Provably secure grouping-proofs for RFID tags

Mike Burmester*, Breno de Medeiros and Rossana Motta

Abstract

In this paper, we investigate an application of RFIDs referred to in the literature as the *group scanning problem*, in which several tags are "simultaneously" scanned by a reader. The security context of this application was first discussed by Ari Juels, who presented a protocol that allows *pairs* of RFID tags to provide evidence of having been simultaneous scanned—a *yoking proof*.

Our goal is to study simultaneous scanning proofs in strong adversarial models. We describe a security model for RFID group scanning proofs, and consider versions of the problem that require privacy (anonymity) of the grouped tags, and/ or forward-security properties. Our security model is based on the Universal Composability framework and supports reusability (through modularity of security guarantees). We also 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 captures RF power broadcast by a reader. In this paper, we focus on EPC Class-1, Gen-2, RFID tags [11] that additionally feature a basic integrated circuit and memory. This IC can be used to process challenges issued by a 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 [11] and the ISO 18000 standard [12].

While admittedly a new technology, 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. RFID tags have also been proposed as mass-market, embedded devices that support ubiquitous applications—for instance, to enable mobile computing appliances to acquire awareness of their environment [18, 27].

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 in particular into the issue of secure identification and authentication, including [1, 2, 3, 5, 6, 8, 9, 16, 17, 21, 10, 22, 23, 24, 25, 26].

*Part of this material is based on work supported by the NSF award 0209092, 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 [13]—that involves generating evidence of simultaneous presence of two tags in the range of a reader. As noted in [13], 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 to implement yoking-proofs, also introduced in [13], was later found to be insecure [20]. Yoking-proofs have been extended to *grouping-proofs* in which groups of tags prove simultaneous presence in the range of a single reader—see e.g. [20, 19]. In this paper, we examine the latter solutions and identify similar weaknesses in their design. We then describe a comprehensive security framework for RFID grouping-proofs, and construct practical solutions guided by the constraints of this novel model. This gives us confidence that our solutions will avoid design pitfalls and stand scrutiny.

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. A different scenario would be to support enforcement of safety regulations requiring that drugs be shipped (or dispensed) accompanied by information leaflets. 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. Similarly, one may want to enforce that access to certain resources only be granted if appropriate groups (or a threshold) of entities/objects are present. 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 indeed be the case with regard to 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 completed at a different site. Moreover, since the grouping-proof problem is a relatively simple primitive, other applications of it are likely to emerge in the future. 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.

The organization of this paper is as follows: A comprehensive adversarial threat model is introduced and discussed in Section 2. In Section 3 we examine the weaknesses of currently available scanning proofs. In Section 4 we propose three RFID proofs for simultaneous group scanning: a basic (non-anonymous) grouping-proof, followed by a version that adds support for anonymity, and an additional version that is both anonymous and forward-secure. In the Appendix, we include proof sketches that the schemes are secure in the Universal Composability framework.

2 RFID deployments and threat model

A typical deployment of an RFID system involves three types of legitimate entities: *tags*, *readers* and a *Verifier* (a back-end server).

The tags are attached to, or embedded in, objects to be identified. They consist of a *transponder* and an *RF coupling element*. The coupling element has an antenna coil to capture RF power, clock pulses and data from the RFID reader. In this paper we focus on passive EPC Gen-2 RFID tags [11], that are smart tags with a CMOS integrated circuit, ROM, RAM and non-volatile EEPROM, but have no power of their own.

The readers typically contain a *transceiver*, a *control unit* and a *coupling element*, to interrogate tags. They implement a radio interface to the tags and also 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. (In principle, it is possible to relax such assumptions and consider mechanisms that prevent the Verifier from collecting user-behavior information, or to make the Verifier auditable. We refer the reader to [22] for approaches in this setting, but do not explore it here.)

Grouping-proofs involve several tags being scanned by a reader in the same session. The reader interrogates the tags either concurrently or sequentially to get 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. They have very limited computation and communication capabilities. However, we assume that they are able to perform basic (symmetric-key) cryptographic operations. Such tags are already commercially available.
- *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 [13].
- *RFID readers establish communication channels* that link them to the tags, and thus indirectly link tags to tags.
- *RFID readers are potentially untrusted.* The only trusted entity is a Verifier, that may share with the tags some secret information, such as cryptographic keys.

The Verifier can be online or offline and different solutions are required in each case. We further distinguish between online *batch* mode and online *fully-interactive* mode. 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 reader intermediates), and checking for legitimate group interactions—the Verifier in batch mode never unicasts messages to particular groups of tags. In contrast, in fully-interactive mode the Verifier can receive and send messages to specifics tags throughout the protocol execution.

