol, where we only describe a solution in the online batch mode.

### 2.1 Attacks on RFID tags

Several types of attacks against RFID systems have been described in the literatrure. While each of these are types known in other platforms, unique aspects of the RFID domain make it worthwhile to discuss them anew.

- *Denial-of-Service* (DoS) attacks: The adversary causes tags to assume a state from which they can no longer function properly.
- *Unauthorized tag cloning*: The adversary captures keys or other tag data that allow for impersonation.
- *Unauthorized tracing*: The adversary should not be able to trace and/or recognizes tags.
- *Replay attacks*: The adversary uses a tag's response to a reader's challenge to impersonate the tag.
- *Interleaving and reflection attacks*: These are *concurrency* attacks in which the adversary combines flows from different instantiations to get a new valid transcript.

These attacks are exacerbated by the mobility of the tags, allowing them to be manipulated at a distance by covert readers.

### 2.2 The threat model for RFID

The extremely limited computational capabilities of RFID tags imply that traditional multi-party computation techniques for securing communication protocols are not feasible, and that instead lightweight approaches must be considered. Yet the robustness and security requirements for RFID applications can be quite significant. Ultimately, security solutions for RFID applications must take as rigorous a view of security as other types of applications. Accordingly, our threat model assumes a Byzantine adversary. In this model all legitimate entities (tags, readers, the Verifier) and the adversary have polynomially bounded resources. The adversary controls the delivery schedule of the communication channels, and may eavesdrop into, or modify, their contents, and also instantiate new channels and directly interact with honest parties.

We are mainly concerned with security issues at the protocol layer and not with physical or link layer issues—For details on physical/link layer issues the reader is referred to [10, 11].

### 2.3 Guidelines for secure RFID applications

Below we present effective strategies that can be used to thwart the attacks described in Section 2.1. These strategies are incorporated in the design of our protocols.

- *DoS attacks*: One way to prevent the adversary from causing tags to assume an unsafe state is by having each tag share with the Verifier a permanent secret key $k_{tag}$, which the tag uses to generate a response when challenged by an RFID reader.
- *Cloning attacks*: The Verifier should be able to check a tag's response, but the adversary should not be able to access a tag's identifying data. This can be assured by using cryptographic one-way functions.

– *Unauthorized tracing*: The adversary should not be able to link tag responses to particular tags. This can be guaranteed by (pseudo-)randomizing the values of the tags' responses.
– *Interleaving and Replay attacks*: The adversary should not be able to construct valid transcripts by combining flows from different sessions. This can be assured by binding all messages in a session to the secret key and to fresh (pseudo-)random values.
– *Generic concurrency-based attacks:* Protocols that are secure in isolation may become vulnerable under concurrent execution (with other instances of itself or of other protocols). To guarantee security against such attacks it is necessary to model security in a concurrency-aware model. In this paper, we use the Universal Composability model.

## 3 Previous Work: RFID Grouping-Proofs

In this section we describe three grouping-proofs proposed in the literature and discuss their vulnerabilities.

*3.1 The Yoking-proof.* This is a proof of simultaneous presence of two tags in the range of a reader [12]. The reader scans the tags sequentially. The tags have secret keys known to the Verifier but not the reader, and counters, and use a keyed message authentication code and a keyed hash function to compute a "*yoking-proof*". Saito and Sakurai observed [17] that a minimalist version of this proof (that does not use counters) is subject to an interleaving attack. The attack was shown [5] to extend to the full version of the proof, but it was also shown that it can be easily be prevented.

There are two other weaknesses we shall discuss here. The first concerns the fact that the tags do not (and cannot) check each other's computation. This implies that in the offline mode unrelated tags can participate in a yoking session, and that the failure will only be detected by the Verifier at some later time, not by the reader. While, from an authentication perspective, this may not represent a security threat, in many practical applications it is an undesirable waste of resources, and could be characterized as a DoS vulnerability. To appreciate how accidental pairing may create challenges to real-world applications—e.g., where yoking is used to ensure that components are grouped in a shipment, consider the following scenario. A reader is configured to take temporary measures after a failed yoking attempt, e.g., notify an assembly worker of a missing component in a shipment pallet. This capability is denied if a tag (either accidentally or maliciously) engages in yoking sessions with unrelated tags, and possibly even with itself—for the latter, we refer the reader to the modified re-play attack scenario described in [17]. Accidental occurrences of this type might not be unlikely, in particular with anonymous yoking-proofs, and they are facilitated by the fact that the scanning range of readers may vary according to different environmental conditions. In order to prevent this kind of vulnerability, in our protocols we use a group secret key $k_{group}$, which is shared by all the tags belonging to that group.[3]

[3] Although group keys will prevent faulty tags from participating in a grouping-proof that involves non-faulty tags, they cannot prevent *malicious* tags from submitting an invalid proof to

A more serious weakness concerns the nature of the "proof" $P_{AB}$ generated by the tags: this is *not* a proof that $tag_A$ and $tag_B$ were scanned simultaneously while in the presence of an authorized reader. Indeed, one cannot exclude the possibility that $P_{AB}$ was generated while the tags were in the presence of a rogue reader, and that at a later time $P_{AB}$ was replayed by a corrupted tag (impersonating successively $tag_A$ and $tag_B$) in the presence of the authorized reader. To avoid this kind of attack in our protocols the challenge of authorized tags will include a nonce ($r_{sys}$).

