Journal of Network and Computer Applications 34 (2011) 833–845

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Flaws on RFID grouping-proofs. Guidelines for future sound protocols

Pedro Peris-Lopez [a,*], Agustin Orfila [a,b], Julio C. Hernandez-Castro [c], Jan C.A. van der Lubbe [a]

[a] *Information Security and Privacy Lab, Faculty of Electrical Engineering, Mathematics, and Computer Science, Delft University of Technology, P.O. Box 5031, 2600 GA, Delft, The Netherlands*
[b] *Department of Computer Science, Carlos III University of Madrid, Leganes, Madrid 28911, Spain*
[c] *School of Computing, Buckingham Building, Lion Terrace, Portsmouth PO1 3HE, UK*

ARTICLE INFO

Article history:
Received 31 October 2009
Received in revised form
25 March 2010
Accepted 28 April 2010
Available online 7 May 2010

Keywords:
RFID
Grouping-proof
Security
Privacy
Cryptanalysis

ABSTRACT

During the last years many RFID authentication protocols have been proposed with major or minor success (van Deursen and Radomirović, 2008). Juels (2004) introduced a different and novel problem that aims to evidence that two tags have been simultaneously scanned. He called this kind of evidence a yoking-proof that is supposed to be verifiable offline. Then, some authors suggested the generalization of the proof for a larger number of tags. In this paper, we review the literature published in this research topic and show the security flaws of the proposed protocols, named RFID grouping-proofs generally. More precisely, we cryptanalyze five of the most recent schemes and we also show how our techniques can be applied to older proposals. We provide some guidelines that should be followed to design secure protocols and preclude past errors. Finally, we present a yoking-proof for low-cost RFID tags, named Kazahaya, that conforms to the proposed guidelines.

© 2010 Elsevier Ltd. All rights reserved.

1. Introduction

A typical RFID system consists of three different types of entities: tags, readers and a verifier. The tags are embedded in, or attached to, objects to be identified. The most expensive are active, i.e. have power supply (usually a battery) that is used to energize the microchip's circuitry and to broadcast a signal to the reader. As they have their own power source, active tags support large memory and processing capabilities. Semi-passive tags, which are also too expensive to place on low-cost items, use a battery to run the microchip's circuitry but communicate by drawing power from the reader. The remaining ones are passive, i.e., have no internal power source neither to energize the microchip nor to communicate to the reader. Thus, the computation and communication capabilities of the latter are very limited. Nevertheless, it is generally assumed that they are able to perform basic cryptographic operations such as generating pseudo-random numbers and evaluating pseudo-random functions (Burmester et al., 2008). RFID tags do not have clocks. However, the activity time of a tag during a single session can be limited using techniques such as measuring the discharge rate of capacitors, as described in Juels (2004). Accordingly timeouts can be implemented on RFID passive tags. FCC regulations require the termination of tag-reading within 400 ms. The readers provide power to the tags in order to communicate with them. The verifier (a back-end server) is a trusted entity that maintains a database containing the information needed to identify tags (e.g. their unique identifiers and their secret keys).

A grouping-proof is an evidence that two or more RFID tags were scanned simultaneously by a reader within its broadcast range. For example, in the pharmaceutical sector, it can prove that a medicine has been sold with its prescription or with the patient information leaflet. The proof should be verifiable by the corresponding verifier. During a grouping-proof protocol execution, the verifier can be in two different modes: online or offline. In the first mode the verifier can send and receive messages from specific tags (via the reader) throughout the protocol execution. In contrast, in offline mode the verifier can only broadcast challenges to the reader. Thus, the verifier in offline mode never unicasts messages to tags. Although it is straightforward to design solutions for the online mode (indeed a proper RFID authentication protocol is enough (Chien et al., 2010), some research has focused on the protocol design for this mode (Leng et al., 2009; Huang and Ku, 2009; Chien et al., 2010). Nevertheless, the interesting case is the offline mode because it does not need the persistent presence of the verifier to generate grouping-proofs.

Some assumptions are generally accepted for the design of grouping-proofs (Burmester et al., 2008):

- RFID readers are potentially untrusted. The only trusted entity is a verifier.

* Corresponding author. Tel.: +31 15 27 87241.
E-mail addresses: P.PerisLopez@tudelft.nl (P. Peris-Lopez), A.OrfilaDiazPabon@tudelft.nl (A. Orfila), Julio.Hernandez-Castro@port.ac.uk (J.C. Hernandez-Castro), J.C.A.vanderLubbe@tudelft.nl (J.C. van der Lubbe).

1084-8045/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2010.04.008

- RFID readers keep a record of proofs for each session. These records cannot be manipulated by the adversary. In the offline case readers must also store private information regarding interrogation challenges obtained from the verifier.
- The verifier is a trusted entity that may share some secret information with the tags such as cryptographic keys. The verifier has a secure channel (private and authenticated) that links it to the (authenticated) RFID readers. In contrast, the channel between tags and the reader is considered insecure.
- For the protocol design of grouping-proofs, the focus is generally set on security issues at the protocol layer and not on physical or link layer issues. Problems such as the coupling design, the power-up and collision arbitration processes or the air-RFID interface are not usually addressed.

There are practical scenarios where grouping-proofs could significatively expand the capabilities of RFID-based systems. For example, (1) in the pharmaceutical sector to prove that a medicine is sold joined with its prescription or with its information leaflet; (2) in the government paperwork to check that a single form is enclosed with its corresponding stamp or label; (3) in meetings or access control systems to generate an evidence that a group of people are present at a specific location. In airport check-in desks to link your boarding card with your passport and baggages; (4) in auto-lending library services to associate a book with the e-identity card of an user.