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. While each of these are classical types known in other platforms, unique aspects of the RFID domain make it worthwhile to discuss them anew.

1. *Denial-of-Service* (DoS) attacks: The adversary causes tags to assume a state from which they can no longer function properly. Tags become either temporarily or permanently incapacitated.
2. *Unauthorized tag cloning*: The adversary captures keys or other tag data that allow for impersonation of the tag.
3. *Unauthorized tracking*: The adversary tracks and/or recognizes tags through rogue readers.

4. *Replay attacks*: The adversary uses a tag's response to a rogue reader's challenge to impersonate the tag.
5. *Interleaving attacks*: These are *concurrency* attacks in which the adversary succeeds in combining 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

RFID tags are a challenging platform from an information assurance standpoint. Their extremely limited computational capabilities 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. The adversary may also instantiate new channels and directly interact with honest parties. However, we assume that the Verifier is a trusted entity that cannot be corrupted, and that the reader-Verifier channels are secure.

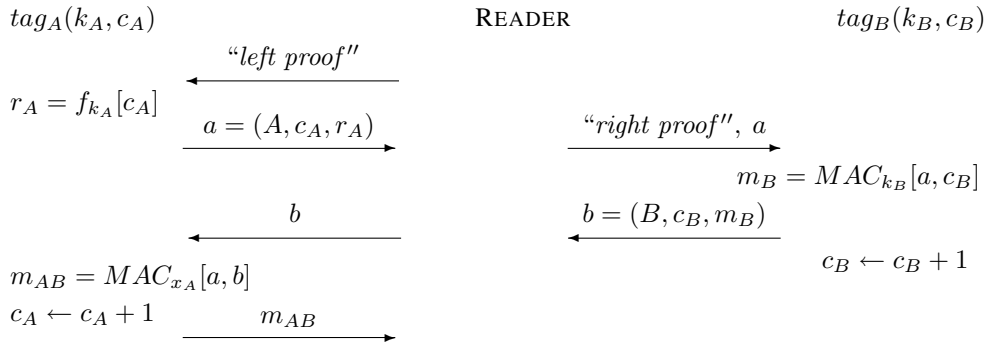
We are mainly concerned with security issues at the protocol layer and not with physical or link layer issues, such as the coupling design, the power-up and collision arbitration processes and the air-RFID interface—For details on physical/link layer issues the reader is referred to [11, 12].

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}. (Assuming here that the cost of public-key cryptography in tags is too high for the application in question, a likely scenario.) When a tag is challenged by a reader it will generate a response using this key.
- *Cloning attacks*: The adversary should not be able to access a tag's identifying data. However the Verifier should be able to check a tag's response. The response must therefore corroborate (but not reveal!) knowledge of the tag's secret data. Of course it should be hard for the adversary to extract secret data from the tag's response. This can be assured by using cryptographic one-way functions.
- *Unauthorized tracking*: The adversary should not be able to link tag responses to particular tags. This can be guaranteed by randomizing the values of the tags' responses. Since all entities in an RFID system are assumed to have polynomially bounded resources, it is sufficient for these values to be cryptographically pseudo-random.
- *Interleaving and Replay attacks*: To prevent the adversary from constructing valid transcripts by combining flows from different sessions, the flows of any particular session should be strongly linked. 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

Figure 1: The yoking-proof



Universal Composability model, which in addition to capturing threats arising from concurrency, allows for secure protocol re-use—e.g., as a building block for more complex applications.

3 RFID grouping-proofs

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

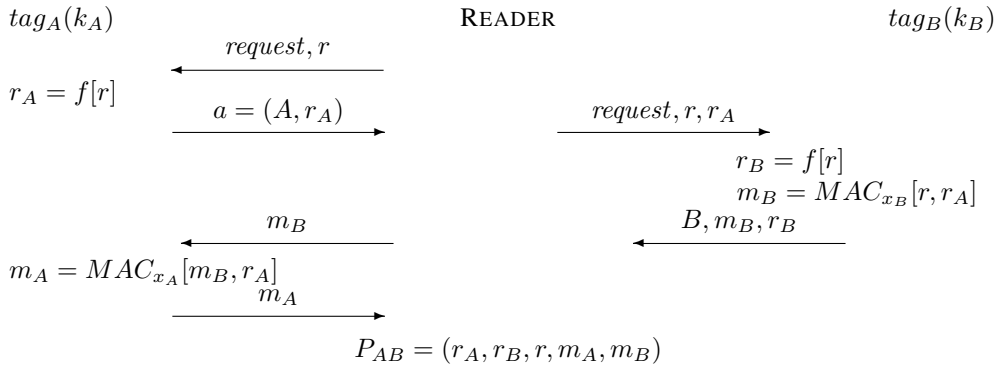
The Yoking-proof [13]. This is a proof of simultaneous presence of two tags tag_A, tag_B in the range of a reader. The tags have secret keys k_A, k_B, known to the Verifier but not the reader, and counters c_A, c_B. The reader scans the tags sequentially. The protocol is described in Fig. 1. $MAC_{(\cdot)}(\cdot)$ is a keyed message authentication code, $f_{(\cdot)}(\cdot)$ a keyed hash function and P_{AB} the resulting *yoking-proof*.

