Distributed Group Authentication for RFID Supply Management

Mike Burmester, *Senior member, IEEE,* and Jorge Munilla

Abstract—We investigate an application of Radio Frequency Identification (RFID) referred to in the literature as *group scanning*, in which an RFID reader device interrogates several RFID tags to establish "simultaneous" presence of a group of tags. Our goal is to study the group scanning problem in strong adversarial settings and show how group scanning can be used in distributed applications for supply chain management. We present a security framework for group scanning and give a formal description of the attending security requirements. Our model is based on the Universal Composability framework and supports re-usability (through modularity of security guarantees). We propose two novel protocols that realize group scanning in this security model, based on off-the-shelf components such as low-cost (highly optimized) pseudorandom functions, and show how these can be integrated into RFID supply-chain management systems.

Index Terms—Distributed RFID systems, supply-chain management, grouping-proofs, authentication.

1 Introduction

The term "Internet of Things" (IOT) was coined in 1999 by Kevin Ashton, a co-founder of the Auto-ID Center [1] whose mission was to create a global RFID (Radio Frequency Identification) based product identification system for inventory and supply management. There are several advantages of RFID technology over barcodes: RFID is wireless, does not require direct line-of-sight, and RFID tags can be interrogated at greater distances, faster and concurrently [2]. This makes the IOT a wireless network of objects and sensors that collect and process information autonomously. RFID tags and sensors enable computers to observe/identify/understand for situational awareness without the limitations of a human in the loop.

RFID is presently a mature technology that is widely deployed for supply-chain management, retail operations, inventory management and automatic identification. A typical RFID deployment involves three main components: i) tags or transponders, which are electronic data storage devices attached to objects to be identified; ii) readers or interrogators, which manage tag population, read data from and write data to tags; and iii) a back-end server, the verifier, which is a trusted entity that exchanges tag information with the readers and processes data according to specific task applications.

- *M. Burmester is with the Department of Computer Science, Florida State University, Tallahassee, FL, 30302. E-mail: burmeste@cs.fsu.edu*
- *J. Munilla is with the Communication Engineering Department, Universidad de Málaga, Spain, 29070.*

This work has been partially supported by Ministerio de Ciencia e Innovacin (Spain) and the European FEDER Fund under project TIN2011-25452.

Most RFID tags are passive and do not have power of their own but get the energy needed to operate from an RFID reader. Thus, tags are inactive until activated by the electromagnetic field generated by a reader that is tuned to their frequency.

Although initial designs of RFID identification protocols focused on performance with little attention paid to resilience and security, this technology has found use in many applications that require the implementation of security mechanisms that: i) take into account features such as the vulnerability of the radio channel, the constrained power of the devices, the low-cost and limited functionality of tags and the request-response operation mode; and ii) make them resistant to privacy/confidentiality threats, malicious traceability and loss of data integrity. The recent ratification of the standard Gen2v2 highlights these security concerns [3].

When RFID technology is used for supply-chain management, concerns regarding the monitoring of tags and transfer of ownership or control of tags need to be addressed. If the transfer is permanent, or even temporal, ownership transfer protocols can be used, for which the rights of an owner are securely transferred to a new owner [4], [5]. However, there are cases when the owner does not want to cede control, even though this may be temporal. For example, a manufacturer may use the services provided by a carrier who, in turn, uses other carriers to transport their products. In such cases it is desirable that the owner and the carrier can periodically check the integrity of a consignment. This requirement is referred to as *group scanning*, or a *grouping-proof* in the literature, and involves multiple tags proving simultaneous presence in the range of an RFID reader [6], [7].

There are several practical scenarios where

grouping-proofs can 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.*, hardware components of a system or kits. A different scenario would be to support enforcement of safety regulations requiring that drugs be shipped, or dispensed, with information leaflets. However, as public key cryptography is beyond the capability of most RFID tags, such proofs can only be checked by a verifier that shares private information with every tag. As a result the carrier may not be able to check directly the integrity of a group.

Our main contributions in this paper are to:

a) Analyze a recently proposed grouping-proof and discuss the challenges of securing such proofs.
b) Present a framework for RFID grouping-proofs that addresses practical settings, in particular supply-chain management, and that allows for side-to-side comparisons with alternative proposals.
c) Present grouping-proofs of integrity that are generated by the tags of the group without sharing any private information with the reader. These can be verified by the owner but also, if needed, by a custodian who has no control over the tags.

The organization of this paper is as follows. In Section 2 we review the literature and analyze a recently proposed grouping-proof. In Section 3 we discuss RFID deployments for supply-chain management and present a high-level description of the security requirements and procedures for group scanning, in particular for compiling evidence during a group interrogation, and discuss the threat model. In Section 4 we propose two RFID protocols for group scanning: a non-anonymous grouping-proof and a version that adds support for anonymity. We then show in Section 5 how these can be integrated into RFID supply-chain management. We summarize our main results in Section 6. In the Appendix we formally define the functionality of our two protocols and show that these are realized in the Universal Composability framework.

2 Background

Ari Juels introduced in 2004 the security context of a new RFID application—which he called a yoking-proof [8], that generates evidence of simultaneous presence of two tags in the range of an RFID reader. This first protocol was later found to be insecure [9], [10] but, the simultaneous scanning application triggered a considerable interest in the research community. Yoking-proofs have been extended to *grouping-proofs* in which multiple tags prove simultaneous presence in the range of an RFID reader—see e.g. [11].

Burmester et at. presented in [12] a protocol which achieved anonymity by using randomized pseudonyms for the group identifer, and forward-security by updating the secret keys and the group keys after each season. This protocol is essentially a proof-of-concept, and not appropriate for lightweight applications. Huang and Ku [13] presented a grouping-proof for passive low-cost tags that uses a pseudorandom number generator to authenticate flows and a cyclic redundancy code to randomize strings. The protocol has several weaknesses, some of which were addressed by Chien et al. [14] who, in turn, proposed a new grouping-proof. Peris-Lopez et al. [15] found other security flaws in these protocols and proposed guidelines for securing them as well as a yoking-proof protocol (for two tags).

More recently, Liu et al. proposed in this journal [6] a grouping-proof for multiple readers and tags. In this proof the reader is a contributing party that shares a private key with the tags of the group. There are several security issues with this proof that we discuss below, and will serve as guideline for designing secure grouping-proofs.

Liu et al.'s Grouping-Proof

This proof is vulnerable to de-synchronization and privacy leaks. De-synchronization occurs when the adversary succeeds in disrupting the protocol by partitioning the group into two parts with one part updating its state while the other does not.

The Liu et al. protocol uses an invertible pseudorandom number generator $PRNG$. Authentication data is obfuscated by using bitwise operations: XOR ($\oplus$), OR ($\vee$) and modular addition (+), operations that have been shown to be insecure [16]. In this particular case private key information leaks.

Since the description of the Liu et al. protocol is quite complex, here we describe only the case when there are two tags T_A, T_B and a single reader R, which is sufficient for our arguments. The tags share a secret key s with R.

Step 1. R generates a pseudorandom number r_0, and sends it together with a pseudorandom flag F_R to T_A and T_B.

$$R \rightarrow T_A, T_B : \quad r_0, F_R$$

Step 2. T_A, T_B check F_R. If it is not correct, the protocol terminates. Otherwise, the tags send to R their flags F_{T_A}, F_{T_B}.

$$T_A, T_B \rightarrow R : \quad F_{T_A}, F_{T_B}$$

Step 3. R checks the flags and if correct links the tags.

$$R \rightarrow T_A, T_B : \quad T_A, T_B \text{ are linked}$$

Step 4. T_A updates its flag: $F'_{T_A} \leftarrow F_{T_A}$, computes $M_A = (PID_{T_A} \oplus F'_{T_A}) \vee r_0$, $N_A = PRNG(F_R \vee PID_{T_A})$ and sends these to R.

$$T_A \rightarrow R : \quad M_A || N_A$$

Step 5. Upon receiving this message, R updates the flag: $F'_{T_B} \leftarrow F_{T_B}$, selects a pseudorandom number r_1, and computes $M_R = (PID_{T_B} \vee F_R) \oplus (s + r_1)$, $N_R = (gid \oplus F'_{T_B}) \vee r_1$, where gid is a group identifier.

$$R \rightarrow T_B : \quad M_R||N_R||N_A$$

The protocol continues: T_B updates its flag in Step 6 and R computes F'_{T_A} in Step 7.

In this protocol, parties update their flags independently. This leads to a synchronization problem. For example, if the protocol is interrupted after Step 4, T_A will have updated F_{T_A} but R will not, so R will be unable to recognize T_A in the next interrogation in Step 2. The interruption may have a natural origin (communication problems) or may be caused intentionally by an adversary: physically, or by first impersonating T_A in Step 1 to get r_0, F_R and then impersonating R to get T_A.