The remainder of this paper is organized as follows. Section 2 presents a critical review of the related work. Next, Section 3 shows a flaw in the protocols proposed by Burmester et al. (2008) that is also present in other yoking protocols. Section 4 discusses the traceability problems of the anonymous protocol proposed by Chien and Liu (2009). Then, Section 5 describes specific attacks on two online proposals and Section 6 show how some of the protocols that have been proposed recently are vulnerable to replay attacks. Consequently, Section 7 proposes some guidelines for future sound grouping-proofs that avoid the security pitfalls described in this paper. Next, Section 8 presents a novel RFID grouping-proof, named Kazahaya, which complies with the guidelines previously defined. Finally the article ends up with the main conclusions.

2. Related work

The idea of generating an evidence that a pair of tags has been scanned simultaneously was introduced by Juels (2004). He named such evidence a yoking-proof and he proposed two protocols to generate it. The first requires more expensive tags while the second is thought for severely resource-constrained ones. Saito and Sakurai (2005) showed that the second protocol is not immune to replay attacks and Bolotnyy and Robins (2006) extended the attacks to the first. In addition, Burmester et al. (2008) pointed out two additional weaknesses: Denial-of-service (DoS) and impersonation attacks were feasible. Yoking-proofs have been extended to prove simultaneous presence of a group of tags in the range of an RFID reader (Saito and Sakurai, 2005). They called this kind of proof a grouping-proof. In order to thwart replay attacks, both for two or a group of tags, their protocol includes timestamps. Nevertheless, Piramuthu (2006) showed Saito's protocol is also vulnerable to replay attacks. Accordingly, he proposed to include random values instead of timestamps. This is important, because timestamps can be predicted, allowing an attacker to collect prior responses and combine them to forge proofs. However, Peris-Lopez et al. (2007) proved Piramithu's protocol used random numbers in such a way that it is vulnerable to multi-proof session replay attacks. Accordingly, Peris-Lopez et al. (2007) also proposed a new grouping-proof protocol.

Since then several grouping-proof protocols have been proposed that have not been cryptanalyzed yet. In this paper, we will examine most recent proposals in depth and we will identify flaws in their design. Then, guidelines for proper grouping-proof protocol design are provided according to the previous analysis.

Cho et al. (2008) proposed a variant of Piramithu's protocol to make it immune to brute force attacks. However, it turns out to be vulnerable to multi-proof session replay attacks. Lin et al. (2007) propose both online and offline grouping-proof protocols to avoid race conditions and to face the problem of determining if tags that are supposed to be present in a grouping-proof are missing. However, the offline protocol is vulnerable to multiple impersonation attack, as describe in Section 3. Burmester et al. (2008) proposed three new protocols. First introduces the utility of group keys to avoid the generation of useless proofs (a kind of DoS) in such a way that a tag can verify that the other belongs to the group before generating an offline proof. Second achieves an anonymous yoking-proof and last (online) provides anonymity and forward security. We will show in Section 3 that they are all vulnerable to multiple impersonation attacks.

The idea of anonymous grouping-proofs was first introduced by Bolotnyy and Robins (2006). Burmester et al. (2008) stated that in Bolotnyy's schema is not clear how the reader can pair tags from their pseudonyms and propose an alternative. Peris-Lopez et al. (2007) and Chien and Liu (2009) also proposed anonymous yoking protocols.

Lien et al. (2008) introduced a reading order independent grouping-proof to improve efficiency and reduce failure rates. Leng et al. (2009) proposed a select-response grouping-proof protocol. They argue that previous proposals assume that a reader queries and computes a proof for the verifier. Thus, there is no information for the reader to judge the completeness of the proof in advance (there can be missing tags due to transmission errors, ineffective tags or interruption attacks, etc.). In addition, DoS is also possible since a malicious tag can avoid a legitimate proof to be generated or force a useless proof to be created. To overcome these problems, they propose an online protocol where the verifier is involved in each step instead of waiting. However, as Chien et al. (2010) stated, if an online verifier is available it can directly authenticate and verify the presence of each tag, and an RFID authentication protocol is usually simpler and more efficient than a grouping-proof protocol as it runs an authentication instance for each tag, respectively, and independently. Accordingly, Chien et al. (2010) present an authentication protocol for this purpose that improves a previous grouping-proof protocol for enhancing inpatient medication safety using online mode (Huang and Ku, 2009). Both protocols conforms to EPCglobal Class-1 Gen-2 specification and both protocols have certain weaknesses as we will discuss later in this paper (see Section 5). Chien et al. (2010) also proposes an offline grouping-proof protocol. Unfortunately it is vulnerable to replay attacks as we show in Section 6.

Finally, Chien and Liu (2009) proposed an anonymous tree based yoking proof to reduce the computational cost of identifying a tag in the verifier from $O(N)$ to $O(1)$. However, this protocol has serious privacy weaknesses as we show in Section 4. Amongst others, it is possible to track when two tags of a certain group are being scanned together to generate a proof.