Saito and Sakurai observed that this yoking-proof is subject to an interleaving attack in which the adversary combines flows a, m_{AB} and b from different sessions (obtained by rogue readers) to get a valid proof P_{AB} [20]. An anonymous version (using pseudonyms) of the yoking-proof was proposed in [14]. No security proof is given, and it is not clear how the reader can pair tags from their pseudonyms—a similar problem occurs with the clumping-proofs which will be discussed in some detail below.

Observe that the yoking-proof does not require that tags 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 this may not represent a security threat from an authentication perspective, 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 [20]. Accidental occurrences of this type might not be unlikely, in particular with anonymous yoking-proofs, and they are facilitated by the fact that scanning range of readers may vary according to different environmental conditions (making it difficult to use topological constraints to limit readers to interact with only one pair or group of tags at a time). Ad-

Figure 2: The multiple RFID tag proof



ditionally, undesirable yoking may result simply because the typically small RFID tags are easily misplaced or surreptitiously introduced.

Proofs for multiple RFID tags [20, 19]. These extend the yoking-proofs to handle arbitrary number of tags in a group, where the group of tags generates a proof of having been scanned nearly simultaneously. The first proposal was presented by Saito and Sakurai [20]. It modified the constructions in [13] to include a time-stamp, with a view to thwart re-play attacks as described above.

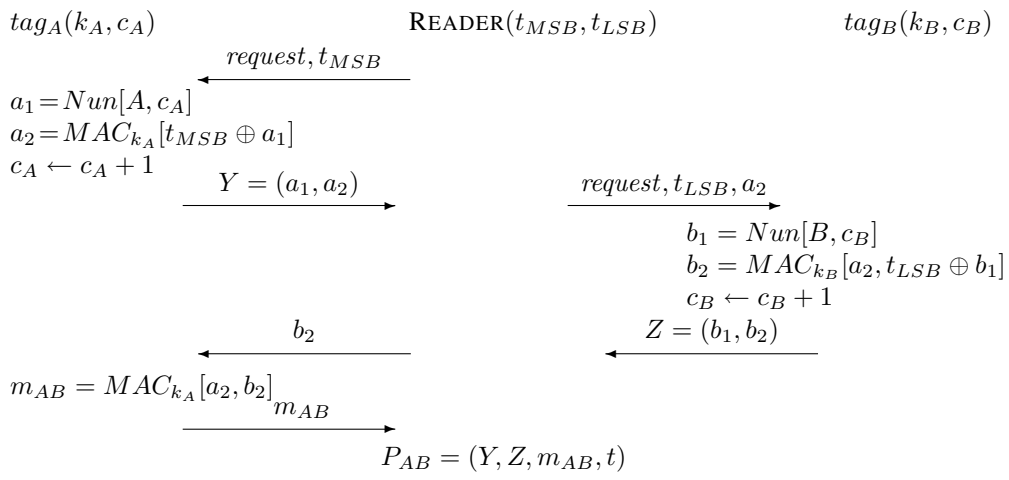
Piramuthu [19] adapted the protocols in [20] to include random values in lieu of time-stamps. This is important, because time-stamps can be predicted, allowing for attacks that collect prior responses and combine them to achieve false proofs of simultaneous interaction. We now describe the approach [19] in more detail (see also Fig. 2). The reader's request to the tags includes a random number r, that the tags use as a seed to generate numbers r_A, r_B. The tags do not use counters. As with the yoking-proofs, these proofs fail to satisfy the security guidelines in Section 2.3. In particular, the numbers r_A and r_B in the flows of the tags are not correlated to the tag's secret authenticating keys.