Assume now that to overcome this problem the flags are not updated independently, but only when each party has made certain that the other has updated. In this case, because of the bitwise operations, secret key information leaks (as in [5]). In fact, if flags remain static, an adversary can impersonate the tags to the reader, by replaying these flags, to get different values of M_R $(M'_R, M''_R, \dots)$ and N_R $(N'_R, N''_R, \dots)$. The computed values can be written:

$$M_R = K_A \oplus (s + r_1) \text{ and } N_R = K_B \vee r_1,$$

where K_A, K_B are constant and r_1 varies with every execution $(r_1, r'_1, r''_1, \dots)$. The value of K_B can be easily retrieved by observing different N_R. Once K_B is known, many bits of different r_1's also get known (those for which the corresponding bits of K_B are zero). For the bits of r_1 that are known, the corresponding i-th least significant bit s^i, can be retrieved as follows:

1) Get two pairs: (r_1, M_R) and (r'_1, M'_R).
2) Compute: $R = r_1 \oplus r'_1$ and $C = M_R \oplus M'_R \oplus R$.
3) For $i = 1 : length(s)$, s^i is computed:
   If $C^i = 1$ and $R^i = 0$ then $s^i = r_1^i \oplus C^{i+1}$;
   Elseif $C^i = 0$ and $R^i = 1$ then $s^i = c^i \oplus C^{i+1}$ provided that c^i can be computed, with $c^i = (c^{i-1} \wedge (r_1^{i-1} \oplus s^{i-1})) \vee (r_1^{i-1} \wedge s^{i-1})$ for $i > 1$ and $c^1 = 0$.
   Else Bit i cannot be retrieved for these pairs.
4) Return to Step 1 until every bit of s (for those positions of r_1 that are known) is disclosed.

Finally, as mentioned earlier, the proof is not generated by the tags themselves, but requires the reader to check and compute authentication data for the tags.

3 Framework Assumptions

A typical deployment of an RFID supply-chain involves three types of legitimate entities—see Figure 1:

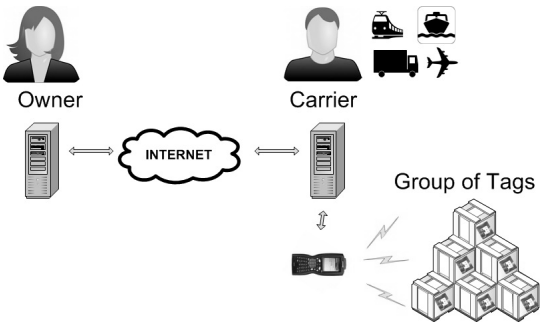


Fig. 1. An RFID Supply-Chain Scenario

a) A *group of tags* (GoT).
b) The *owner* of GoT, who keeps the digital rights of the tags; in particular she knows the private information stored by the tags. However, the GoT may not be in the physical range of the owner.
c) The *carrier*, whose services are contracted by the owner. He has physical possession of the GoT and can access the GoT through his reader, but does not have control over it, other than, when allowed, verifying its integrity.

We consider two modes for supply-chain management: Mode A, where the owner monitors the integrity of a GoT via a carrier, and Mode B where the carrier directly monitors the GoT.

3.1 RFID Deployments

We assume the following regarding the environment that characterizes RFID group scanning applications.

3.1.1 RFID tag capabilities

Passive UHF tags are the most common for supply-chain applications. They have no power of their own, operate in the far field, and use backscatter communication [17]. Such tags work at greater distances (than inductive tags) but the delivered power is low, and therefore lightweight cryptographic tools should be utilized [18]. However, we can assume that tags are able to perform basic symmetric-key cryptographic operations such as selecting pseudorandom numbers and evaluating a pseudorandom function $f : \{0,1\}^* \rightarrow \{0,1\}^n$ (*e.g.*, a pseudorandom hash function that is one-way and collision resistant [19]).

Public key cryptography, tamper-resistant shielding and on-board clocks are beyond the capabilities of most tags. 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 [8].

3.1.2 RFID Reader and verifier/server capabilities

By contrast readers and verifiers/servers are able to perform complex cryptographic operations. Although in practice they may be implemented on the same

device, with two communication ports: one for the tags in range, the other for the rest of the high-level entities.

In our model these entities are independent. In particular, readers will just manage the communication between tags (Section 3.2, Assumption 2) and interface between the verifier and the GoT.

3.1.3 RFID communication channels

Direct communication between passive RFID tags is not possible. Such tags can only communicate with readers that are in wireless range (they backscatter the reader's electromagnetic signal). However readers can establish logical wireless channels that link the tags of a group.

To establish a channel, the tags must provide the reader with identifying origin information to establish an association with the reader. After the tags get identified (not necessarily authenticated), they get linked with a wireless communication channel via the reader (in practice: origin and destination tag information is appended to all exchanged messages). We shall not discuss physical/link layer details such as the coupling design, the power-up and collision arbitration processes. For details on these issues the reader is referred to [3], [20].

RFID wireless channels are particularly vulnerable because tags are restricted to lightweight cryptographic protection. By contrast, the communication channel between high level entities (*i.e.* readers and verifiers) is secure since fully-fledged cryptographic techniques can be used. However, these channels may enjoy continuous connectivity (*e.g.*, if they are implemented on the same device), or may not (*e.g.*, if they link the carrier's reader and the owner's verifier).

3.2 Grouping-proof assumptions

A grouping-proof encompasses evidence that corroborates the presence of a GoT during a reader interrogation. The evidence consists of records of temporal events from which a verifier can infer the presence of the tags of a group.

Let G a GoT and R an authorized reader. There are two basic requirement regarding the interrogation evidence and how it is compiled to get a proof of simultaneous presence.

- (*Completeness*) If all tags of G are in the range of R then a grouping-proof is generated.
- (*Soundness*) If when R interrogates the tags of G a grouping-proof is generated then all tags of G were scanned by R.

The evidence for a grouping-proof is generated using symmetric-key operations since public-key cryptography goes beyond the capabilities of lightweight tags (Section 3.1.1). Consequently grouping-proofs are not "proofs" in the sense that they are not transferable and can only be validated by those who share the private keys used to generate them.

During an interrogation the verifier can be online or offline, and different solutions for the grouping-proof problem are required in each case. We further distinguish between fully-interactive online and batch online [7]. In batch mode the interactions of the verifier are restricted to: $i)$ broadcasting a challenge that is valid for a (short) time span and, $ii)$ collecting responses from the tags (via intermediate readers) to check the simultaneous presence of the tags of a group. The verifier in batch mode never unicasts messages to specific tags.

In contrast in the fully-interactive mode the verifier can send/receive messages to/from specific tags throughout the protocol execution. It is straightforward to check the integrity of a GoT in this mode. Indeed it is sufficient for individual tags to authenticate themselves to the verifier, who will then decide on the validity of the grouping-proof by using auxiliary data, *e.g.,* an identifier of the GoT. Therefore, in this paper, we focus on offline solutions.

Assumption 1. The verifier is either offline or batch online.

The interrogating reader may, or may not share private keys with the GoT. In the first case we further distinguish two cases: the reader may or may not be trusted. If the reader is trusted then it can share the private information of the tags, thus becoming a de facto online verifier. This violates Assumption 1. If the reader is not trusted, then it should not share any private information with GoT. Consequently we focus on grouping-proofs that are generated independently by the GoT.

Assumption 2. The reader is a communication enabler that links the GoT with reliable channels, but is not involved in any tag computation and does not share any private keys with the GoT.

We view a grouping-proof as an independent process in which the GoT generates corroborative evidence given a certain input. A GoT consists, in turn, of tags with similar characteristics, none of which requires special features or is assigned an unbalanced computation load. In particular, no tag of GoT will assume the role of a "centralized" verifier; *i.e.*, the proof should be distributed.

Assumption 3. The tags of a group have similar hardware capabilities and the computation load per tag for generating a grouping-proof is balanced.

In general some of the tags of a group may get compromised. It is easy to see that if this happens then it is not possible to generate evidence that will support simultaneous presence in any meaningful way. Indeed if T is a tag that is controlled by the adversary, then T can prevent a grouping-proof from being generated by not participating actively, and

conversely force a grouping-proof to be generated when GoT is not complete.

Assumption 4. The tags of a GoT are not compromised.

The motto "all for one and one for one" extends to private shared keys and the definition of a grouping-proof. Indeed since a group of tags is compromised if any of its tags gets corrupted, from the perspective of generating a grouping-proof there is no reason for tags to have different keys. However these keys are restricted to a specific GoT.