*3.2 Proofs for multiple RFID tags.* These extend yoking-proofs to handle arbitrary number of tags in a group [17] and use time-stamps, to thwart re-play attacks. Piramuthu [16] replaced the time-stamps by random numbers. This is important, because time-stamps can be predicted, allowing for attacks that collect prior responses and combine them to forge proofs of simultaneous interaction. As with the yoking-proofs, these fail to satisfy the security guidelines in Section 2.3. In particular, the random numbers used and are vulnerable to a multi-proof session attacks [15].

*3.3 Clumping-proofs for multiple RFID tags* [15]. These combine the strengths of yoking-proofs and multiple tag proofs and address some of their weaknesses. The tags use counters and the reader uses a keyed hash of a time-stamp, obtained from the Verifier, to make its requests unpredictable. For details, we refer the reader to [15].

Clumping-proofs use counters to reduce the search complexity of the Verifier. However their value is updated regardless of the received flows, so they can be incremented arbitrarily by the adversary (via rogue readers). Therefore, they cannot be relied upon to identify tags, and in the worst case an exhaustive search through the keys may have to be used. A security proof is provided in the Random Oracle Model [4]. However, this does not address concurrency threats, a substantial limitation of the analysis, considering that the original yoking-proofs [12] admit a similar security proof and are vulnerable to concurrency-based attacks.

## 4 Our Protocols: Robust grouping-proofs

We present three RFID grouping proofs. The first one does not provide anonymity, the second adds support to anonymity and the third improves on the second by incorporating forward-secrecy.

In the first protocol, the proof sent from the tags to the reader and from the reader to the Verifier includes a group identifier $ID_{group}$. For the second protocol, no identifier is passed to the reader: the proof uses values that depend on the group's identifier and key and on the Verifier's challenge but the dependency is known only to the Verifier. Thus, only the Verifier is able to match the proof with a given group of tags: this guarantees unlinkability and anonymity. In the third protocol the secret keys and the group keys of the tags are updated after each execution, thus providing forward-secrecy.

There are two reasons why we present different protocols. First, prior work on group scanning has considered both the anonymous and non-anonymous settings. Since

a reader, since proofs can only be verified by the Verifier. Our last protocol (Section 4.3), in which the groups are authenticated by the reader, addresses this issue.

anonymizing protocols requires additional computational steps and correspondingly larger tag circuitry, simpler alternatives are preferred whenever anonymity is not a concern. Second, the introduction of protocols of increasing complexity follows a natural progress that facilitates the understanding of the protocols structure.

Although for simplicity we illustrate our protocols with two tags, the extension to any number of tags is straightforward. Irrespective of the number of tags involved, a specific tag in the group always plays the role of "initiator," transmitting either a counter (in the non-anonymous protocol), a random number, or a random password (in the other versions). This has the security benefit of curtailing reflection attacks. To implement this feature, it is not necessary that tags engage in any sort of real-time agreement protocol, it is sufficient to hard-code the behavior of tags.

We consider situations in which the Verifier is not online while the tags are scanned. Each tag stores in non-volatile memory two secret keys (both shared with the Verifier): a *group key* $k_{group}$ used to prove membership in a group, and an *identification key* $k_{tag}$ used to authenticate protocol flows. Tags instances are denoted as $tag_A$ or $tag_B$, and the key for instance $tag_A$ is written in shorthand as $k_A$.

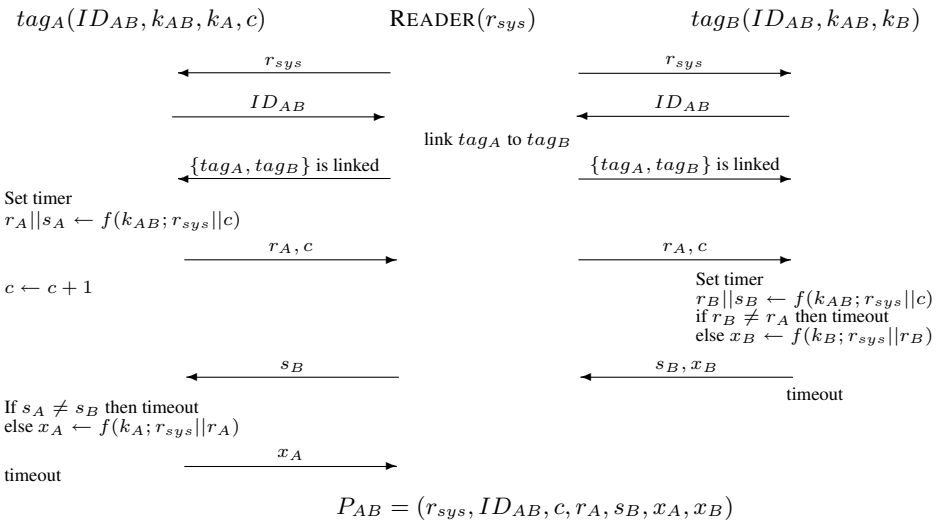
Each protocol starts with a reader broadcasting a random challenge $r_{sys}$, which is obtained from the trusted Verifier at regular intervals. This challenge defines the scanning period, i.e., each group should be scanned at most once between consecutive challenge values. In other words, the Verifier cannot (without further assumptions) determine simultaneity of a group scan to a finer time interval than the scanning period.