Clumping-proofs for multiple RFID tags [15]. These proofs, introduced by Lopes et. al., combine the strengths of yoking-proofs and multiple tag proofs and address some of their weaknesses. The tags use counters and the Reader uses the keyed hash $t = g_{k_V}(timestamp)$ of a time-stamp, obtained from the Verifier, to make its requests unpredictable (k_V is a secret key shared by the reader and the Verifier). The Reader parses t as $t_{MSB}||t_{LSB}$, where "||" denotes concatenation, and uses each part in requests to distinct tags. The protocol is illustrated in Fig. 3, and it uses a special-purpose function $Nun[n, m]$ to anonymize tags. The function is lightweight (using only shifts and adds) and satisfies a Strong Avalanche Criterion, to make it more resilient to cryptanalysis. For details, we refer the reader to [15].

Note that the clumping-proofs protocol uses counters to reduce the search complexity of the Verifier. However their value is updated regardless of the received flows, so they can be incremented by the adversary (via rogue readers). Therefore, they cannot be relied upon to identify tags. No alternative method is described to be used by readers to recognize these tags, though we observe that exhaustive search through the keys could be employed in the worst case.

A security proof in the Random Oracle Model [4] of the clumping-proofs protocol is provided in [15]. However, it does not address concurrency threats, a substantial limitation of the analysis, considering that the original yoking-proofs [13] admit a similar security proof and are vulnerable to concurrency-based attacks.

Figure 3: The clumping proof



4 Robust grouping-proofs

We hereby present three protocols, which differ in that the first does not provide either for anonymity or forward-security, the second adds support to anonymity, and the third improves on the second by also incorporating forward-secrecy.

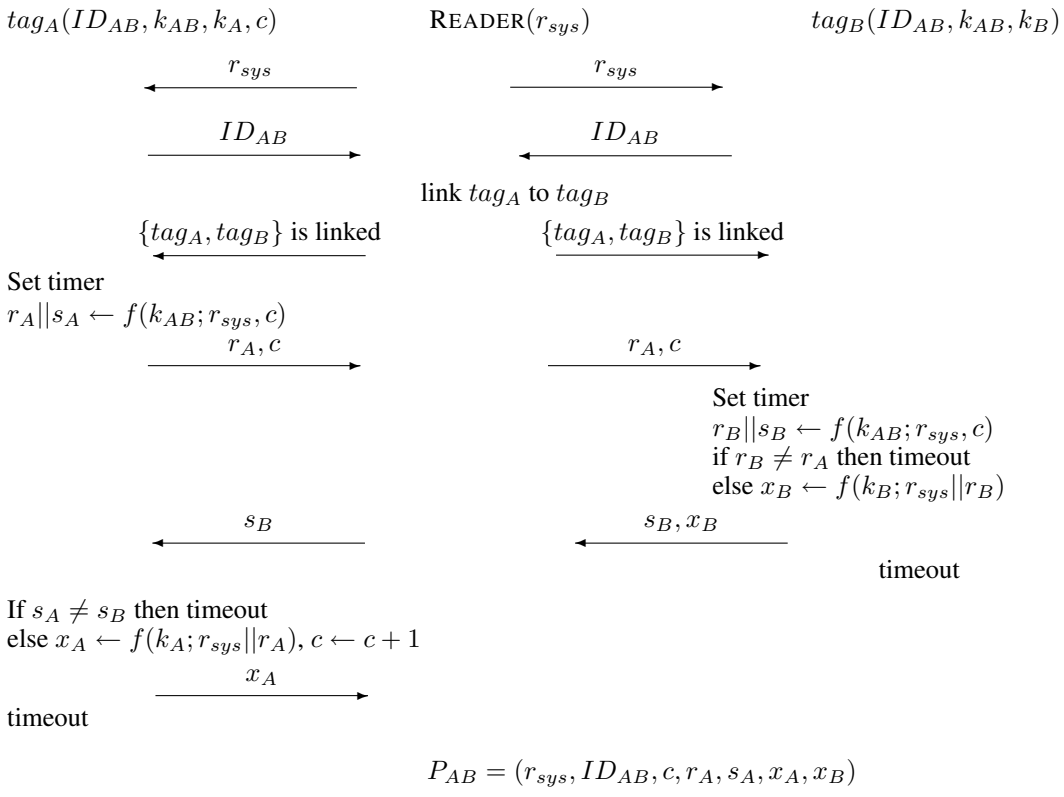
In the first protocol, the grouping-proof sent from the tags to the reader and from the reader to the Verifier includes the ID of the pair or group of tags (ID_{AB}). In the second protocol, no ID is passed to the reader: the grouping-proof uses values that depend on the group's ID and key and on the 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 anonymizing protocols require additional computational steps and correspondingly larger tag circuitry, simpler alternatives are preferred whenever anonymity is not a concern. Secondarily, the introduction of protocols of increasing complexity follows a natural pedagogical 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 c (in the non-anonymous protocol), or a random number r_A (in the other versions). This has the security benefit of curtailing reflection attacks. (Where a tag performs multiple roles in the same protocol instance.) To implement this feature, it is not necessary that the tags engage in any sort of real-time agreement protocol, it is sufficient to hard-code the behavior in the tags.