Assumption 5. The tags of a GoT share the same private information.

Assumption 6. Grouping-proofs apply to specific GoT: for a subgroup or extensions a different, independent proof should be sought.

RFID readers broadcast messages, but as pointed out in Section 3.1.3, if the tags get identified then the reader can link them via unicast/multicast channels. To enhance privacy we may wish to consider grouping-proofs for which the tags are *not identified* by the reader, and only broadcast communication is used. In this case tags broadcast messages via the reader that are checked by *all tags* in range (possible destination). There are two cases to consider: i) only tags listed as destination tags respond, and ii) all tags respond (possibly with not-valid messages). The former reveals membership in a group, and does not provide any advantage as we shall see in Section 4.3.2. The latter, by contrast, has the potential to offer *complete anonymity*, but leads to an ever-increasing number of broadcast messages which hinders its implementation.

Assumption 7. The tags of a group get identified by the reader and linked via the reader to allow for unicast/multicast communication.

RFID communication is a sequential process and therefore the concept of interrogation simultaneity can only be captured by an "exposure-time" window. That is, events are considered as happening simultaneously only if they take place within this window.

Assumption 8. Grouping-proofs use session numbers (or timestamps) to define interrogation time windows for simultaneity.

Finally, we distinguish two ways in which the verifier can request evidence for the presence of a GoT. The verifier can request evidence for all GoT in range, or evidence for an specific group. In our protocols we shall use the first, which is more general and provides stronger anonymity. However, both cases can be accommodated in the general architecture described in Section 5.

Assumption 9. The verifier requests evidence for any GoT present in the range of the reader.

3.3 Threat Model for RFID Group Authentication

We discuss the components of grouping-proofs and their vulnerabilities.

3.3.1 The RFID-air interface

Wireless channels are particularly vulnerable to adversarial threats. We shall assume the Dolev-Yao threat model [21] in which the adversary controls the communication channels, and may eavesdrop, block, modify and or inject messages in any communication between tags and readers.

However, in practice, reader-tag (forward) channels are easier to intercept than tag-reader (backward) channels, since the signal in the latter case is much weaker.

3.3.2 RFID Readers and tags

We shall assume that (authorized) readers provide reliable communication channels for all tags in their range, when requested, and do not share any private information with a GoT (Section 3.2, Assumption 4). Authorized readers use a pseudorandom number obtained from the verifier to challenge a GoT. This serves as an identifier for authorized sessions, and defines the corresponding validity period.

Tags are computationally constrained, restricted to the air interface and have no clocks other than timers. The tags of a GoT are bounded by Assumptions 3, 4, 5, 7 and 8.

3.3.3 Attacks on RFID Systems

Several types of attacks against RFID systems have been described in the literature. Some are well known in other platforms. In particular, the adversary may attempt to perform impersonation, DoS, interleaving and reflection attacks and other passive or active attacks. Additionally, the unique aspects of RFID applications highlight other vulnerabilities such as unauthorized tracking, a privacy concern in which the adversary tries to trace and/or recognize tags or GoTs.

There are also attacks on RFID systems that are usually excluded from the security model used, such as *online man-in-the-middle relay attacks* [22], *side channel* or *power analysis* [23] attacks (*cf.* Section 3.1.1). In particular, if no distance-bounding mechanism [24] is used, our protocols in Section 4 will be subject to active attacks that involve relaying flows between tags faster than the time window of tag timers. These attacks affect all RFID protocols [25], including grouping-proof protocols [26], and can only be addressed by making certain that precise timing mechanisms are used.

3.3.4 Supply-Chain Management

The owner of a GoT (a back-end server) is assumed to be a trusted entity that shares private information with the tags such as cryptographic keys. As for the

carrier we distinguish two cases, depending on the supply-chain management mode. For Mode A, when the owner monitors the integrity of a GoT in transit via the carrier, the carrier is not trusted. For Mode B, the carrier monitors the integrity of the GoTs and should be trusted.

3.3.5 Universal Composability, Modularity

A protocol that is secure in isolation may become vulnerable under concurrent execution (with other protocols or instances of itself—see *e.g.*, [27]). To guarantee security against such attacks it is necessary to model security in a concurrency-aware model. In the Appendix we shall use the Universal Composability framework to capture security, 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.

4 Group Authentication

We describe two grouping-proof protocols that can be integrated into a supply-chain management system. The first (Section 4.2) does not provide anonymity. For this protocol the messages the tags send to the RFID reader include a group identifier ID_{gp}. The second (Section 4.3) uses pseudonyms *PS* that depend on the group identifier, but the dependency is known only to the tags. The pseudonyms are updated with every interaction, providing "session unlinkability", a form of anonymity that will be defined later.

Our grouping-proofs are generated distributively, without the verifier being involved, and can be seen as an independent process that compiles temporal corroborative evidence given a certain input.

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 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 facilitates their understanding.

The parties involved in our proofs are the GoT and the reader. The reader manages the communication between the tags and interfaces with the verifier. The reader does not share any information with the GoT, other than an input it receives from the verifier (Section 3.2, Assumption 2).

In our protocols a specific tag of the group, the *authenticator*, generates the proof after checking that all the tags in its group are accounted for. We assume that the tags of a group store in non-volatile memory a shared secret *group key* K used to prove membership in the group, and that the authenticator tag has also an additional key k shared with the verifier (discussed in Section 5), and used to generate a confirmation. Tags instances are denoted as tag_i, $i = 1, 2, \ldots$.

4.1 Compiling Evidence: the Domino Effect

Our proofs are based on temporal corroborative evidence provided by authenticators generated by the tags of a group. It is sometimes contended (*e.g.*, [6], [11], [15]) that to guarantee simultaneous presence of a group of tags, the values that each tag computes should be *explicitly* linked to the values the other tags in the group compute. Our protocols use a different approach in which the values that the tags compute (the authenticators) are *implicitly* linked via a causality relation. This approach makes the verification much simpler and does not require that tags of a group share different private information (Section 3.2, Assumption 5). We shall now discuss this approach and justify it from a theoretical point of view.

The evidence that corroborates a grouping-proof consists of records of temporal events. There are several ways to compile such records. Clearly one may discard records that can be inferred from others. There are several ways to further reduce the number of such records needed for a grouping-proof. Here we describe one such approach. Suppose that a linear causality relation on the event space that defines a grouping-proof can be established:

$$event(n) \Rightarrow event(n-1) \Rightarrow \cdots \Rightarrow event(1).$$

Then only $event(1)$ is needed to corroborate $event(i)$, $i = n, \ldots, 2$. This "domino" chain can be linear (as above), or more generally tree-based: it is characterized by events that cause other events, leading to a root event.

The following illustration captures our approach for compiling temporal evidence. Suppose that the verifier has access to records of the state of an interrogated GoT as captured by an exposure time window that only shows a root event. Then, if we assume an appropriate causality structure for the events, the verifier can infer the simultaneous presence of the tags of GoT during an interrogation by just observing the state of the root.

4.2 A Non-Anonymous Grouping-Proof

Our first protocol is presented in Figure 2, with the tags arranged in a logical ring: $(tag_1, tag_2, \ldots, tag_n)$, with tag_1 the authenticator tag. There are three phases. In the first, the reader challenges the tags in its range with a pseudorandom number r_{sys}, and the tags respond with their group identifier ID_{gp} except for tag_1 which also includes a random session number sn. In the second, the tags of ID_{gp} get linked by channels through the reader (Assumption 7, Section 3.2).

In the third phase tag_1 computes an authenticator aut_2 for tag_2, and the other tags generate authenticators in reverse order starting from tag_n, that sends to tag_{n-1} the authenticator aut_n. Each tag_i, where $i = n-1, ..., 2$, computes *two* authenticators: aut_{i+1} and aut_i. The first is used to authenticate tag_{i+1} in the

Fig. 2. A non-anonymous grouping-proof

Parties: READER: $[r_{sys}]$ AUTHENTICATOR TAG (tag_1): $[1, ID_{gp}, k, K]$ tag_i : $[i, ID_{gp}, K]$, $i = 2, \ldots, n$

Phase 1 a) READER $\rightarrow *$: r_{sys} (a random number is broadcast)

b) tag_1 : Generate a random number sn, set timer

$tag_1 \rightarrow$ READER : 1, sn, ID_{gp}

c) For each $1 < i \leq n$

$tag_i \rightarrow$ READER : i, ID_{gp}

Phase 2 The READER links the tags of ID_{gp} : for each $1 \leq i \leq n$.