### 4.1 A robust grouping-proof

Our first non-anonymous grouping-proof is presented for two tags, $tag_A$ and $tag_B$, where $tag_A$ is the initiator tag—see Fig. 1. The current state of the group is determined by a counter $c$ stored by the initiator tag. The counter is updated with each execution of the protocol. Each group is assigned an identifier $ID_{group}$ and the Verifier stores these values together with the private keys of each tag in a database $D = \{(ID_{tag}, k_{tag}, k_{group})\}$. The protocol has three phases. In the first phase the reader challenges the tags in its range with $r_{sys}$ and the tags respond with their group identifier $ID_{AB}$. In the second phase—which takes place at the data-link layer—the tags are linked by channels through the reader. In the third, the tags prove membership in their group.

Each phase can be executed concurrently with all the tags in the group, except that the third phase must be initiated by the initiator tag ($tag_A$ in the diagram). The various phases cannot be consolidated without loss of some security feature. If we removed the first phase ($r_{sys}$) the protocol would be subject to a "full-replay" attack (Section 2.3). If we removed the second phase (the exchange of $ID_{AB}$), the reader would be unable to match the tags. Phase three consists of three rounds of communication, and each is crucial to provide the data for the proof. If we were to suppress the exchange of $s_B$ and $x_B$, or if we did not implement the timeout, then replay attacks would be successful. Also, the implementation of the third round enables an authorized reader to detect certain protocol failures immediately, namely those that lead the initiator tag to timeout. The update of the counter $c$ immediately after it is sent by $tag_A$ allows the state

**Fig. 1.** A robust grouping-proof—for two tags




to be updated even if the protocol round should be interrupted. This, along with timers prevents replay attacks.

The extension of the protocol to groups of $n > 2$ tags with identifier $ID_{group}$ is achieved as follows. In the first and second phases, the reader communicates with all tags of $ID_{group}$ concurrently. In the first round of the third phase, the reader communicates only with the initiator tag. In this case the initiator parses $f(K_{group}; r_{sys}||c)$ into $n$ parts $r_A||s_A||s'_A||\dots$ of equal length, and sends $(r_A, c)$. The reader forwards $(r_A, c)$ concurrently to all the other tags of $ID_{group}$ in the second round, to get their responses $(s_B, x_B)$, $(s'_C, x_C)$, $\dots$ and then provides the initiator tag with concatenated answers $s_B||s'_C||\cdots$ from the second round. The initiator checks the correctness of each component $s_B$, $s'_C$, $\dots$, for each other tag of $ID_{group}$; if all are correct the initiator completes the proof by sending $x_A$ to the reader.

In the protocol each $tag_X$ uses its group key $k_{AB}$ to evaluate $f(k_{AB}; r_{sys}||c)$, where $f$ is a pseudo-random function and "||" denotes concatenation. This is parsed to get numbers $r_X, s_X$ of equal length, used to identify the parties of the group and prove membership in the group. Tags use their secret key to confirm correctness of the proof. The proof of simultaneous scanning is $P_{AB} = (r_{sys}, ID_{AB}, c, r_A, s_B, x_A, x_B)$. In our protocol, it is possible for an authorized reader to know whether grouped tags were actually scanned or not because, in the latter case, one or more of the tags would timeout. This represents an improvement over the past protocols, in which the success or failure of the yoking- or grouping-proof could only be detected by the Verifier. This protocol can be implemented very efficiently, with a footprint of fewer than 2000 Gate-Equivalents. For a discussion on optimized implementations of pseudo-random functions suitable for RFID applications, we refer the reader to [24].

**Security analysis.** The universal composability (UC) framework defines the security of a protocol in terms of its simulatability by an idealized functionality $\mathcal{F}$ (which can be thought of as specifications of the achievable security goals for the protocol). $\mathcal{F}$ is a trusted entity that can be invoked by using appropriate calls. We say that a protocol $\rho$ *UC-realizes* $\mathcal{F}$, if for any adversary $\mathcal{A}$, any real-world simulation of $\rho$ in the presence of $\mathcal{A}$ can be emulated by an ideal-world simulation in which the adversary invokes $\mathcal{F}$, in such a way that no polynomial-time environment $\mathcal{Z}$ can distinguish between the two simulations. In ideal-world simulations, the adversary has access to all the outputs of $\mathcal{F}$, as in the real-world it can eavesdrop into all communications.

For our first protocol the functionality $\mathcal{F}_{group}$ comprises the behavior expected of a grouping-proof. It is invoked by five calls: activate, initiate, link, complete, and verify. The first call is used by the environment $\mathcal{Z}$ to activate the system by instantiating the Verifier, an authorized reader and some tags. Note that keys initially shared between the Verifier and the tags are not under control of the adversary in this model—in the UC model this is called a trusted setup. The second call is used by readers to initiate an interrogation session, and corresponds to an $r_{sys}$ challenge, and by tags to declare their group membership. The call link, links the tags specified in activate, and the call initiate for tags gives their response to the reader's challenge. The call complete is for initiator tags and completes a proof: it corresponds to $x_A$. The call verify can be used to submit a putative proof transcript to the Verifier. The adversary can arbitrarily invoke $\mathcal{F}_{group}$ and mediates between all parties involved in its interactions with $\mathcal{F}_{group}$.