We only consider situations in which the Verifier is not online while the tags are scanned, as this is the most challenging case. As commonly done, we assume that readers and the Verifier communicate through authenticated channels. We also assume that each tag stores in non-volatile memory two secret keys (both shared with the Verifier). These are a *group key* k_{group}, used to prove membership in a group, and an *identification key* k_{tag} used to authenticate protocol flows. The Verifier stores these values (for each tag) in a database $D = \{ID_{tag}, k_{group}, k_{tag}\}$. Tags instances are denoted as tag_A or tag_B, and the key for instance tag_A is written in shorthand as k_A, instead of the more cumbersome k_{tag_A}.

Figure 4: A robust grouping-proof—for two tags



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

In our non-anonymous protocol the tags use a counter to update their state. The initiator tag stores the current value of the counter c—see Fig. 4. 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 ID. 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, or worse, of determinate outcome. In fact, if we removed the first two phases (the exchange of r_{sys} and ID_{AB}), the READER would be unable to match the tags, so that the group itself would be undefined. Phase three consists of three rounds of communication, and each is crucial to provide the data for the grouping-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 (honest) reader to detect certain protocol failures immediately, namely those that lead the initiator tag to timeout. The extension of the protocol for more than two tags is achieved as follows. In the first and second phases, the READER communicates with all tags concurrently. In the first round of the third phase, the READER communicates only with the initiator tag; it communicates with all other tags concurrently in the second round; and again with the initiator tag in the third round, providing it with concatenated answers from the second round.

In the protocol each tag tag_Z uses its group key k_{AB} to compute $f(k_{AB}; r_{sys}, c)$, where f is a pseudo-random function. This is parsed as $r_Z||s_Z$, and the parsed values are used to identify the group and to prove membership in the group. Tags use their secret key to confirm correctness of the grouping-proof. $P_{AB} = (r_{sys}, ID_{AB}, c, r_A, s_A, x_A, x_B)$ is the proof of simultaneous scanning. In our protocol, it is possible for a (honest) 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 a significant 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 [26].

4.2 A robust anonymous grouping-proof

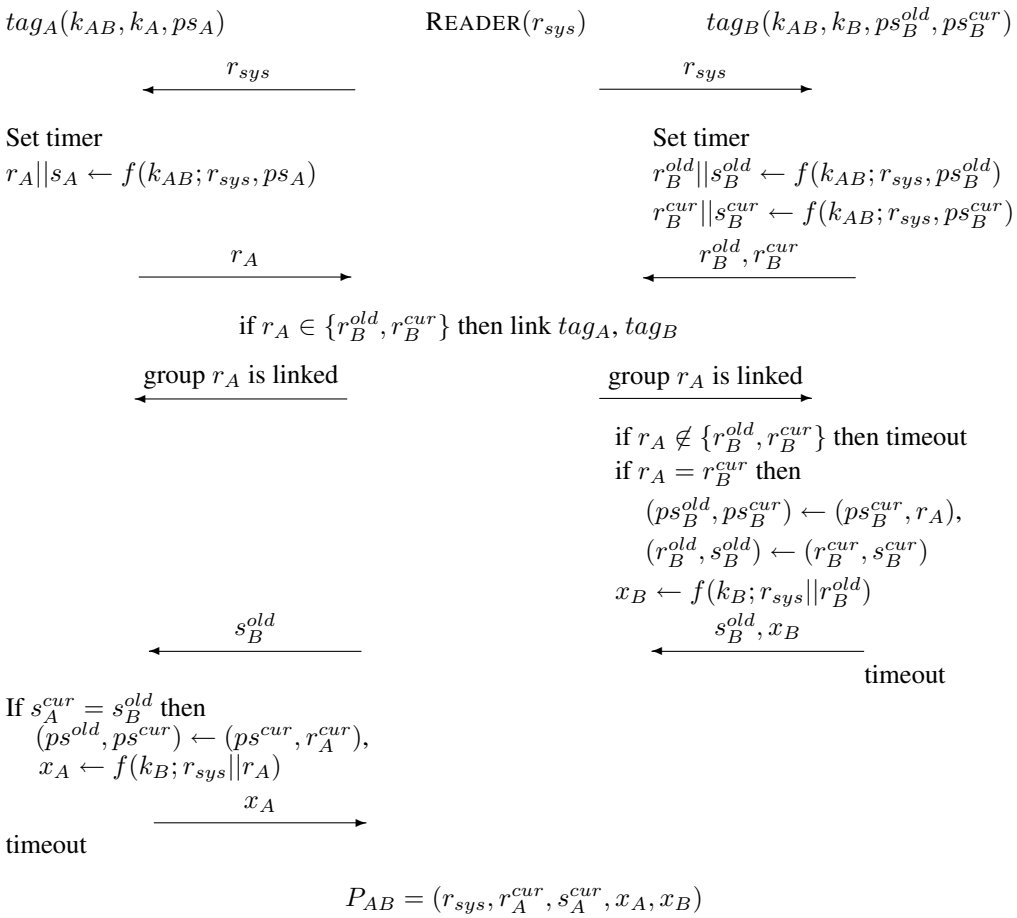
In our second protocol, in order to guarantee anonymity, group IDs are replaced by randomized pseudonyms ps stored in the non-volatile memory of tags. In order to synchronize the pseudo-random computations, one (or more in the group case) of the tags must maintain both a "current" and an "earlier" version of the pseudo-random function, to prevent against a de-synchronization failure or attack.