READER $\rightarrow tag_i$: sn, ID_{gp} is linked,

Phase 3 tag_1 : Compute $aut_2 = f(K; ID_{gp}||r_{sys}||sn||2)$

a) tag_n : Compute $aut_n = f(K; ID_{gp}||r_{sys}||sn||n)$,

$tag_n \rightarrow$ READER $\rightarrow tag_{n-1} : i, aut_n$; timeout

b) For each $1 < i \leq n-1$

tag_i : Set timer; compute $aut_j = f(K; ID_{gp}||r_{sys}||sn||j)$, $j = i, i+1$

If $(i+1, X_{i+1})$ is received with $X_{i+1} = aut_{i+1}$ then

$tag_i \rightarrow$ READER $\rightarrow tag_{i-1} : i, aut_i$; timeout

else timeout

c) tag_1 : If $(2, X_2)$ is received with $X_2 = aut_2$ then compute $mac = f(k; ID_{gp}||r_{sys})$

$tag_1 \rightarrow$ READER : mac ; timeout

else timeout

READER : If an authenticator is received from the tag_1 then compile:

$$MAC = (ID_{gp}, r_{sys}, mac)$$

ring while the second is sent to tag_{i-1}. The authenticators are obtained by evaluating $f(K; ID_{gp}||r_{sys}||sn||j)$, $j = i, i+1$, where f is a pseudorandom function (Section 3.1.1). Each tag checks that the authenticator it received is correct before sending its authenticator. Eventually tag_1 gets aut_2. If this is correct then it sends to the reader a message authentication code $mac = f(k; ID_{gp}||r_{sys})$. Finally the reader compiles the grouping-proof: $MAC = (ID_{gp}, r_{sys}, mac)$. To validate the proof, the verifier who keeps in a database D pairs of values (k, ID_{gp}) first retrieves k from D and then checks MAC.

Each phase of the protocol can be executed concurrently with all tags in the group (this includes computing authenticators: *e.g.*, tag_2 can evaluate aut_2, aut_3 immediately after being linked), except for Phase 3. The phases cannot be consolidated without loss of some security feature, or worse, of determinate outcome. Indeed, the first phase incorporates randomness provided by the verifier's challenge r_{sys} and the randomness provided by the session number sn, which prevent replay attacks. The random challenge defines the scanning period for the verifier, and the session number the interrogation period of the GoT. In particular, the verifier cannot (without further assumptions) determine simultaneity of a group scan to a finer time interval than the scanning period. Phase 2 is used to define and link the group. Phase 3 consists of three rounds of communication, and each one is crucial to provide the data for the grouping-proof. If we were to suppress the exchange of aut_i, aut_{i+1}, or did not implement timeout, then replay attacks would be successful.

In this protocol the verifier is not authenticated which, in this particular case and from a theoretical point of view (provided that f is secure), is not a concern because the integrity of the GoT is captured by a message authentication code for the verifier, who provides the challenge.

We assume that the identifier ID_{gp}, the challenge r_{sys}, the keys K, k, the session number sn, and the authenticators $aut_1, \ldots, aut_n, mac$, all have the same (bit) length κ, which is the *security parameter* of the protocol.

This protocol can be implemented very efficiently: it is distributed, each tag needs only to compute two values (tag_n only one), and the verifier just needs to perform one check. For efficient hardware implementations of pseudorandom functions the reader is referred to [18], [28].

Due to its sequential nature, the time taken to identify all the tags of the group is linear in n, which could lead to problems when n is large. In the next subsection, we capture concurrency by modifying Phase 3.

4.2.1 A Concurrent Grouping-Proof

Arrange the tags of the group in a logical binary tree with $m = \lceil \log_2 n \rceil$ layers and root tag_1, see Figure 3. The only difference is in Phase 3 of the protocol in which we get group authentication by having all

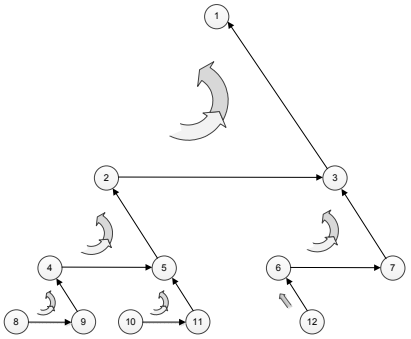


Fig. 3. A binary tree arrangement for 12 tags and authentication flows for a concurrent execution

triangles of the tree at layer j, $j = m-1, \ldots, 1$:

$$\langle \mathsf{leftchild}(tag_a), \mathsf{rightchild}(tag_a), tag_a \rangle$$

authenticated concurrently. Left-children of the last layer start concurrently (as tag_n did in the sequential case). Each tag sets a timer and waits until it receives an authenticator from its right-child and/or left-sibling, depending on its position in the tree. When the authenticator(s) is (are) received, if it is (they are) correct and the timer has not expired, the tag sends its own authenticator to either its right-sibling or its parent. Thus, when all the tags in a layer get authenticated the previous layer can get authenticated. Until only one triangle rooted at tag_1 is left, or at timeout. If any authentication fails (one of the tags is not authenticated) the process cannot be completed. For the concurrent protocol, the tags need compute at most three values, with the tags at the last layer computing either one or two values. The verifier still performs only one check.

4.2.2 Analysis

This protocol is secure in the Universal Composability (UC) framework (details are given in Appendix A). Here we discuss informally the most common attacks.

Replay attacks. The first pass of the protocol, with r_{sys} and sn, prevents replay attacks. The verifier will not re-use the same challenge r_{sys} and the authenticator will not re-use the same session number sn.

Impersonation attacks on tags are prevented by using private keys. Impersonation attacks on an authorized reader will not yield a valid proof: only authorized readers have access to the verifier's challenge r_{sys}.

DoS attacks. It is not possible to de-synchronize a GoT since the state of a tag is defined by ID_{gp} and its private key K, that are immutable.

4.3 An Anonymous Grouping-Proof

For our second grouping-proof the identifier ID_{gp} is replaced by a randomized group pseudonym PS, see Figure 4. Again the tags are arranged in a ring with tag_1 the authenticator. Tags store a private counter cnt_i, and initially all counters have the same value. We use a Blum-Micali encryption [29] (that simulates a one-time-pad) to obfuscate the value of counters.

In Phase 1, tag_1 updates $cnt_1 \leftarrow cnt_1 + 1$, computes the pseudonym $PS = f(K; ID_{gp}||cnt_1)$ and the session number $SN = f(K; PS) \oplus cnt_1$, sets the timer and sends the pair (SN, PS) to the reader, which in turn, broadcasts this pair. Then, each tag_i, $1 < i \leq n$, computes $cnt = SN \oplus f(K; PS)$ and checks that $cnt > cnt_i$. If so, tag_i verifies the pseudonym $PS = f(K; ID_{gp}||cnt)$; if it is correct, it updates $cnt_i \leftarrow cnt'_i$ and sends the pair (i, PS) to the reader. The reader links those tags whose responses include this pseudonym (Phase 2).

Phase 3, with $sn = SN$, is identical to the corresponding phase of the non-anonymous grouping-proof except for the compilation, in which ID_{gp} is removed. That is, $MAC = (r_{sys}, mac)$, with $mac = f(k; ID_{gp}||r_{sys})$. The verifier keeps in a database D pairs of values (k, ID_{gp}), and therefore is able to match the group by exhaustive search over all pairs using mac and r_{sys}.

4.3.1 A Protocol Amendment

There are two privacy concerns with this proof. The first involves the number of participating tags in an interrogation. This cannot be addressed without violating Assumption 7 of Section 3.2. We shall discuss this in Section 4.3.3 where we will prove that full anonymity cannot be achieved in our grouping-proof framework. The second involves the order i of a specific tag_i in the group. This concern can addressed if it is an issue. Below we sketch a modified protocol that randomizes the order of tags, including the authenticator.

Phase 1. In Step b (Figure 4), tag_1 selects a random number $t \in [1 : n]$, a nonce r_1, computes $PS = f(K; ID_{gp}||(t||cnt_1))$, $SN = f(K; PS) \oplus (t||cnt_1)$, and then send to READER: (r_1, SN, PS) (t is a short string of length $\ll \kappa$). In Step d, each tag_i computes $t||cnt = SN \oplus f(K; PS)$, checks that $cnt > cnt_i$ and that PS is correct, and if so, selects a nonce r_i, sets $cnt_i \leftarrow cnt$, and sends to READER: (r_i, PS).

Phase 2. The READER assigns to each tag_i a unique number $j_i \in [1 : n]$, with $j_1 = 1$, and sends to each tag_i: (r_i, j_i, PS) is linked.

Phase 3. Each tag_i assumes position j_i in the group, and includes this together with t in its authenticator. After sending its authenticator tag_t assumes the role of authenticator in subsequent interrogations. Upon sending mac, tag_1 relinquishes its role as authenticator.