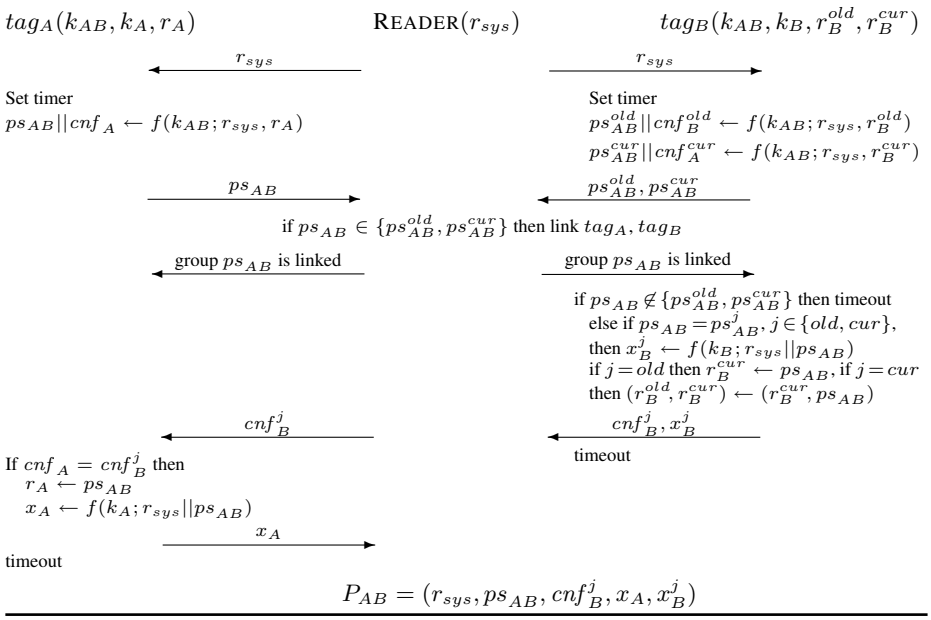
All the outputs resulting from calls to $\mathcal{F}_{group}$, except for the tag calls that produce identifiers, are random strings. The functionality keeps a record of every output string, and uses these strings in the same way as the protocol $\rho$ uses the corresponding outputs. $\mathcal{F}_{group}$ will only accept (verify) those proofs that it has generated itself during a particular session as a result of the activation of the system, the initiation and linking by an authorized reader, the initiation of all the tags that belong to a particular group, and the completion by an initiator tag (in this order). In the full version of this paper we shall show that our first protocol UC-realizes the $\mathcal{F}_{group}$ functionality.

### 4.2 A robust anonymous grouping-proof

For our second protocol, group identifiers are replaced by randomized group pseudonyms $ps_{group}$. To protect against de-synchronization failure or attacks, one (or more in the group of $n > 2$ tags) of the tags must maintain both a current and an earlier version of the state of their pseudonyms. For this purpose all tags in a $group$ store in non-volatile memory one or more values of a pseudo-random number $r_{tag}$.[4] Initiator tags store only the current value, while the other tags store two values, $r_{tag}^{old}$, $r_{tag}^{cur}$. These values are used to compute the group pseudonym. First $f(k_{group}; r_{sys}||r_{tag})$ is evaluated, where $r_{sys}$ is the random challenge of the Verifier. Then, this is parsed to get two numbers $ps_{group}$, $cnf_{tag}$, of equal length, where $cnf_{tag}$ is a confirmation used to authenticate the pseudonym. Initiator tags compute one pseudonym $ps_{group}$; the other tags compute two pseudonyms $ps_{group}^{old}$ and $ps_{group}^{cur}$ (in a similar way).

[4] We use $r_{tag}$ instead of $r_{group}$ to distinguish between the actions of individual tags in $group$ during the execution of the protocol. The values of these numbers are the same for all tags in $group$ when the adversary is passive.

**Fig. 2.** A robust anonymous grouping-proof—for two tags




The tags in $group$ update the value(s) of their group pseudonyms with each successful execution of their part of the grouping protocol. The protocol is presented in Fig. 2, where $tag_A$ is the initiator and for simplicity we depict only one additional tag, $tag_B$. It is easy to see how this protocol can be extended to groups of $n > 2$ tags. In particular, the reader will link all the tags for which at least one pseudonym is $ps_{group}$, provided there are $n$ such tags.

The Verifier keeps a database $D = (r_{sys}, \{(k_{tag}, k_{group}, ps_{group})\})$ that links, for session $r_{sys}$, the secret key of each tag to its group key and the group pseudonym of the corresponding initiator tag. The pseudonyms are updated with each successful execution of the protocol (using the next value of $r_{sys}$). The database $D$ is also used to optimize the performance of the protocol: if the adversary has not challenged the tags of $group$ since their last interaction (e.g., via rogue readers), then the value of the pseudonym in $D$ will be the one that is actually used by the initiator tag, and therefore the corresponding secret keys can be found directly (one lookup) and used to verify the correctness of the authenticator $x_{tag}$ of the initiator tag. The secret keys of the other tags in $group$ can be found in the database $D$ from the group key $k_{group}$, and used to verify the correctness of their authenticators. If no value in $D$ corresponds to the pseudonym used by the initiator tag then the Verifier will have to find the secret key of the initiator from its authenticator $x_{tag} = f(k_{tag}; r_{sys}||ps_{group})$ by exhaustive search over all secret keys (of initiator tags). The pseudo-random numbers $r_{tag}$ are initialized with random values $r_A$: for the initiator $tag_A$: $r_{tag} \leftarrow r_A$, while for all other $tag_X$ in its group: $(r_X^{old}, r_X^{cur}) \leftarrow (r_A, r_A)$.