The initiator tag_A stores only the current value ps_A, while all other tags tag_X store two values ps_X^{old}, ps_X^{cur}. The Verifier keeps a database $D' = \{k_A, k_{group}, ps_A\}$ that links the secret keys of initiator tags to the current value of their group pseudonym. This database is updated at each successful execution of the protocol. The updated value is obtained by first evaluating $f(k_{group}; r_{sys}, ps_A)$ for each initiator tag_A, where r_{sys} is the next random challenge of the Verifier. This is parsed as $r_A||s_A$, with r_A the next value of the pseudonym of tag_A. The database D' is used also 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 response r_A and the authenticator x_A. The secret keys of the other tags of $group$ can be found in database D of the Verifier from the group key k_{group}. If no value in D' corresponds to the pseudonym used by the initiator then the Verifier will have to find the secret key of the initiator tag from its authenticator $x_A = f(k_A; r_{sys}, ps_A)$, by exhaustive search over all secret keys (of initiator tags).

The pseudonyms of the tags are initialized with random values: for the initiator tag: $ps_A \leftarrow r_A^1$, while for all other tags in its group: $(ps_X^{old}, ps_X^{cur}) \leftarrow (r_X^0, r_A^1)$. The proof is presented in Fig. 5, 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 r_A, provided there are n such tags.

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 at least one pseudonym ps^* in common. There will always be at least one tag_X in this set for which $ps_X^{cur} = ps^*$. The READER elects an initiator tag among those tags sharing the common

Figure 5: A robust anonymous grouping-proof—for two tags



pseudonym as the current value (deterministically, probabilistically, or in some ad hoc way: e.g., the first to respond). The READER then 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.