If the interrogation is not completed then we may get a collision with two (or more) candidates (tag_1, tag_t) for authenticator tag. There are two possibilities: A) tag_1, tag_t have different counter values (and pseudonyms). The reader proceeds as if they were

Fig. 4. An anonymous grouping-proof

Parties: READER: $[r_{sys}]$ AUTHENTICATOR TAG (tag_1): $[ID_{gp}, k, K, cnt_1]$ tag_i : $[i, ID_{gp}, K, cnt_i]$, $i = 2, \ldots, n$

Phase 1 a) READER $\rightarrow *$: r_{sys} (a random number is broadcast)

b) tag_1 : Set $cnt_1 \leftarrow cnt_1 + 1$; compute $PS = f(K; ID_{gp}||cnt_1)$ and $SN = f(K; PS) \oplus cnt_1$; set timer
$tag_1 \rightarrow$ READER : SN, PS

c) READER $\rightarrow *$: SN, PS

d) For each $1 < i \leq n$
tag_i : Set timer; compute $cnt = SN \oplus f(K; PS)$
If $cnt > cnt_i$
If $PS = f(K; ID_{gp}||cnt)$ then set $cnt_i \leftarrow cnt$ and
$tag_i \rightarrow$ READER : i, PS
else timeout
else timeout

Phase 2 READER links the tags with pseudonym PS: for each $1 \leq i \leq n$.
READER $\rightarrow tag_i$: PS is linked,

Phase 3 It is identical to that of the non-anonymous case with $sn = SN$ (Figure 2), except for the compilation:
READER : If an authenticator is received from the tag_1 then compile:
$$MAC = (r_{sys}, mac)$$

two different groups, but only the pseudonym PS that corresponds to the highest value will be accepted by all the tags of the group. tag_1 or tag_t will take part in a grouping-proof that uses a PS with a cnt value higher than its own, and relinquish its role as authenticator tag after sending its authenticator. $B)$ tag_1, tag_t share the same counter value. The same pseudonym PS is used and it is up to the READER to appoint an authenticator tag (by selecting a corresponding nonce) and request this tag to send a new PS (computed with an updated cnt). We now proceed as in the first case.

4.3.2 Session Unlinkability

We cannot thwart the physical tracing carried out by an adversary who stays in contact with the tags. While the adversary is physically tracing a GoT, the adversary can determine which executions of the protocol are linked to this GoT. The concept of untraceability is then related to the capability of an adversary to link interrogations once this physical contact is temporarily interrupted.

For groups of a specific order n, our protocol provides *session unlinkability*. Formally, session unlinkability is defined in terms of an experiment $\texttt{Exp}^b_{\mathcal{O}}$, $b \in \{0, 1\}$, involving a probabilistic polynomial-time (PPT) observer $\mathcal{O}$ and the RFID system. Let G^0 and G^1 be two GoTs of size n. $\mathcal{O}$ has access to a history of earlier tag interrogations and is given a group interrogation $\texttt{int}_1$ involving G^0, and an interrogation $\texttt{int}_2$, such that either $i)$ $\texttt{int}_1$ took place before[1] $\texttt{int}_2$ and $\texttt{int}_1$ completed normally (successfully), or $ii)$ an intermediate interrogation involving G^0 completed normally. $\texttt{Exp}^0_{\mathcal{O}}$ corresponds to the event that G^0 was involved in $\texttt{int}_2$ while $\texttt{Exp}^1_{\mathcal{O}}$ corresponds to the event that G^1 was involved in $\texttt{int}_2$.

Definition. A grouping proof provides *session unlinkability* if the advantage of any PPT observer $\mathcal{O}$:

$$Adv_{\mathcal{O}} = |\text{Prob}[\texttt{Exp}^0_{\mathcal{O}} = 1] - \text{Prob}[\texttt{Exp}^1_{\mathcal{O}} = 1]|$$

is negligible (in terms of the security parameter κ), where the probabilities are taken over the coin tosses of $\mathcal{O}$.

This means that in any experiment $\texttt{Exp}^b_{\mathcal{O}}$ involving the interrogations $\texttt{int}_1$ and $\texttt{int}_2$, the observer cannot decide with probability better than negligible whether the group involved is the same G^0 or another G^1.

4.3.3 Impossibility of Full Unlinkability

It is easy to show that under our assumptions in Section 3.2 it is not possible to capture full unlinkability. Assume by contradiction that there is a secure grouping-proof that provides full unlinkability, and that the group G has only two tags: tag_1, tag_2. Let $flow^*$ be the first protocol flow in which tag_1 sends a message to tag_2. Suppose the adversary $\mathcal{A}$ interrupts an authorized interrogation of G at $flow^*$ (our threat model allows for this), causing the execution to abort. Then $\mathcal{A}$ replays the earlier flow of the interrupted interrogation to tag_2. If only tag_2 responds then we lose unlinkability, which is a contradiction. If tag_2 does not respond then we have protocol failure. Again this is a contradiction since we are assuming that the grouping-proof is secure, and the channels are reliable. The only remaining case is for *all* tags to respond (all but one sending pseudorandom strings), but this would violate Assumption 7 of Section 3.2.

1. A temporal relationship, as observed by $\mathcal{O}$. Note that if two interrogations that overlap in time are observed, then it can be asserted that they do not belong to the same tag, since tags are currently technologically limited to single-threaded execution.

4.3.4 Analysis

This protocol is secure in the Universal Composability (UC) framework (details are given in Appendix B). Here again we discuss informally the most common attacks.

Our earlier comments regarding replay attacks and impersonation attacks in Section 4.2.2 apply here as well. Thus we focus on de-synchronization attacks (*DoS attacks*) and traceability.

De-synchronization attacks. If a protocol execution completes successfully, then all tags will share the same counter value. As a result, no tag will accept any previously used PS, which guarantees session unlinkability. Even if the tags do not share the same counter value, there are no synchronization concerns; though some tags (that did not update their counter) will accept a previously used PS.

Traceability. If an adversarial reader interrupts an interrogation preventing some tags from updating their counter, and then re-uses the same values against the non-updated tags, it is able to link these tags to the previous interrogation. However, the power of this attack is limited because after a successful execution, group unlinkability is restored.

Additionally, as mentioned in Section 4.3.1 our protocols leak information about the size of a group—that cannot be prevented easily, and the order of a tag in the group—that can be prevented.

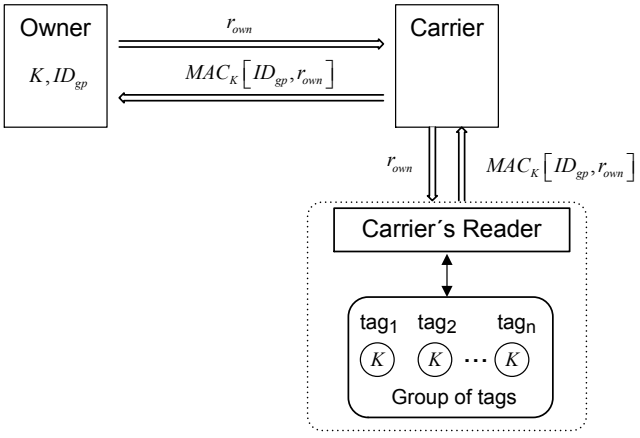
5 Supply-Chain Management

This section describes how grouping-proofs can be used as modules in a high-level setting for supply-chain management. Two operation modes are considered: Mode A in which the owner of tags wants to verify the integrity of a set of tagged goods given to a custodian (carrier), and Mode B in which the custodian wants to verify integrity directly.

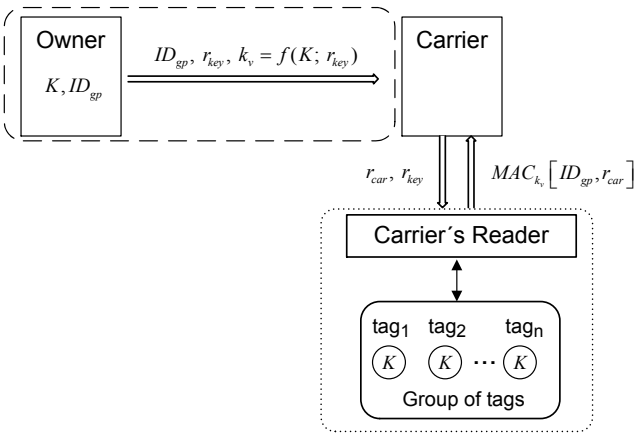
Figure 5 describes protocols for the two modes. For simplicity, the carrier's reader and the GoT are considered as a single entity which generates a grouping-proof according to the protocols described in Section 4. The owner and tags share secret information (the key K), while the carrier does not have any information about the tags other than that given by the owner (and that derived during the execution of the protocol). The owner stores in a database for each GoT an identifier and a private key: (ID_{gp}, K). The arrows "$\Longrightarrow$" correspond to secure communication channels. These are trusted channels between high level entities. In mode A, the carrier is an untrusted entity that may try to deceive the owner by forging a proof.