Observe that initiator tags respond with only one pseudonym and therefore can be distinguished from other tags (which respond with two pseudonyms). There are several ways to address this privacy issue, if it is of concern. One way is to assign to all tags a pair of pseudonyms, and identify groups by selecting those sets of tags that have one pseudonym $ps^*$ in common. There will always be at least one tag in this set for which $ps^{cur}_{group} = ps^*$. The reader elects an initiator tag among those tags sharing the common pseudonym deterministically, probabilistically, or in some ad hoc way: e.g., the first to respond. The reader informs the initiator tag of its selection and indicates to the other tags which pseudonym $ps^*$ is current. In this modification of the protocol all rounds are executed concurrently.

As in the previous protocol, each step is essential. The main difference is that in the anonymous protocol, the tags exchange pseudonyms $ps_{AB}$ and $ps^{old}_{AB}$, $ps^{cur}_{AB}$, rather than a group identifier. The functionality provided by this step, however, is analogous in the two protocols and enables the Verifier to identify the group.

It is important to notice that even though the values that the reader receives for each completed round vary, if a malicious reader interrupts the session (round), preventing the pseudonym update, and then re-uses $r_{sys}$, it can link the two scannings. However, the power of this attack is limited because a single round with a non-faulty reader at any point will restore unlinkability. We shall refer to this property as, *session unlinkability*. More formally we have:

**Definition 1.** *An RFID protocol has* session unlinkability *if, any adversary, given any two tag interrogations $Int_1$, $Int_2$, (not necessarily complete, or by authorized readers), where $Int_1$ takes place before*[5] *$Int_2$, and a history of earlier interrogations, cannot decide (with probability better than 0.5 +$\varepsilon$, $\varepsilon$ negligible) whether these involve the same tag or not, provided that either:*

- *The interrogation $Int_1$ completed normally (successfully), or*
- *An interrogation of the tag involved in $Int_1$ completed successfully after $Int_1$ and before $Int_2$.*

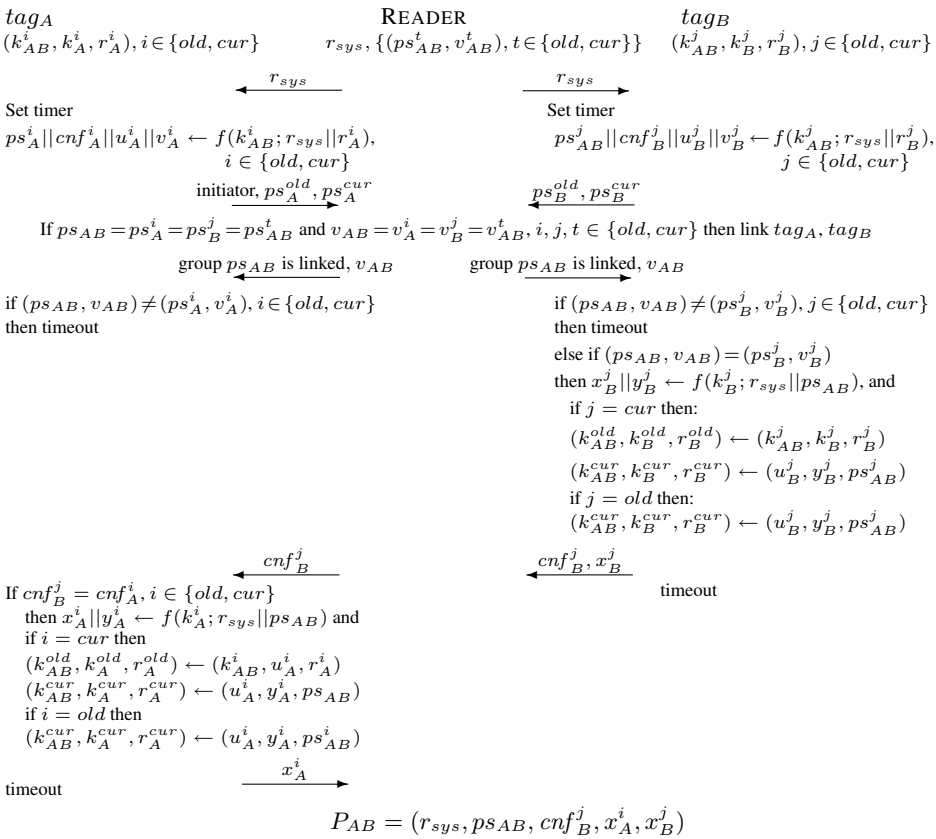
**Security analysis.** The functionality $\mathcal{F}_{sa\_group}$ of our second protocol comprises the behavior expected of an anonymous grouping-proof with session unlinkability. The functionality $\mathcal{F}_{sa\_group}$ is that same as $\mathcal{F}_{group}$ except that:

1. The outputs of all its invocations are random numbers, including tag identifiers.
2. If a tag is initiated with the same reader challenge in a session, as in an earlier session that it was not allowed to complete (and no sessions with this tag completed in the interim), then $\mathcal{F}_{sa\_group}$ will output identical values.

This means that the adversary can link the (uncompleted) scannings throughout a given session. However in the next session, $\mathcal{F}_{sa\_group}$ will use a different (random) number, so linkability does not extend to any other sessions. In the full version of this paper we shall show that our second protocol UC-realizes the $\mathcal{F}_{sa\_group}$ functionality.

[5] A temporal relationship, as observed by the adversary. Note that if the adversary observes two interrogations overlapping in time, it can certainly assert that they do not belong to the same tag, since tags are currently technologically limited to single-threaded execution.

**Fig. 3.** An anonymous grouping-proof with forward-secrecy—for two tags




Notice that our second protocol is not able to provide forward-security: secrecy is no longer guaranteed if the secret keys are compromised.