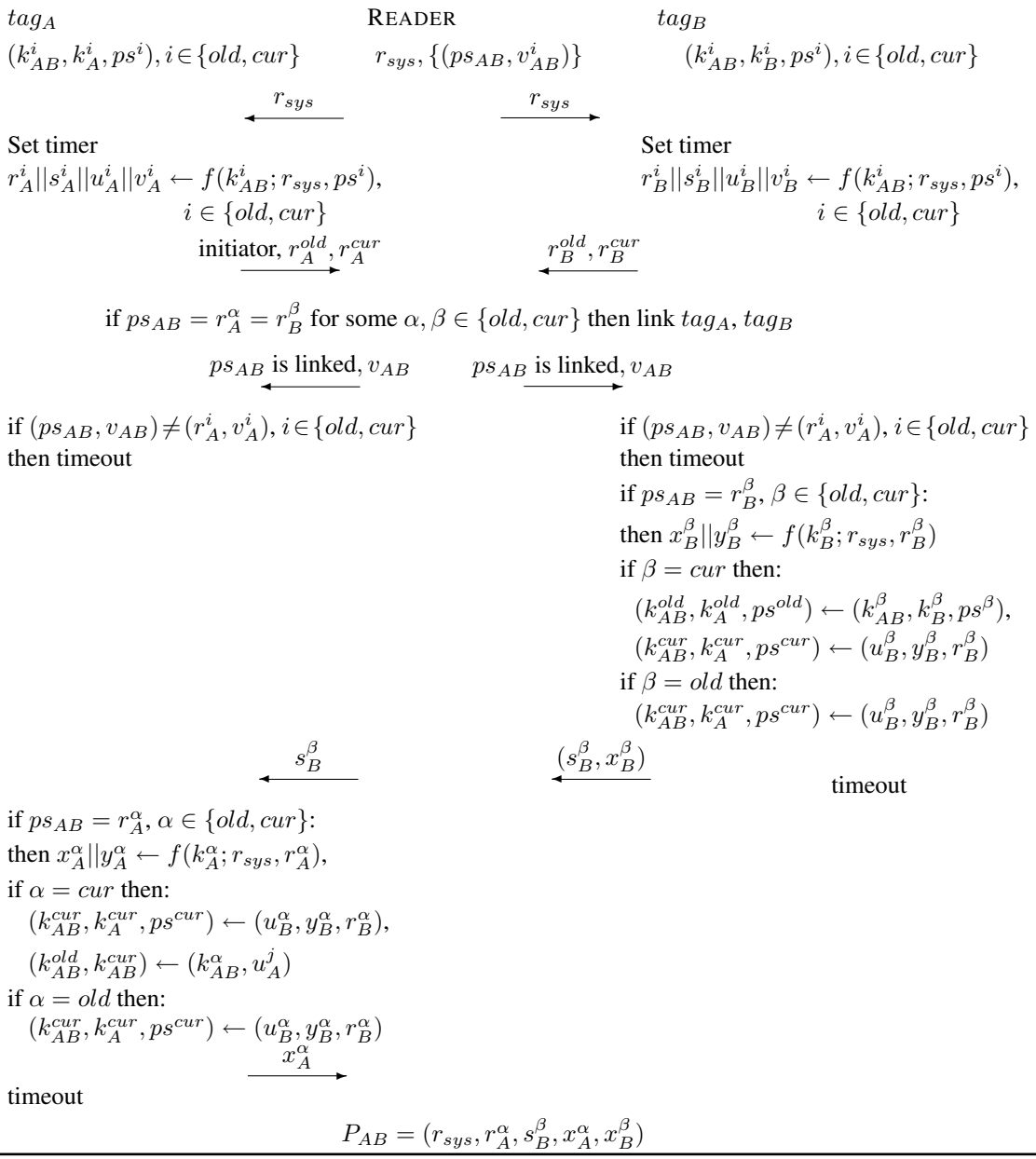
It is important to notice that the data transmitted from the tags to READER depend on r_{sys} and on some ps. Thus, at every round, the values that READER receives vary, even if a malicious READER attempts to reuse the same value of r_{sys} in multiple rounds. This provides unlinkability and anonymity. (If an unauthorized READER interrupts the round, preventing the pseudonym update, and then re-uses r_{sys}, it can link the two sessions. However, the power of this attack is limited because a single round with an honest reader at any point will restore unlinkability.)

As in the previous protocol, each step is essential. The main difference is that in the anonymous protocol, the tags exchange r_A, r_B^{old} and r_B^{cur}, rather than the group identifier. The functionality provided by this step, however, is analogous in the two protocols and enable the Verifier to identify the group.

Notice that this 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

Figure 6: An anonymous grouping-proof with forward-secrecy—for two tags



In our third protocol, for forward-secrecy, the secret keys and the group keys of the tags are updated after each protocol execution. All tags tag_Z (where $Z = 1, 2, \cdots, N$), including the initiator tags, store two pairs of keys: the secret key k^i_{group}, $i \in \{old, new\}$, the group key k^i_X, $i \in \{old, new\}$, and a pair of group pseudonyms ps^i, $i \in \{old, new\}$. The Verifier stores in a database the values $D = \{(k_{group}, k_Z, ps)\}$: this allows it to link the current values of the keys of each tag_Z to the corresponding group pseudonym. At the end of each r_{sys} challenge session, the entries in D of all tags in groups for which the Reader has returned

a valid proof P_{group} are updated: $(k_{group}, k_Z, ps) \leftarrow (y, u, r)$, using the parsings $f(k_{group}; r_{sys}, ps) = r||s||u||v$ and $f(k_z; r_{sys}, r) = x||y$. Here k_{group}, ps are the current values in D and r_{sys} is the current challenge (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 of 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 $\hat{D} = (r_{sys}, \{(ps_{group}, v_{group})\})$ whose values can be used to authenticate authorized readers to tags. The entries in this table are obtained by parsing $f(k_{group}; r_{sys}, ps_{group}) = r||s||u||v$ as above, and assigning: $(ps_{group}, v_{group}) \leftarrow (r, v)$. 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. 6 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 $k^{old}_{group}, k^{old}_A$ only after the Verifier has confirmed that it has updated its corresponding values. More specifically, if READER is not adversarial, then $ps_{AB} = r^{cur}_A = r^{cur}_B$, and the tags will update both current and old key and pseudonym values. If READER is adversarial and has not returned the proof P_{AB} then $ps_{AB} \neq r^{cur}_A, r^{cur}_B$, and the updates will not affect old values, which therefore remain the same as those stored in the database D. The state of the tags will only return to the stable state when an honest reader returns a valid grouping-proof to the Verifier. Note that due to the state synchronization requirements, the protocol in Fig. 6 can only be implemented in online batch mode, not true offline mode. In the full paper, we discuss the batch offline case.