Mode A

In this mode the owner wants to check the integrity of a consignment that is handled by the carrier, while the carrier is an enabler that just relays messages from the owner to the GoT via his reader. In this mode batch connectivity between the owner and carrier is required. The protocol has four passes:



(a) A. Owner monitoring



(b) B. Carrier monitoring

Fig. 5. Supply-Chain Management Modes

Pass A1-2. The owner selects a random session number r_{own} and sends it to GoT via the carrier:

$$\text{Owner} \Rightarrow \text{Carrier} \Rightarrow \text{GoT}: \ r_{own}$$

Pass A3-4. The reader interrogates tags in its range and the GoT executes a grouping-proof protocol (Section 4) with: $r_{sys} = r_{own}$ and $k = K$. If the interrogation is successful (all tags are present), a proof is generated and sent to the owner, via the carrier:

$$\text{GoT} \Rightarrow \text{Carrier} \Rightarrow \text{Owner}: \ MAC_K[ID_{gp}, r_{own}].$$

Upon receiving the proof, the owner searches in her database for the group identifer—the search in the non-anonymous case is direct while in the anonymous case an exhaustive search is required, and then verifies the proof.

Mode B

In this mode the carrier wants to verify the integrity of a GoT directly.

Pass B0. The owner selects a random number r_{key}, computes the private key $k_v = f(K; r_{key})$ and sends these together with ID_{gp} to the carrier:

$$\text{Owner} \Rightarrow \text{Carrier}: \ (ID_{gp}, r_{key}, k_v).$$

This phase (boxed in the figure) only needs to take place once. The carrier can then challenge the GoT and get a grouping-proof as follows:

Pass B1. The carrier stores the pair (ID_{gp}, k_v), selects a random number r_{car} and sends this along with r_{key} to GoT:

$$\text{Carrier} \Rightarrow \text{GoT}: \ (r_{car}, r_{key}).$$

Pass B2. The authenticator tag of GoT computes k_v using r_{key}, executes a grouping-proof protocol with $r_{sys} = r_{car}$ and $k = k_v$, and sends the proof to the carrier.

$$\text{GoT} \rightarrow \text{Carrier}: \ MAC_{k_v}[ID_{gp}, r_{car}].$$

Upon receiving the proof, the carrier finds the pair (ID_{gp}, k_v) in his database, and verifies the proof.

6 Conclusion

Several RFID grouping-proofs have been proposed in the literature. Most assume communication models and capabilities which either are not properly defined or are not practical.

In this paper we addressed the group scanning problem in a strong adversarial setting. Our contribution was threefold. First, we reviewed the literature related to grouping-proofs and analyzed the Liu et al.'s protocol, recently presented in this journal. Then, we defined a comprehensive framework for grouping-proofs, that includes basic assumptions and criteria for designing practical group scanning. Finally, we proposed two novel grouping proofs that are generated by the tags in a distributed way, without sharing any private information with the reader, and that can be integrated into RFID supply-chain management systems. The security of these protocols is proven in the Universal Composability framework which addresses security under concurrent executions and supports re-usability.

Appendix A
The non-anonymous grouping-proof

The formal security specifications of a grouping-proof capture the requirements for a *proof of simultaneous presence* of the tags of a group. Since RFID tags are often used as components of more complex systems, we use a security framework that supports composability.

The Universal Composability (UC) framework defines the security of a protocol in terms of its simulatability by an idealized functionality $\mathcal{F}$, and supports robust composability with arbitrary protocols [30], [31], [32], [33]. A protocol ρ *UC-realizes* $\mathcal{F}$ if, for every probabilistic polynomial-time adversary $\mathcal{A}$ there is a simulator $\mathcal{S}$ that translates runs of ρ in the real world in the presence of $\mathcal{A}$ into runs controlled by the functionality $\mathcal{F}$ in an ideal world in the presence of an ideal world adversary $\widehat{\mathcal{A}}$ in such a way that: no probabilistic polynomial-time *environment* $\mathcal{Z}$ can distinguish with non-negligible probability whether it is interacting with an instance of ρ with $\mathcal{A}$ or an instance of $\mathcal{F}$ with $\widehat{\mathcal{A}}$.

Group authentication is captured by parties having ideal access to the functionality $\mathcal{F}_{gp}$ (Figure 6). We first describe its basic components and attributes and then its behavior.

Parties. There are three types: verifier, reader and tag. In each subsession, there is a single instance of type reader and arbitrarily many instances of tag. $\mathcal{A}$ and $\mathcal{Z}$ are not UC parties, though $\mathcal{A}$ may control several protocol parties. Upon successful completion of a subsession, the reader generates a grouping-proof.

Sessions. A single session spans the entire lifetime (simulation instance) of the grouping-proof. It consists of several concurrent subsessions, initiated by protocol parties upon receiving input INITIATE from $\mathcal{Z}$. All parties in any subsession are given a unique session identifier sid by $\mathcal{Z}$. While the reader and tags initiate subsessions, $\mathcal{A}$ controls the concurrency and interaction between these subsessions.

Group authentication. Successful group authentication in the real world is a result of sharing common secrets. The choice of group partners is decided by $\mathcal{A}$, who has full control of the network. In the ideal world, this is emulated by invocations of the commands AUTHENTICATE and PROVE.

Activation sequence. $\mathcal{Z}$ is the first entity to be activated, and the last to halt. $\mathcal{Z}$ activates $\mathcal{A}$ and initializes all the protocol parties. The parties instantiate the protocol in the real world. In our protocol and functionality, the receiving party of any message or subroutine output is activated next. If no outgoing message or output is produced in the processing of an incoming message, then by convention $\mathcal{Z}$ is activated. $\mathcal{F}_{gp}$ only specifies the behavior of parties that are not controlled by the adversary, and is activated by an INITIATE input from a reader. For such parties it generates and stores locally messages of type $\texttt{init}(s, \text{reader})$, $\texttt{init}(s', ID)$ or $\texttt{init}(s', ID, i)$, where s, s' are newly created subsession identifier labels and $ID = ID_{gp}$ is a group identifier. These messages are released to the adversary, see Figure 6.

For clarity of presentation in the description of the functionality, we distinguish between messages

Session identifier sid (effectuated by $\mathcal{Z}$)

Upon input INITIATE **at reader:** If reader is adversarial, ignore. Else: delete any `init` record of reader; generate a unique ssid s; record and output: $\texttt{init}(s, reader)$.

Upon input INITIATE **at** tag_1 : If tag_1 is adversarial, ignore. Else: delete any `init`, `aut` or `conf` record of tag_1; generate an ssid s'; record and output: $\texttt{init}(s', ID)$.

Upon input INITIATE **at** $tag_i, i > 1$: If tag_i is adversarial, ignore. Else: if there is a record $\texttt{init}(s', ID)$ then delete any records `init`, `aut`, `conf` of tag_i and record and output: $\texttt{init}(s', ID, i)$.

Upon input AUTHENTICATE **at** tag_n : If there is a record $\texttt{init}(s', ID, n)$ then record and output: $\texttt{aut}(s', ID, n)$; delete the `init` record of tag_n.

Upon input AUTHENTICATE **at** $tag_i : i = n-1, \ldots, 2$. If there are records $\texttt{init}(s', ID, i)$ and $\texttt{aut}(s', ID, i+1)$ then record and output: $\texttt{aut}(s', ID, i)$; delete $\texttt{init}(s', ID, i)$.

Upon input CONFIRM **at** tag_1**:** If there are records $\texttt{init}(s', ID)$ and $\texttt{aut}(s', ID, 2)$ then record and output: $\texttt{conf}(s', ID)$; delete $\texttt{init}(s', ID)$ and all `aut` records.

Upon request PROVE(s, s', ID)**:** If there are records $\texttt{init}(s, reader)$, $\texttt{conf}(s', ID)$ then record and output: $\texttt{proof}(s, s', ID)$; delete the `conf` record.

Upon request IMPERSONATE(s, s', tag_i)**:** If there is a record $\texttt{init}(s, reader)$ and tag_i is an adversarial tag of ID then record and output: $\texttt{proof}(s, s', PS)$.

Upon input VERIFY$(\texttt{proof}(s, s', ID))$ **at verifier:** If there is a record $\texttt{proof}(s, s', ID)$ then output: `valid`.