It is important to agree on a common way of naming grouping-proofs because different names can be found in the literature: yoking-proofs (Juels, 2004), grouping-proofs (Saito and Sakurai, 2005), existence-proofs (Piramuthu, 2006), clumping-proofs (Peris-Lopez et al., 2007), and coexistence-proofs (Lin et al., 2007). In this paper, we refer to yoking-proofs when the protocol concerns exclusively two tags and grouping-proofs when the protocol is designed for a set of tags.

3. Multiple impersonation attack

Burmester et al. (2008) proposed three RFID protocols in strong adversary models. The first scheme does not guarantee anonymity, the second supports anonymity and finally the third adds forward security to the set of properties supported by the second. In this section, we analyze the first of these proposals, named a robust grouping-proof. Although the reader is referred to the original paper for a description of the two last protocols, the attack described against the first scheme can be also put in action over the later ones.

3.1. *Robust grouping-proof (Burmester et al., 2008)*

Tags are divided into groups, which are identified by a group identifier (ID_{group}). Each tag stores two private keys: a group key (K_{group}) proves its membership to a specific group, and a secret key (K_{tag}) facilitates the authentication of tags. For each tag, the above information is stored in a database ($D=\{ID_{group}, K_{tag}, K_{group}\}$). The protocol has three phases (as described at the top part of Fig. 1):

First phase: The reader broadcasts a random challenge r_{sys}, which is generated by a trusted verifier at regular intervals. The tags in its range backscatter their group identifier ID_{group}.

Second phase: Tags are linked by channels to the reader. Note that this phase takes place at data link layer.

Third phase: In these phase, Tag_A plays the role of the initiator of the proof. The counter c of this tag determines the current state of the group and is updated during every protocol execution. The messages exchanged between two tags to prove their simultaneous reading is described below. f symbolizes a pseudo-random function and $\|$ denotes concatenation operator.

(1.0) Tag_A computes $r_A \| s_A = f(K_{group}; r_{sys}, c)$ and sends $\{r_A, c\}$ to the reader. Finally, Tag_A updates its counter ($c = c+1$).

(1.1) The reader stores the received values and forwards them to Tag_B.

(2.0) Tag_B computes its local version of the token ($r_B \| s_B = f(K_{group}; r_{sys}, c)$) to prove Tag_A belongs to the group. If $r_A \neq r_B$ the protocol is aborted. Otherwise, Tag_B computes an authentication message $x_B = f(K_B; r_{sys} \| r_B)$ and sends the pair $\{s_B, x_B\}$ to the reader.

(2.1) The reader stores x_B and forwards s_B value to Tag_A.

(3.0) Tag_A checks if Tag_B belongs to the group ($s_A = s_B$). If so, Tag_A computes its authentication message $x_A = f(K_A; r_{sys} \| r_A)$ and sends this token to the reader. Otherwise, the protocol is aborted.

(4.0) The completion of the proof is achieved and the reader builds the grouping-proof ($P_{AB} = (r_{sys}, ID_{group}, c, r_A, s_A, x_A, x_B)$).

3.1.1. *Multiple impersonation attack on robust grouping-proof*

We now show that the robust grouping-proof is vulnerable to *multiple impersonation attack*. Basically, an adversary can generate a proof of simultaneous reading of Tag_B and Tag_X, where X symbolizes any tag belonging to the same group. In order to perform this attack, an adversary has to eavesdrop the public messages transmitted over the insecure radio channel during the execution of a proof between an initiator Tag_A and the target Tag_B. Then, the captured messages are replayed to Tag_X to build a counterfeit proof. Following, the attack, which has two stages, is described in detail (see Fig. 1).

Stage 1. *Legitimate robust grouping-proof*:

First and second phase: $\{Tag_A, Tag_B\}$ are linked and r_{sys} represents the random number associated with the current session.