### 4.3 A robust grouping-proof with forward-secrecy

In our last protocol—see Fig.3, the secret keys and the group keys of tags are updated after each protocol execution for forward-secrecy. All tags, including initiator tags, store two pairs of keys: group keys $k^i_{group}$ and secret keys $k^i_{tag}$, $i \in \{old, cur\}$, as well as a pair of random numbers $r^i_{tag}$, $i \in \{old, cur\}$. The Verifier stores in a database $D$ the current values $(r_{sys}, \{(k^t_{tag}, k^t_{group}, ps^t_{group}), t \in \{old, cur\}\})$: this allows it to link the values of the keys of each tag to the corresponding group pseudonym. At the end of each $r_{sys}$ challenge session, the entries in $D$ of all tags in the groups for which the reader has returned a valid proof $P_{group}$ are updated: $(k^t_{tag}, k^t_{group}, ps^t_{group}) \leftarrow (y^t, u^t, r^t)$, $t \in \{old, cur\}$, using the equal-length parsings $f(k^t_{group}; r_{sys}||r^t_{tag}) = r^t||s^t||u^t||v^t$ and $f(k^t_{tag}; r_{sys}||r^t) = x^t||y^t$ (the use of the other parsed values is explained below).

Since there are no non-volatile values to anchor the key and pseudonym updates to, we shall use the update chain itself as an anchor. This means that the state of the tags and the Verifier must be synchronized. In particular the adversary should not be able to manipulate valid group scans so as to de-synchronize the system. There are several ways in which this can be achieved. The solution we propose is to have the Verifier (a trusted party) give all authorized readers a table $\hat{D} = (r_{sys}, \{(ps^t_{group}, v^t_{group}), t \in \{old, cur\}\})$ whose values can be used to authenticate authorized readers to tags. The entries in this table are obtained by parsing $f(k^t_{group}; r_{sys}, r^t_{tag}) = r^t||s^t||u^t||v^t$ as above, and assigning: $(ps^t_{group}, v^t_{group}) \leftarrow (r^t, v^t)$, $t \in \{old, cur\}$. In this case however the *next* value of the challenge $r_{sys}$ is used in the evaluation of $f$. The values in $\hat{D}$ are updated for *all* groups in the system, at the beginning of each *new* $r_{sys}$ session.

The protocol is given in Fig. 3 for two tags, $tag_A, tag_B$, with $tag_A$ the initiator. In this protocol, adversarial readers cannot disable tags permanently by de-synchronizing them from the Verifier, because the tags discard old key values $k^{old}_{group}, k^{old}_{tag}, r^{old}_{tag}$ only after the Verifier has confirmed that it has updated its corresponding values. More specifically, if the reader is not adversarial, then $ps^{cur}_{AB} = ps^{cur}_{A} = ps^{cur}_{B}$, and the tags will update both current and old key and number values. If the reader is adversarial and has not returned the proof $P_{AB}$ then $ps^{cur}_{AB} \neq ps^{cur}_{A}$, or $ps^{cur}_{B}$, and the updates will not affect old values, which therefore remain the same as those stored in the database $\hat{D}$. The state of the tags will only return to stable when an authorized reader returns a valid proof to the Verifier. Note that due to the state synchronization requirements, the protocol in Fig. 3 can only be implemented in online batch mode, not true offline mode. In the full paper, we discuss the batch offline case.

Forward-secrecy applies to periods during which the groups of tags are scanned by authorized readers that are not faulty. More specifically, a group of tags that is compromised can be traced back to the first interaction after the last non-faulty scanning session, and no further. We shall refer to this property as, *forward session-secrecy*. More formally we have:

**Definition 2.** *An RFID protocol has* forward session-secrecy *if session unlinkability holds for all sessions $Int_1$ and $Int_2$ as in Defn. 1, provided that either $Int_1$ successfully completed prior to the corresponding tag(s) being compromised, or that a later session of the tag(s) involved in $Int_1$ completed successfully prior to its (their) compromise.*

**Security analysis.** The functionality $\mathcal{F}_{fss\_group}$ of our third protocol comprises the behavior expected of an anonymous grouping-proof with forward session-secrecy. The functionality $\mathcal{F}_{fss\_group}$ models key compromise, that is it allows for adaptive corruption. Otherwise it is similar to $\mathcal{F}_{sa\_group}$. This means that the adversary can link incomplete scannings of non-compromised tags throughout a given session, but that this does not extend to other sessions. In the full version of this paper we shall show that our third protocol UC-realizes the $\mathcal{F}_{fss\_group}$ functionality.

## 5 Conclusion

Our main contribution in this paper is to present a security model for the group scanning problem. In previous work, this application has been described at relatively infor-

mal levels, making it difficult to provide side-to-side comparisons between alternative proposals. We have proposed three grouping-proofs that are provably secure in a very strong setting that guarantees security under concurrent executions and provides for safe re-use as a building block for more complex protocols. These proofs are also practically feasible, requiring only pseudo-random functions, which can be instantiated very efficiently in integrated circuits using a variety of primitives as a starting point, such as pseudo-random number generators or block ciphers.