Fig. 6. Functionality $\mathcal{F}_{gp}$

coming from protocol parties, called *inputs*, and messages coming from the adversary, called *requests*, and for simplicity use the abbreviation *ssid* for subsession identifiers. A grouping-proof is generated by invoking $\mathcal{F}_{gp}$ with the command PROVE. This only succeeds if all tag_i, $i > 1$, in the group are AUTHENTICATED in the same subsession s, s' and tag_1 has CONFIRMED this, in which case $\texttt{proof}(s, s', ID)$ is generated. Finally the adversary can attempt to impersonate tags in the ideal world and forge a proof by using the command IMPERSONATE(s, s', tag); this only succeeds if the impersonated tag is controlled by the adversary. We shall now show that:

Theorem A.1: The non-anonymous grouping-proof UC-realizes the functionality $\mathcal{F}_{gp}$.

Proof: We summarize the key features of the non-anonymous grouping-proof, which represent real world protocol flows:

- The challenge r_{sys} of the reader is a random number generated by the verifier.
- The choice of authenticator tag_1 in any group is hard-coded in tags. Only one tag in a group is the authenticator.
- Tags transmit group identifiers ID_{gp}.
- Communication among tags is mediated by the reader. $\mathcal{A}$ may disrupt/modify traffic on the reader-tag channels of adversarial readers.
- $\mathcal{A}$ cannot tamper with the contents of the reader-verifier channel of a non-adversarial reader.
- Timeouts are implemented on tags.

To emulate real world protocol runs the simulator $\mathcal{S}$ performs the following actions:

- Simulates copies: $\widehat{\mathsf{verifier}}$, $\widehat{\mathsf{reader}}$, $\widehat{\mathsf{tag}}_i$, $\widehat{\mathcal{A}}$, and then activates $\widehat{\mathcal{A}}$.
- Adds/removes keys in a database $\widehat{D}$ of $\widehat{\mathsf{verifier}}$ that contains keys of adversarial/non-adversarial tags.
- Faithfully translates real world protocol flows into their ideal world counterparts for all protocol parties, including actions of the adversary.
- Simulates interactions with $\mathcal{Z}$, *i.e.,* the externally visible part of the protocol. More specifically, $\mathcal{S}$ invokes $\mathcal{F}_{gp}$ with the call INITIATE to instantiate parties, and simulates the ideal world protocol flows invoking $\mathcal{F}_{gp}$ with calls: AUTHENTICATE, CONFIRM, PROVE, IMPERSONATE and VERIFY to generate the corresponding responses (Figure 6), which are then forwarded to the ideal world counterparts of the real world parties, as shown in Figure 7. The request PROVE(s, s', ID) is used when the real world adversary $\mathcal{A}$ forwards unmodified inputs between honest tags and the reader, and the request IMPERSONATE(s, s', tag_i) is used when $\mathcal{A}$ succeeds in authenticating the adversarially controlled tag_i. Note that keys shared between the verifier and the tags are not under the control of the adversary.
- Prevents non-adversarial readers from outputting $\texttt{proof}(s, s', ID_{gp})$ in the ideal world when $\mathcal{A}$ tampers with messages created by non-adversarial tags.

The main difference between the ideal and real world simulations is that in the ideal world, the (non-persistent) values exchanged in tag subsessions are uniformly random and independent, since a true random function is used to evaluate the authenticators and confirmation (Figure 7), whereas in the real world a pseudorandom function f is used.

The specification of $\mathcal{F}_{gp}$ make several security guarantees obvious: unforgeability, freedom from replays and other types of attacks are achieved because to violate them, the adversary would have to guess unseen random, independently generated values. Since the protocol may fail in a variety of ways, to prove our theorem we must ensure that no combination of failures exists, which may enable a probabilistic, polynomial-time environment $\mathcal{Z}$, that selects the initial inputs and observes the final outputs of all parties, and which may interact with the adversary in an arbitrary fashion during the execution of the protocol,

Upon INITIATE **at** $\widehat{\text{reader}}$ **from** $\widehat{\mathcal{A}}$**:** input this to $\mathcal{F}_{gp}$; on output $\texttt{init}(s, reader)$ select a fresh random number r_{sys} from a list in database $\widehat{D}$ and send r_{sys} to $\widehat{\mathcal{A}}$.

Upon INITIATE **on** $\widehat{\text{tag}}_i$ **from** $\widehat{\mathcal{A}}$**:** input this to $\mathcal{F}_{gp}$; on output $\texttt{init}(s', ID)$ assign to $\widehat{\text{tag}}_i$ key (K, k) when $i = 1$ or K when $i > 1$ obtained from $\widehat{D}$, and select a number sn from $\widehat{D}$. Send (i, ID, sn) to $\widehat{\mathcal{A}}$.

Upon AUTHENTICATE **on** $\widehat{\text{tag}}_i, i > 1$, **from** $\widehat{\mathcal{A}}$**:** input this to $\mathcal{F}_{gp}$; on output $\texttt{aut}(s', ID)$, $\widehat{\text{tag}}_i$ computes aut_i as in the non-anonymous grouping-proof, but using a random function instead of f, and sends this to $\widehat{\text{reader}}$ and $\widehat{\mathcal{A}}$.

Upon CONFIRM **on** $\widehat{\text{tag}}_1$ **from** $\widehat{\mathcal{A}}$: input this to $\mathcal{F}_{gp}$; on output $\texttt{conf}(s', ID)$, $\widehat{\text{tag}}_i$ computes mac as in the non-anonymous grouping-proof but using a random function instead of f, and sends this to $\widehat{\text{reader}}$ and $\widehat{\mathcal{A}}$.

Upon PROVE(s, s', ID_{gp}) **at** $\widehat{\text{reader}}$ **from** $\widehat{\mathcal{A}}$**:** input this to $\mathcal{F}_{gp}$; on output $\texttt{proof}(s, s', ID_{gp})$, $\widehat{\text{reader}}$ sends (ID, r_{sys}, mac) to $\widehat{\mathcal{A}}$.

Upon IMPERSONATE(s, s', tag_i) **on** tag_i **from** $\mathcal{A}$: input this to $\mathcal{F}_{gp}$; on output $\texttt{proof}(s, s', ID_{gp})$, send this to $\widehat{\mathcal{A}}$.

Upon VERIFY$(\texttt{proof}(s, s', ID)$ **on** $\widehat{\text{verifier}}$**:** input this to $\mathcal{F}_{gp}$; on output valid, send this to $\widehat{\mathcal{A}}$.

Fig. 7. The simulator $\mathcal{S}$

that can distinguish between real and ideal world simulations.

Observe that if the function f is assumed to be true random, then the transcripts exchanged in tag subsessions are uniformly random and mutually independent. Under this assumption, the real and ideal simulations might differ only when S intervenes to prevent PROVE in the ideal-world, that is, when $\mathcal{A}$ tampers with messages from honest parties (via reflection, re-play, mangling, injection, delay, etc.) and manages to reproduce valid random values (a forgery), which result in a proof in the real-world. This can only happen if there is a collision between the outputs of f: *i.e.*, if it generates the same valid value from unequal inputs (valid and invalid). However, since f is truly random (by assumption), the adversary cannot count on that happening with more than negligible probability. In particular, for each given reader subsession r_{sys} divergence happens with probability at most $2^{-\kappa} m\ell$, where κ is the security parameter, m the total number of groups managed by the verifier, and ℓ the number of interrogations during this subsession. This is because to get a valid proof when one or more of the tags of a group are not present, the adversary $\mathcal{A}$ must select the correct value for the confirmation mac, or the authenticator aut_i, and the probability for this event is $2^{-\kappa}$. For the concurrent grouping-proof we get the same probability, because again $\mathcal{A}$ must select the correct value of mac or aut. Therefore the probability of simulation distinguishability for the grouping-proof is bounded by $2^{-\kappa} mL$, where L

Session identifier sid (effectuated by $\mathcal{Z}$)

Upon input INITIATE **at reader:** If reader is adversarial, ignore. Else: delete any `init` record of reader; generate a unique ssid s; record and output: $\texttt{init}(s, reader)$.

Upon input INITIATE **at** tag_1**:** If tag_1 is adversarial, ignore. Else: delete any `init`, `aut` or `conf` record of tag_1; generate an ssid s' and a random number PS; record and output: $\texttt{init}(s', PS)$.

Upon input INITIATE **at** $tag_i, i > 1$: If tag_i is adversarial, ignore. Else: delete any `init`, `aut` record of tag_i; if there is a record $\texttt{init}(s', PS)$ then record and output: $\texttt{init}(s', PS, i)$.

The actions for inputs AUTHENTICATE, CONFIRM, VERIFY and requests PROVE, IMPERSONATE are similar to those of $\mathcal{F}_{gp}$ (Figure 6) with ID replaced by PS.