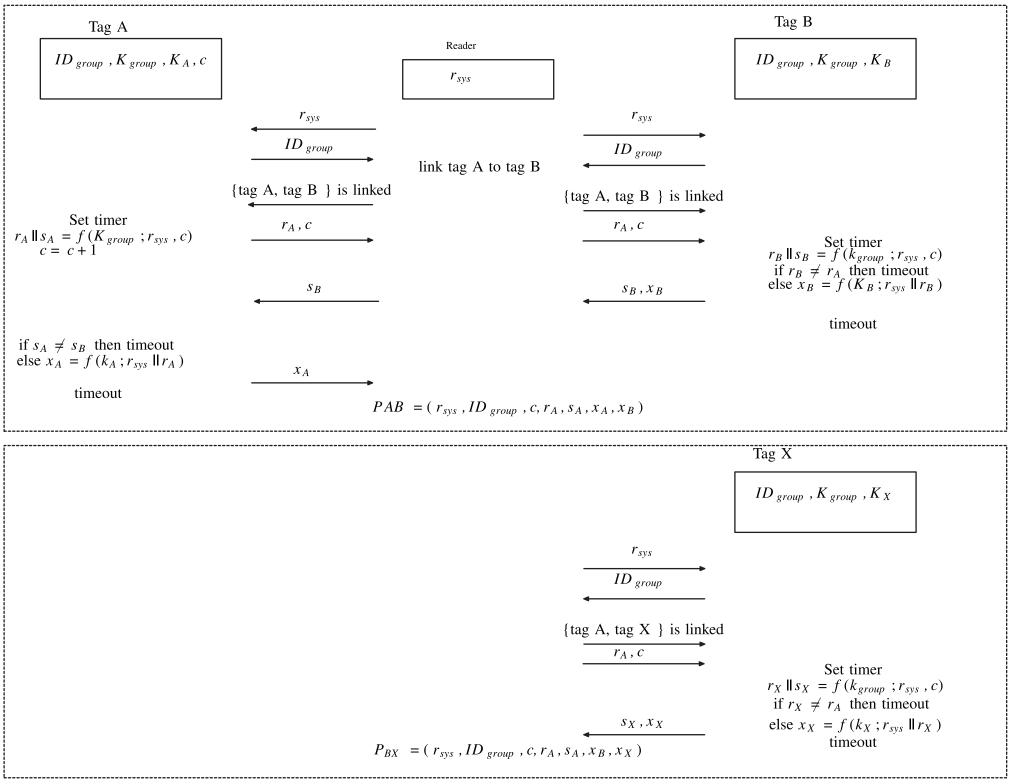
Tag A
$ID_{group}, K_{group}, K_A, c$
Reader
r_{sys}
Tag B
$ID_{group}, K_{group}, K_B$
r_{sys}
ID_{group}
link tag A to tag B
{tag A, tag B } is linked
r_A, c
Set timer
$r_A \| s_A = f(K_{group}; r_{sys}, c)$
$c = c+1$
Set timer
$r_B \| s_B = f(k_{group}; r_{sys}, c)$
if $r_B \neq r_A$ then timeout
else $x_B = f(K_B; r_{sys} \| r_B)$
timeout
s_B
s_B, x_B
if $s_A \neq s_B$ then timeout
else $x_A = f(k_A; r_{sys} \| r_A)$
timeout
x_A
$PAB = (r_{sys}, ID_{group}, c, r_A, s_A, x_A, x_B)$
Tag X
$ID_{group}, K_{group}, K_X$
{tag A, tag X } is linked
Set timer
$r_X \| s_X = f(k_{group}; r_{sys}, c)$
if $r_X \neq r_A$ then timeout
else $x_X = f(k_X; r_{sys} \| r_X)$
timeout
s_X, x_X
$P_{BX} = (r_{sys}, ID_{group}, c, r_A, s_A, x_B, x_X)$

Fig. 1. Multi-proof replay attack on Burmester's scheme.

Third phase: Each tag checks the membership of the other participant and generates an authentication message. The messages exchanged between Tag_A and Tag_B to proof their simultaneous reading is described below.

(1.0) Tag_A computes $r_A \| s_A = f(K_{group}; r_{sys}, c)$ and sends $\{r_A, c\}$ to the reader. Finally, Tag_A updates its counter ($c = c+1$).
(1.1) The reader stores the received values and forwards them to Tag_B.
(2.0) Tag_B computes its local version of the token ($r_B \| s_B = f(K_{group}; r_{sys}, c)$) to prove Tag_A belongs to the group. If $r_A \neq r_B$ the protocol is aborted. Otherwise, Tag_B computes an authentication message $x_B = f(K_B; rys \| r_B)$ and sends the pair $\{s_B, x_B\}$ to the reader.
(2.1) The reader stores x_B and forwards s_B value to Tag_A.
(3.0) Tag_A checks if Tag_B belongs to the group ($s_A = s_B$). If so, Tag_A computes its authentication message $x_A = f(K_A; r_{sys} \| r_A)$ and sends this token to the reader. Otherwise, the protocol is aborted.
(4.0) The completion of the proof is achieved and the reader builds the grouping-proof ($P_{AB} = (r_{sys}, ID_{group}, c, r_A, s_A, x_A, x_B)$).

At this point of the attack, the adversary knows the values $\{ID_{group}, c, r_A, s_A, x_A, x_B\}$ passed over the insecure channel. The knowledge of these values can be exploited by an attacker to generate a proof of the simultaneous reading of Tag_B and any other tag (Tag_X) of the group. The procedure followed by the attacker is described below.

Stage 2: *Multiple impersonation attack*:

First and second phase: The impersonated Tag_A and Tag_X are linked and r_{sys} represents the random number associated with the current session.

Third phase: The messages exchanged between the adversary — impersonating Tag_A — and Tag_X are described below:

(1.0) The adversary replays the tuple $\{r_A, c\}$ to Tag_X.
(2.0) Tag_X computes its local version of the token ($r_X \| s_X = f(K_{group}; r_{sys}, c)$) to prove Tag_A belongs to the group. As $r_A = r_X$, the tag computes an authentication message $x_X = f(k_X; r_{sys} \| r_X)$ and sends the tuple $\{s_X, x_X\}$ to the adversary.
(3.0) The adversary builds the bogus grouping-proof ($P_{BX} = (rys, ID_{group}, c, r_B, s_B, x_B, x_X)$).[1]

The attacker may thus deceive the verifier into thinking that Tag_B and Tag_X are read simultaneously. Indeed, the attack is viable as the robust grouping-proof does not satisfy one of the premises necessary to generate a secure grouping-proof (see Guideline 2). Basically, the attacker exploits the fact that, in step (3.0) of the protocol, the authentication message computed by the initiator tag is independent of the other participant computations. Readers should note that the other two protocols proposed (a robust anonymous grouping-proof and a robust grouping-proof with forward secrecy) suffer from the same weaknesses as they follow the same scheme.

Finally, we have examined how to apply these attacks to different grouping-proofs found in the literature. The attack results completely effective against recent proposals such as the yoking-proof (Juels, 2004), the groping-proof (Saito and Sakurai, 2005) and the enhanced yoking-proof (Lin et al., 2007).