## References


1. G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable RFID tags via insubvertible encryption. In *Proc. ACM Conf. on Computer and Communication Security (ACM CCS 2005)*, pages 92–101. ACM Press, 2005.
2. G. Avoine and P. Oechslin. A scalable and provably secure hash based RFID protocol. In *Proc. IEEE International Workshop on Pervasive Computing and Communication Security (PerSec 2005)*. IEEE Computer Society Press, 2005.
3. Gildas Avoine and Philippe Oechslin. A scalable and provably secure hash-based RFID protocol. In *Proc. IEEE Intern. Conf. on Pervasive Computing and Communications (PerCom 2005)*, pages 110–114. IEEE Press, 2005.
4. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.
5. L. Bolotnyy and G. Rose. Generalized "Yoking-Proofs" for a group of Radio Frequency Identification Taga. In *International Conference on Mobile and Ubiquitous Systems, MOBIQUITOUS 2006*, San Jose, CA, 2006.
6. Stephen C. Bono, Matthew Green, Adam Stubblefield, Ari Juels Aviel D. Rubin, and Michael Szydlo. Security analysis of a cryptographically-enabled RFID device. In *Proc. USENIX Security Symposium (USENIX Security 2005)*, pages 1–16. USENIX, 2005.
7. M. Burmester, T. van Le, and B. de Medeiros. Provably secure ubiquitous systems: Universally composable RFID authentication protocols. In *Proceedings of the 2nd IEEE/CreateNet International Conference on Security and Privacy in Communication Networks (SECURECOMM 2006)*. IEEE Press, 2006.
8. Tassos Dimitriou. A lightweight RFID protocol to protect against traceability and cloning attacks. In *Proc. IEEE Intern. Conf. on Security and Privacy in Communication Networks (SECURECOMM 2005)*. IEEE Press, 2005.
9. Tassos Dimitriou. A secure and efficient RFID protocol that can make big brother obsolete. In *Proc. Intern. Conf. on Pervasive Computing and Communications, (PerCom 2006)*. IEEE Press, 2006.
10. EPC Global. EPC tag data standards, vs. 1.3. http://www.epcglobalinc.org/standards/EPCglobal_Tag_Data_Standard_TDS_Version_1.3.pdf.
11. ISO/IEC. Standard # 18000 – RFID air interface standard. http://www.hightechaid.com/standards/18000.htm.
12. Ari Juels. "Yoking-Proofs" for RFID tags. In *PERCOMW '04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, pages 138–142, Washington, DC, USA, 2004. IEEE Computer Society.
13. David Molnar, Andrea Soppera, and David Wagner. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In *Proc. Workshop on Selected Areas in Cryptography (SAC 2005)*, volume 3897 of *LNCS*. Springer, 2006.

14. Y. Oren and A. Shamir. Power analysis of RFID tags. Invited talk, RSA Conference, Cryptographer's Track (RSA-CT 2006). Available at http://www.wisdom.weizmann.ac.il/ yossio/rfid, 2006.
15. P. Peris-Lopez, J.C. Hernandez-Castro, J.M. Estevez-Tapiador, and A. Ribagorda. Solving the simultaneous scanning problem anonymously: clumping proofs for RFID tags. In *Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, SecPerl07*, Istambul, Turkey, 2007. IEEE Computer Society Press.
16. Selwyn Piramuthu. On existence proofs for multiple RFID tags. In *IEEE International Conference on Pervasive Services, Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing – SecPerU 2006*, Lyon, France, June 2006. IEEE, IEEE Computer Society Press.
17. J. Saito and K. Sakurai. Grouping proof for RFID tags. In *19th International Conference on Advanced Information Networking and Applications, 2005. AINA 2005*, volume 2, pages 621–624, March 2005.
18. S. Sarma, S. Weis, and D. Engels. RFID systems and security and privacy implications. In *Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002)*, LNCS, pages 454–470. Springer, 2002.
19. S.Engberg, M. Harning, and J.C. Damgøard. Zero-knowledge device authentication: Privacy & security enhanced RFID preserving business value and consumer convenience. In *Proc. Conf. on Privacy, Security, and Trust (PST 2004)*, 2004.
20. S. E. Sharma, S. A. Wang, and D. W. Engels. RFID systems and security and privacy implications. In *Proc. of the Workshop on Cryptographic Hardware and Embedded Systems (CHES 20002)*, volume 2523 of *LNCS*, pages 454–469. Springer, 2003.
21. C. Tan, B. Sheng, and Q. Li. Serverless search and authentication protocols for RFID. In *IEEE International Conference on Pervasive Computing and Communications (PerCom 2007)*. IEEE Press, 2007.
22. G. Tsudik. YA-TRAP: Yet another trivial RFID authentication protocol. In *Proc. IEEE Intern. Conf. on Pervasive Computing and Communications (PerCom 2006)*. IEEE Press, 2006.
23. Istvan Vajda and Levente Buttyan. Lightweight authentication protocols for low-cost RFID tags. In *Proc. Workshop on Security in Ubiquitous Computing (UBICOMP 2003)*, 2003.
24. T. van Le, M. Burmester, and B. de Medeiros. Universally composable and forward-secure RFID authentication and authenticated key exchange. In *Proc. of the ACM Symp. on Information, Computer, and Communications Security (ASIACCS 2007)*. ACM Press, 2007.

## A Proof of Security and Security analysis

In the following, we give a sketch for the proof of security of our second protocol, for anonymous RFID grouping-proofs (Section 4.2), in a universally composable framework. Note in particular that we do not consider forward-security in our analysis and moreover that we do not model key compromise. Corruption is therefore non-adaptive: The adversary indicates at the beginning of the protocol which parties are corrupt. The proof for the forward-secure case is deferred to a full version of the paper.