Forward-secrecy is restricted 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.

It is worth highlighting that it is possible to augment the table $\hat{D}$ for batch interrogations. This would require that the values of (ps_{group}, v_{group}) in $\hat{D}$ be obtained by parsing evaluations that use a fixed initial set of values for the group keys, the secret keys and pseudonyms of the tags. Interrogated groups will assume the state $\alpha = \beta = old$, and thus keep the values of their old keys and pseudonyms until the next batch interrogation. This restricts forward-secrecy for batch periods: a compromised group can be traced during a batch session by a faulty reader.

5 Conclusion

The three protocols that we have hereby proposed 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. They 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.

The choice of any of the three protocols should be based on the cost of tags, anonymity and secrecy requirements. Whenever keeping the ID of the tags secret is not a concern, the first, simpler and non-anonymous protocol is recommended, while the second and third, which require more computational steps and additional storage, should be reserved for applications where anonymity and forward-security are required.

In the Appendix, we provide a sketch of the security proof in the Universally Composable framework.

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] 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.

[6] 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.

[7] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Symposium on Foundations of Computer Science (FOCS), 2001*, 2001.

[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] S. Engberg, M. Harning, and J.C. Damgärd. 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.

[11] EPC Global. EPC tag data standards, vs. 1.3. `http://www.epcglobalinc.org/standards/EPCglobal_Tag_Data_Standard_TDS_Version_1.3.pdf`.

[12] ISO/IEC. Standard # 18000 – RFID air interface standard. `http://www.hightechaid.com/standards/18000.htm`.

[13] 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.

[14] Ari Juels. Generalized "yoking-proofs" for a group of RFID tags. In *MOBIQUITOUS 2006*, 2006.

[15] P. Peris Lopes, J. Hernandez-Castro, J.M. Estevez-Tapiador, and A. Ribagorda. Solving the Simultaneous Scanning problem Anonymously: Clumping proofs for RFID Tags. Unpublished Manuscript, Carlos III University of Madrid, 2007.

[16] 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.

[17] Y. Oren and A. Shamir. Power analysis of RFID tags. Invited talk, RSA Conference, Cryptographer's Track (RSA-CT 2006). Available at urlhttp://www.wisdom.weizmann.ac.il/ yossio/rfid, 2006.

[18] T. Pering, R. Ballagas, and R. Want. Spontaneous marriage of mobile devices and interactive spaces. *Communications of the ACM*, 48(9):53–59, 2005.

[19] 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.

[20] J. Saito and K. Sakurai. Grouping proof for RFID tags. In *19th International Conference on Advanced Information Networking and Applications, AINA 2005.*, volume 2, pages 621–624, March 2005.

[21] S. Sharma, 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, 2003.

[22] 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.

[23] 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.

[24] 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.

[25] Istvan Vajda and Levente Buttyan. Lightweight authentication protocols for low-cost RFID tags. In *Proc. Workshop on Security in Ubiquitous Computing (UBICOMP 2003)*, 2003.

[26] 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.

[27] R. Want. An introduction to RFID technology. *IEEE Pervasive Computing*, 5(1):25–33, 2006.

A Proof of Security and Security analysis

In the following, we prove the 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 [7]. More precisely, consider an ideal functionality $\mathcal{F}$ that is securely realized by a protocol ρ. Let π be an arbitrary protocol that makes ideal calls to multiple instances of $\mathcal{F}$. Let π^ρ denote the composed protocol which consists of running the protocol π and substituting calls to each instance of $\mathcal{F}$ by calls to a distinct (fresh) instance of ρ. The universal composability theorem states that the protocol π^ρ has essentially the same effect of the protocol π, even if π^ρ 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. ρ 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 the READER (if the latter is corrupt) or generates them (if the READER is honest);
- answer challenges on behalf of the honest tags;
- decides which tags are grouped;
- collects the data for grouping-proofs from corrupt tags;
- implement 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 ρ evaluates the function $f(\cdot;\cdot,\cdot)$ on a triple $(k;r,ps)$, the functionality $\mathcal{F}$ first checks whether it has a record $(\mathsf{funtion\_value}, k; r, ps, t)$ 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 $(\mathsf{funtion\_value}, k; r, ps, t)$ 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, re-play, 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.