Countermeasure: As dictated by Guideline 2 in Section 7, when just two tags are involved, the computations of a tag — except the first message in the protocol — should be dependent on the values computed by the fellow tag participating in the proof.

[1] Similarly, a counterfeit grouping-proof can be also built at this point yoking Tag_A and Tag_X ($P_{AX} = (r_{sys}, ID_{group}, c, r_A, s_A, x_A, x_X)$).

4. Privacy attacks

One of the fundamental issues still to be addressed on RFID systems is privacy. Products labeled with tags reveal sensitive information, such as their static identifier, when queried by readers, and they do it indiscriminately. A problem closely related to privacy is tracking, or violations of location privacy. Most of the times, tags provide always the same identifier, which will allow a third party to easily establish an association between a given tag and its holder or owner. Even in the case in which tags try not to disclose their identity, there are situations where, by using a constellation of tags, this tracing is still possible (Juels, 2006).

In most of the grouping-proof proposals tag identifier is sent in plaintext (e.g. Piramuthu, 2006; Lin et al., 2007; Chien et al., 2010). As a consequence, privacy is compromised. Those few schemes that face the problem propose some kind of pseudonym to hide the identity of tags during transmission. Although this is a solution for anonymity it is not, by itself, for traceability. This is the case of Chien and Liu (2009) that presents serious traceability problems. First we introduce their protocol and then we explain four possible traceability attacks.

4.1. Tree-based yoking-proof (Chien and Liu, 2009)

Chien and Liu (2009) proposed a tree based yoking proof to reduce the computational cost of identifying a tag in the back-end server (verifier) from $O(N)$ to $O(1)$. In their proposal the tags are assigned to the leaves of a tree and the path from the root to the leaf serves as the identity of a tag. Those tags belonging to the same group are assigned to the same subtree and those tags of the same group (within the same sub-tree) are potentially to be yoked. It is assumed that the verifier is offline and the channel between the reader and the verifier is secure. Chien's proof only allows a yoking-proof of two tags of the same group. However, it may be useful to generate a proof of tags in different groups. This situation could not be managed by Chien's protocol.

The identity of a tag is represented by a path that is split into two parts. The first (namely $path^1$) serves as the group identity and the second (namely $path^2$) is used to identify the tag of the group. Fig. 2 shows one example of the tree organization of tags, where the triangles with dash lines denote groups. Each tag, say Tag_i, has its group identity $Path^1_{T_i}$ and its distinct tag identity $Path^2_{T_i}$. It also has three keys which are gk_{GY}, lk_{Ti}, rk, where gk_{GY} is the shared group key of the same group, and lk_{Ti} is Tag_i's secret key and rk is a root key shared by every tags. The verifier (back-end server) maintains all the paths and the secret keys of every tag, while the reader only keeps the group level information of each tag; that is rk, $Path^1_{T_i}$ and gk_{GY}.

The reader periodically receives an authenticated random number, r_{sys}, from the verifier. It is used to verify whether two tags were simultaneously scanned. Table 1 summarizes the notation used and Fig. 3 shows the protocol, which consists of two phases. In the first phase (Rounds 1–3) the reader, based on the group key and the responses from the tags, links the tags from the same group. In the second phase (Rounds 4–8) reader and tags cooperate to generate the evidence of the simultaneous presence of two tags within the specified time window. In detail the rounds can be described as follows:

Round 1: $Reader \rightarrow Tag_A$ and Tag_B: r_{sys}.

The reader receives the authenticated r_{sys} from the verifier and forwards it to Tag_A and Tag_B.

Round 2: $Tag_A \rightarrow Reader$: r_A, P'_{T_A}, h_A; $Tag_B \rightarrow Reader$: r_B, P'_{T_B}, h_B.

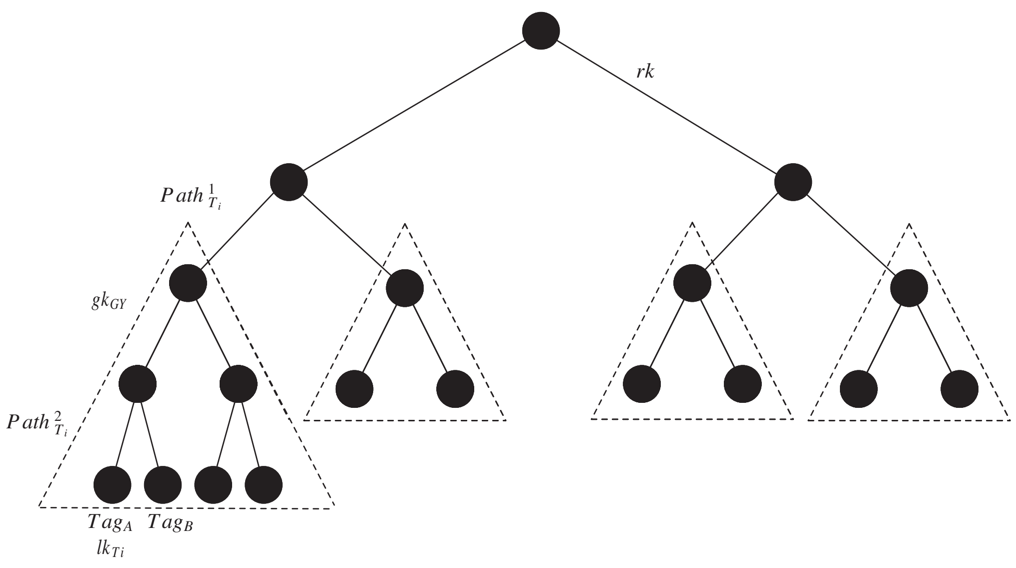

Fig. 2. Tree organization of group of tags.