The universal composability (UC) framework defines security in terms of simulatability of protocols by idealized functionalities (which can be thought of as specifications of the achievable security goals for the protocols). The great advantage of the UC model is provided by the composability theorem [**?**]. More precisely, consider an ideal functionality $\mathcal{F}$ that is securely realized by a protocol $\rho$. Let $\pi$ be an arbitrary protocol

that makes ideal calls to multiple instances of $\mathcal{F}$. Let $\pi^\rho$ denote the composed protocol which consists of running the protocol $\pi$ and substituting calls to each instance of $\mathcal{F}$ by calls to a distinct (fresh) instance of $\rho$. The universal composability theorem states that the protocol $\pi^\rho$ has essentially the same effect as protocol $\pi$, even if $\pi^\rho$ has no access to $\mathcal{F}$.

In our case, $\mathcal{F}$ comprises the behavior expected of a yoking- or grouping-proof. $\mathcal{F}$ is secure by specification. $\rho$ represents our protocol, and we wish to prove that it realizes $\mathcal{F}$. $\mathcal{F}$ takes the following roles in the idealized protocol execution:

- Receives the challenges from corrupt readers, or generates them for honest readers;
- Answers challenges on behalf of the honest tags;
- Decides which tags are grouped;
- Collects the data for grouping-proofs from corrupt tags;
- Implements timeouts on honest tags (however, it does not provide punctual time information, i.e., when the scanning occurred).

In order to prove the security of our anonymous grouping-proofs, we now show that each behavior securely provided by $\mathcal{F}$ can be achieved, in the real world, through the protocol. In other words, we simulate the operation of the protocol with access to $\mathcal{F}$ by the real world operation of the protocol (which cannot rely on $\mathcal{F}$).

We summarize key features of our protocols, which represent the real-world run:

- The challenges are represented by $r_{sys}$ and are received from the Verifier through the reader.
- The communication among tags is always mediated by the reader. This means that if a tag wants to talk to one or more other tags, it sends the message to the reader, which forwards it to the proper tag(s). Of course, the reader in question could be corrupt and modify messages, and moreover the adversary can directly modify or interrupt any channels at will—with exception that, if the reader is honest, the adversary cannot tamper with the contents of the channel connecting the reader and the Verifier.
- In order to establish which tags are grouped, tags reply to the reader's challenge, transmitting their group pseudonyms. At this point, the reader is able to match the tags that belong to the same group.
- Timeouts are implemented on tags, through capacitor discharges. The time needed to discharge a capacitor is well known in advance.
- The choice of initiator tag is hard-coded in tags. One, and one only, of the tags belonging to a group is the initiator ($tag_A$).

In the ideal world, honest parties are controlled by an ideal functionality. The main difference is that the values emitted by the ideal functionality $\mathcal{F}$ (as answers of tags) are generated as truly random values, as opposed to pseudo-random. More precisely, whenever $\rho$ evaluates the function $f(\cdot;\cdot,\cdot)$ on a triple $(k_{group}; r_{sys}, r_{tag})$, the functionality $\mathcal{F}$ first checks whether it has a record $(\mathsf{funtion\_value}, k_{group}; r_{sys}, r_{tag}, t)$ in its database. If so, it produces the value $t$ as the result of the function evaluation. If a record is not found, it enters one $(\mathsf{funtion\_value}, k_{group}; r_{sys}, r_{tag}, t)$ in its database, where $t$ is a freshly generated random value, and returns $t$.

We note that the above specification of $\mathcal{F}$ makes several security guarantees obvious: Unforgeability, anonymity, freedom from replays and other types of attacks are achieved because to violate these guarantees the adversary would have to guess unseen random, independently generated values.

Since the protocol may fail in a variety of ways, we must ensure that no combination of failures exists, which may enable a probabilistic, polynomial-time environment $\mathcal{Z}$, which selects the initial inputs and observes the final outputs of all parties in a protocol run, and which may interact with the adversary in an arbitrary fashion during the run, to distinguish (with non-negligible probability) between real and ideal protocol runs. That is the definition of secure simulation in the UC model.

The real and ideal protocol runs could diverge in many ways. We discuss below the significant cases, and why they cannot be used to distinguish between them.

1. A match (successful proof or partial proof among a subset of the tags) occurs in the real-world, while in the ideal-world the match is unsuccessful. This implies that the adversary was able to modify some values in the channels (via reflection, replay, mangling, injection, delay, etc.) and force tags to reproduce pseudo-random values correctly from unequal inputs. However, since the adversary ignores the value of the honest tags' authentication keys, the pseudo-random values observed by the adversary in exchanges are indistinguishable from random and therefore the adversary cannot have a non-negligible probability of forcing such outcome.
2. A mismatch (failed proof attempt) occurs in the real-world, while in the ideal world the same proof (or partial proof involving a subset of the tags) succeeds. This implies that the randomly generated values in the ideal world corresponding to evaluations of the function $f(\cdot;\cdot,\cdot)$ on different triples led to a coincidental match. Since in this case the values are generated by $\mathcal{F}$ independently and at random, the chance of a coincidence is negligible.
3. A non-corrupt tag has a slightly different timeout value that can be used to distinguish it from other tags, violating anonymity. We note that, if such physical measurement attacks are to be considered, and anonymity must be provided, then tags must be manufactured to sufficient precision that such variations cannot be profitably used to distinguish tags.

Among the cases discussed above, the only interesting one is (1), which exploits the definition of pseudo-randomness. In particular, case (1) can be quantified. For instance, the probability of real-ideal distinguishability can be directly related to the strength of the pseudo-random function to resist a pre-specified number of queries. In the full version of this paper we plan to elaborate on the concrete security guarantees provided by the scheme.