Fig. 8. Functionality $\mathcal{F}_{\text{an\_}gp}$

is the number of interrogations through the entire simulation. This is negligible in the security parameter κ.

It follows that if the environment $\mathcal{Z}$ can distinguish real simulations with a pseudorandom function f from ideal simulations then it can also distinguish real simulations with a pseudorandom function from real simulations with a random function. This contradicticts the definition of pseudorandomness since $\mathcal{Z}$ is polynomial-time. □

APPENDIX B
ANONYMOUS GROUPING-PROOF

The functionality $\mathcal{F}_{\text{an\_}gp}$ of our second protocol comprises the behavior expected of a grouping-proof with session unlinkability. This is similar to $\mathcal{F}_{gp}$ (Figure 6) except that in this case the tags return pseudonyms PS instead of the identifier ID_{gp}. $\mathcal{F}_{\text{an\_}gp}$ is described in Figure 8. To maintain synchrony, the tags of a group must share a "loose-synchronized" counter cnt which, for session unlinkability, is refreshed by all tags with each complete interrogation. There are seven inputs/requests: INITIATE, AUTHENTICATE, CONFIRM, PROVE, IMPERSONATE and VERIFY. We shall now show that:

Theorem B.1: The anonymous grouping-proof UC-realizes $\mathcal{F}_{\text{an\_}gp}$.

Proof: As in Theorem A.1, the simulator $\mathcal{S}$ makes copies of $\widehat{\text{verifier}}$, $\widehat{\text{reader}}$, $\widehat{\text{tag}}_i$, $\widehat{\mathcal{A}}$, adds/removes/updates keys and records in a database $\widehat{D}$ of $\widehat{\text{verifier}}$, and faithfully translates real world protocol flows into their ideal world counterparts. However, apart from the forgery case when a valid proof is generated in the real world while it is not in the ideal world, for the anonymous grouping-proof we must also consider a DoS case when a valid proof is generated in the ideal world while it is not in the real world.

This occurs when the adversary $\mathcal{A}$ succeeds in de-synchronizing the tags by modifying some values in the channels (via reflection, re-play, mangling, injection, delay, etc.) and sending the tags valid messages (PS, C) that force them to update their state to an invalid state (*e.g.*, by setting a very high value to cnt).

As in Theorem A.1, if we assume that f is true random then this can only happen if there is a collision between the outputs of f, which only happens with negligible probability (bound by $2^{-\kappa}mL$). So the probability of simulation distinguishability is neglible. If follows that if $\mathcal{Z}$ can distinguish real simulations with a pseudorandom function from ideal simulations then it can also distinguish real simulations with a pseudorandom function from real simulations with a random function. This is not possible since $\mathcal{Z}$ is polynomial-time. □

ACKNOWLEDGMENTS

The authors would like to thank...

REFERENCES

[1] K. Ashton, "That 'Internet of Things' Thing," *RFID Journal*, 2009.

[2] K. Finkenzeller, *RFID Handbook : Fundamentals and Applications in Contactless Smart Cards and Identification*, 2nd ed. John Wiley & Sons, May 2003.

[3] EPC Global "UHF Air Interface Protocol Standard Generation2/Version2," http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2.

[4] G. Kapoor and S. Piramuthu, "Single RFID Tag Ownership Transfer Protocols," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 42, no. 2, pp. 164–173, 2012.

[5] F. G. Jorge Munilla and W. Susilo, "Cryptanalysis of an EPCC1G2 Standard Compliant Ownership Transfer Protocol," *Wireless Pers Commun*, no. 72, pp. 245–258, 2013.

[6] H. Liu, H. Ning, Y. Zhang, D. He, Q. Xiong, and L. T. Yang, "Grouping-proofs-based authentication protocol for distributed rfid systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1321–1330, 2013.

[7] M. Burmester and J. Munilla, *Security and Trends in Wireless Identification and Sensing Platform Tags: Advancements in RFID*. IGI Global, 2013, ch. RFID Grouping-Proofs.

[8] A. Juels, ""Yoking-proofs" for RFID tags," in *PERCOMW '04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 138–142.

[9] J. Saito and K. Sakurai, "Grouping proof for RFID tags," in *19th International Conference on Advanced Information Networking and Applications, AINA 2005.*, vol. 2, March 2005, pp. 621–624.

[10] A. Juels, "Generalized "yoking-proofs" for a group of RFID tags," in *MOBIQUITOUS 2006*, 2006.

[11] S. 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*, IEEE. Lyon, France: IEEE Computer Society Press, June 2006.

[12] M. Burmester, B. de Medeiros, and R. Motta, "Provably Secure Grouping-Proofs for RFID Tags," in *CARDIS*, ser. Lecture Notes in Computer Science, G. Grimaud and F.-X. Standaert, Eds., vol. 5189. Springer, 2008, pp. 176–190.

[13] H.-H. Huang and C.-Y. Ku, "A rfid grouping proof protocol for medication safety of inpatient," *Journal of Medical Systems*, 2008.

[14] H.-Y. Chien, C.-C. Yang, T.-C. Wu, and C.-F. Lee, "Two rfid-based solutions to enhance inpatient medication safety," *Journal of Medical Systems*, 2009.

[15] P. Peris-Lopez, A. Orfila, J. C. Hernandez-Castro, and J. C. A. van der Lubbe, "Flaws on rfid grouping-proofs. guidelines for future sound protocols," *J. Netw. Comput. Appl.*, vol. 34, no. 3, pp. 833–845, May 2011. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2010.04.008

[16] Z. Ahmadian, M. Salmasizadeh, and M. Aref, "Recursive linear and differential cryptanalysis of ultralightweight authentication protocols," *Information Forensics and Security, IEEE Transactions on*, vol. 8, no. 7, pp. 1140–1151, 2013.

[17] D. Paret, *RFID and Contactless Smart Card Applications*. John Wiley & Sons, 2005.

[18] International Organization for Standardization, "ISO/IEC 29192-1:Information Technology- Security Techniques - Lightweight cryptography - Part 1: General. ISO/IEC, 2012."

[19] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.

[20] ISO/IEC, "Standard # 18000 – RFID Air Interface Standard," http://www.hightechaid.com/standards/18000.htm.

[21] D. Dolev and A. C.-C. Yao, "On the Security of Public Key Protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–207, 1983.

[22] S. Bengio, G. Brassard, Y. Desmedt, C. Goutier, and J.-J. Quisquater, "Secure implementations of identification systems," *J. Cryptology*, vol. 4, no. 3, pp. 175–183, 1991.

[23] S. Mangard, T. Popp, and M. E. Oswald, *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer, 2007, vol. (ISBN: 0-387-30857-1).

[24] C. H. Kim, G. Avoine, F. Koeune, F.-X. Standaert, and O. Pereira, "The Swiss-Knife RFID Distance Bounding Protocol," in *ICISC*, ser. Lecture Notes in Computer Science, P. J. Lee and J. H. Cheon, Eds., vol. 5461. Springer, 2008, pp. 98–115.

[25] J. Munilla, A. Ortiz, and A. Peinado, "Distance Bounding Protocols with Void-Challenges for RFID," in *Workshop on RFID Security – RFIDSec'06*. Graz, Austria: Ecrypt, July 2006.

[26] D. N. Duc and K. Kim, "On the security of rfid group scanning protocols." *IEICE Transactions*, vol. 93-D, no. 3, pp. 528–530, 2010.

[27] M. Burmester, B. de Medeiros, J. Munilla, and S. Peinado., "Secure EPC Gen2 Compliant Radio Frequency Identication," International Association for Cryptological Research, Tech. Rep. E-print #2009/147, 2009. [Online]. Available: http://eprint.iacr.org/2009/149

[28] T. V. Le, M. Burmester, and B. de Medeiros, "Universally composable and forward-secure RFID authentication and authenticated key exchange," in *ASIACCS*, F. Bao and S. Miller, Eds. ACM, 2007, pp. 242–252.

[29] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits," *SIAM J. Comput.*, vol. 13, no. 4, pp. 850–864, 1984.

[30] O. Goldreich, S. Micali, and A. Widgerson, "How to play any mental game," in *19th Symposium on Theory of Computing (STOC 1987)*. ACM Press, 1987, pp. 218–229.

[31] D. Beaver, "Foundations of secure interactive computing," in *Proc. Advances in Cryptology (CRYPTO 1991)*, ser. LNCS, vol. 576. Springer, 1991, pp. 377–391.

[32] R. Canetti, "Security and composition of multi-party cryptographic protocols," *Journal of Cryptology*, vol. 13:1, pp. 143–202, 2000.

[33] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proc. IEEE Symp. on Foundations of Computer Science (FOCS 2001)*. IEEE Press, 2001, pp. 136–145.