Table 1
Notation of tree-based yoking protocol.

Notation	Description
r_{sys}	Random number from the verifier
gk_{GY}	The key of a group
$Path^1_{T_i}$	The identity of a group; that is the path from the root to the first node of the group
$h()$	Hash function
lk_{T_i}	The secret key of a leaf node (Tag_i)
rk	The root key
$Path^2_{T_i}$	The path from the first node of a group to the leaf node of the tag Tag_i
r_i	Random number generated by Tag_i

When Tag_A and Tag_B receive r_{sys}, Tag_A computes $P'_{T_A} = h(rk) \oplus Path^1_{T_A}$ and $h_A{=}h(gk_{GY}, r_{sys}, r_A)$, and, $P'_{T_B} = h(rk) \oplus Path^1_{T_B}$ and $h_B{=} h(gk_{GY}, r_{sys}, r_B)$, where r_A and r_B are two random numbers chosen by Tag_A and Tag_B, respectively. From the received P'_{T_A} and P'_{T_B} the reader is able to derive $Path^1_{T_A}$ and $Path^1_{T_B}$ because she can compute $h(rk)$. Thus, she verifies whether both tags belong to the same group. If so, the reader uses the corresponding group key gk_{GY} to verify h_A and h_B. If the verification succeeds, the reader is convinced that the tags belong to the same group, and proceeds to Round 3 to link the two tags; otherwise, it stops the protocol.

Round 3: $Reader \rightarrow Tag_A$ and Tag_B: h_A, h_B, $h(gk_{GY}, h_A, h_B)$.

The reader sends to the tags h_A, h_B, $h(gk_{GY}, h_A, h_B)$ from which they can verify whether they belong to the same group, and, if so, they proceed to Rounds 4–8 to co-operatively generate the evidence of the simultaneous presence.

Round 4: $Tag_A \rightarrow Reader$: P''_{T_A}, a_1.

Tag_A computes $P''_{T_A} = h(gk_{GY}, r_{sys}) \oplus Path^2_{T_A}$ to securely convey its identity $Path^2_{T_A}$ and calculates $a_1 = h(lk_{T_A}, h_A, h_B, r_{sys})$.

Round 5: $Reader \rightarrow Tag_B$: a_1.

The reader forwards a_1 to Tag_B.

Round 6: $Tag_B \rightarrow Reader$: P''_{T_B}, b.

Tag_B computes $P''_{T_B} = h(gk_{GY}, r_{sys}) \oplus Path^2_{T_B}$ to securely convey its identity $Path^2_{T_B}$. It computes $b = h(lk_{T_B}, h_A, h_B, a_1, r_{sys})$ as well.

Round 7: $Reader \rightarrow Tag_A$: b.

The reader forwards b to Tag_A.

Round 8: $Tag_A \rightarrow Reader$: a_2.

Tag_A computes $a_2{=}\ h(lk_{TA}, h_A, h_B, b, r_{sys})$ and sends it to the reader. The final proof P_{AB} consists of $\{r_{sys}, P''_{T_A}, P''_{T_B}, h_A, h_B, a_1, a_2, b\}$, from which the verifier later can certify the simultaneous presence of the two tags Tag_A and Tag_B.

4.1.1. Traceability attacks to tree-based yoking-proof

Chien and Liu (2009) claim that their protocol is unsusceptible to traceability attack. They argue that the encrypted identity $P''_{T_A} = h(gk_{GY}, r_{sys}) \oplus Path^2_{T_A}$ is random and independent for different sessions what avoids traceability. Following we show that four traceability attacks are possible. Three are passive and one is active.

First, an eavesdropper is able to know if two tags (T_A and T_B) that are being scanned simultaneously belong to the same group or not. In order to do this it is only necessary to eavesdrop P'_{T_A} and P'_{T_B} in Round 1. If $P'_{T_A} \oplus P'_{T_B} = 0$ then T_A and T_B belong to the same group, else they do not. This is due to the involution property of XOR operator: $P'_{T_A} \oplus P'_{T_B} = h(rk) \oplus Path^1_{T_A} \oplus h(rk) \oplus Path^1_{T_B} = Path^1_{T_A} \oplus Path^1_{T_B}$. If $P'_{T_A} \oplus P'_{T_B} = 0$ then $Path^1_{T_A} = Path^1_{T_B}$.

Second, if an eavesdropper captures the message P'_{T_i} of two different sessions, she is able to know if both were sent from tags of the same group. In such a case the XOR of the messages would be zero ($P'^{old}_{T_i} \oplus P'^{current}_{T_i} = 0$).

Third, it is also possible to track tags on a pair basis. In other words, it is possible to know if a current attempt to generate a grouping-proof of two tags in a group (e.g. P_{AB}) corresponds to past attempts. An eavesdropper is able to fingerprint pair associations in a group and recognize them afterwards when a value is repeated. Formally, from Rounds 4 and 6, an eavesdropper is able to sniff P''_{T_A} and P''_{T_B}. An XOR of both values results in $P''_{T_A} \oplus P''_{T_B} = h(gk_{GY}, r_{sys}) \oplus Path^2_{T_A} \oplus h(gk_{GY}, r_{sys}) \oplus Path^2_{T_B} = Path^2_{T_A} \oplus\ Path^2_{T_B}$. This value can be stored and if the same value is computed in the future (from new eavesdropped messages) then the attacker knows that a grouping-proof of the same pair was generated in the past. It is important to note that it is not necessary to eavesdrop every possible combination of two tags in a group to compute every possible fingerprint. This is due once again to XOR involution property. For instance if the pairs P''_{T_A}, P''_{T_B} and P''_{T_A} and P''_{T_C} are intercepted then it is straightforward to compute $P''_{T_B} \oplus P''_{T_C}$. In general, for a group of N tags, instead of being necessary to eavesdrop combinations of N elements taken two at a time ($C_{N,2} = \binom{N}{2}$) it is enough to eavesdrop N different combinations. Although $Path^2_{T_i}$ are not revealed, traceability is compromised.

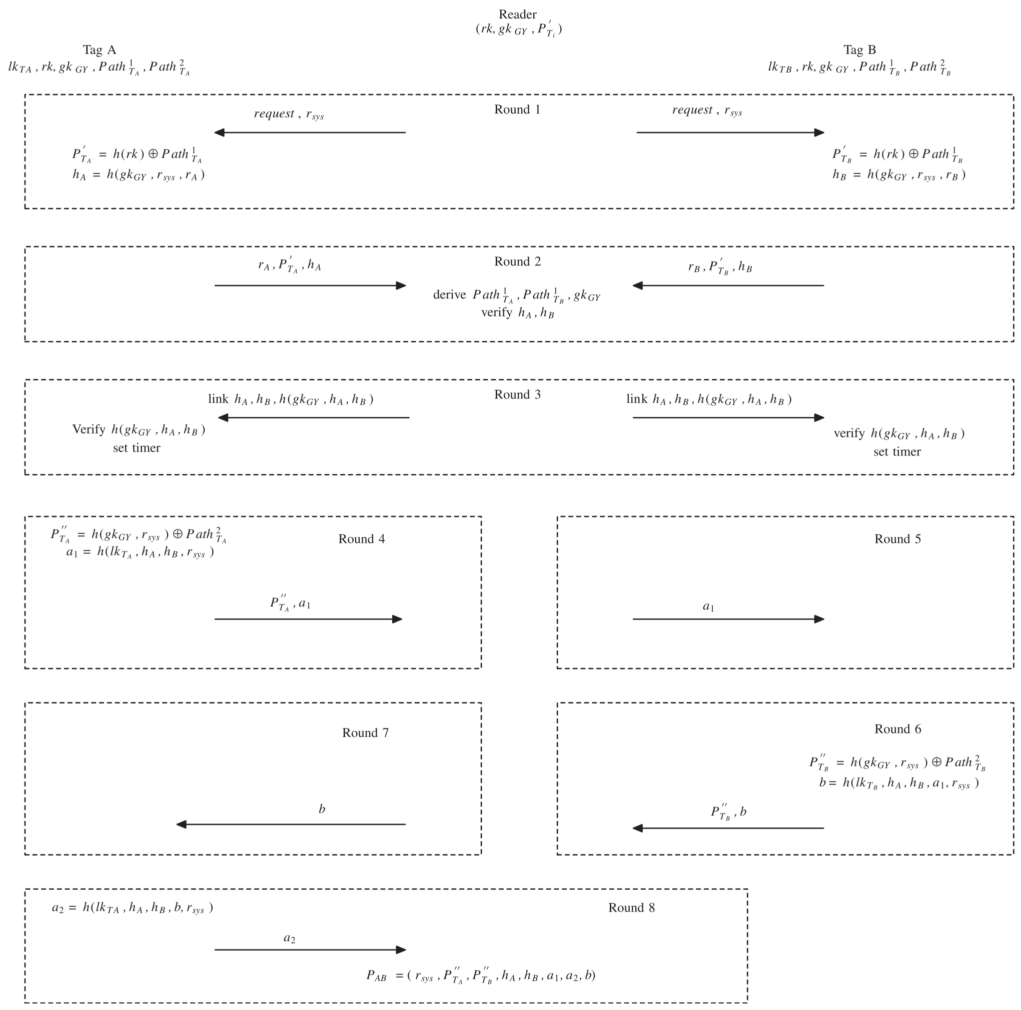
Reader
(rk, gk_{GY}, P'_{T_i})
Tag A
$lk_{TA}, rk, gk_{GY}, Path^1_{T_A}, Path^2_{T_A}$
Tag B
$lk_{TB}, rk, gk_{GY}, Path^1_{T_B}, Path^2_{T_B}$
Round 1
request, r_{sys}
$P'_{T_A} = h(rk) \oplus Path^1_{T_A}$
$h_A = h(gk_{GY}, r_{sys}, r_A)$
$P'_{T_B} = h(rk) \oplus Path^1_{T_B}$
$h_B = h(gk_{GY}, r_{sys}, r_B)$
Round 2
r_A, P'_{T_A}, h_A
r_B, P'_{T_B}, h_B
derive $Path^1_{T_A}, Path^1_{T_B}, gk_{GY}$
verify h_A, h_B
Round 3
link $h_A, h_B, h(gk_{GY}, h_A, h_B)$
Verify $h(gk_{GY}, h_A, h_B)$
set timer
verify $h(gk_{GY}, h_A, h_B)$
set timer
Round 4
$P''_{T_A} = h(gk_{GY}, r_{sys}) \oplus Path^2_{T_A}$
$a_1 = h(lk_{T_A}, h_A, h_B, r_{sys})$
P''_{T_A}, a_1
Round 5
a_1
Round 6
$P''_{T_B} = h(gk_{GY}, r_{sys}) \oplus Path^2_{T_B}$
$b = h(lk_{T_B}, h_A, h_B, a_1, r_{sys})$
P''_{T_B}, b
Round 7
b
Round 8
$a_2 = h(lk_{TA}, h_A, h_B, b, r_{sys})$
a_2
$P_{AB} = (r_{sys}, P''_{T_A}, P''_{T_B}, h_A, h_B, a_1, a_2, b)$

Fig. 3. Tree based yoking protocol.

It is also feasible an active attack to achieve traceability at tag level. Let us consider that a grouping-proof is generated for T_A and T_B and an attacker has eavesdropped the message r_{sys} and P''_{T_i}. Under these circumstances, the attacker (via a rogue reader) can replay r_{sys}. Then the attacker can eavesdrop a new P''_{T_i} and if its value matches with one previously captured, the corresponding tag is tracked. This situation happens because the messages exchanged in Rounds 4–7 do not include random numbers generated by the tags. Therefore, no freshness is provided.

Countermeasures: The traceability attacks are based on two main weaknesses of the protocol. First, the author assume that tags support on-chip a hash function and a PRNG function. Nevertheless, the protocol abuses of the XOR operator, which facilitates its cryptanalysis. Bitwise operations (e.g. XOR and AND operators) should be combined with non-triangular operations (e.g. bit rotation) to hinder the task to cryptoanalysts (see Guideline 1 in Section 7 for details). Second, as suggested by Guideline 3 in Section 7, tags should include fresh random numbers as one of inputs to the functions invoked. Furthermore, the composition of the input parameters — external and internal values — of these functions should prevent that an adversary may set arbitrarily its input value.

5. Forged proofs

A grouping-proof generates an evidence that two or more tags are scanned simultaneously. An attacker should not be able to impersonate one or more tags to generate a grouping-proof. However, some protocols are not carefully designed and leak private information in the messages transmitted over an insecure radio channel (see Guideline 1). In this section, we show how two very recent protocols (Huang and Ku, 2009; Chien et al., 2010) suffer from the aforementioned vulnerability. Both were designed for online mode.

5.1. *Huang and Ku (2009) online protocol for medication safety of inpatient*

Huang and Ku (2009) proposed an online grouping-proof compatible to Gen-2 standard (EPC Class-1 Generation-2, 2008; ISO/IEC 18006-C, 2005), which is one of the most relevant standards for low-cost RFID tags. Unlike previous proposals that use a message authentication code (MAC) and hash functions, operations supported on tags are limited to 16-bit pseudo-random number generation (PRNG), bitwise operations (e.g. exclusive OR operation), and cyclic redundancy check (CRC) function. Additionally, tags have two passwords of 32-bit each: (1) access password controls the access to the reserved memory; (2) kill password deactivates the tag upon its reception, being an irreversible operation.

The authors proposed a scheme to generate an evidence that {Tag_1, Tag_2, …, Tag_n, *Pallet Tag*} are scanned simultaneously (see Fig. 4). Reader is referred to the original paper for a detailed explanation of the analyzed protocol. We now focus on the

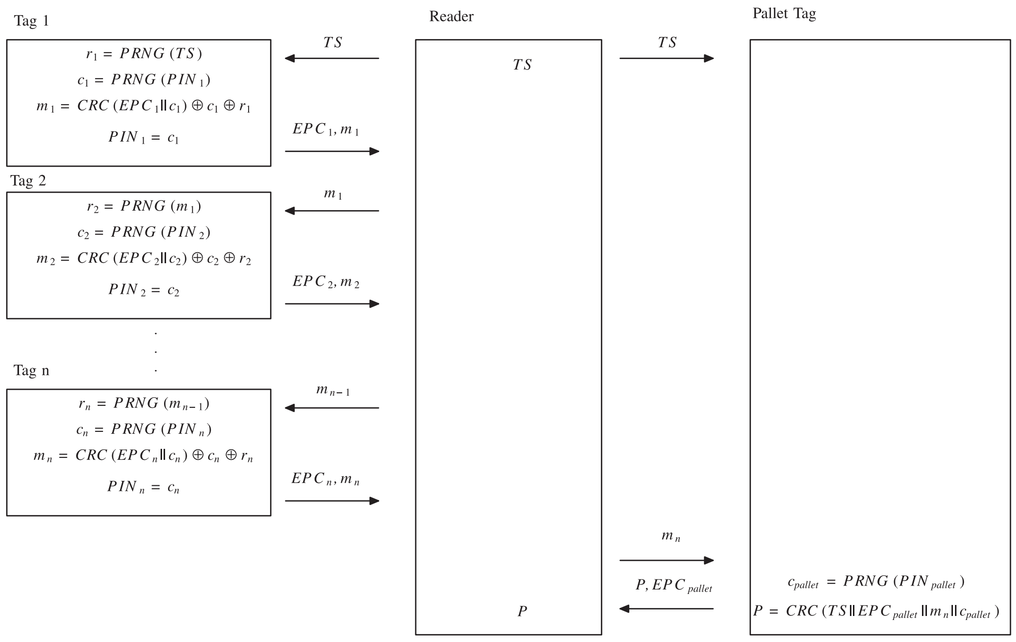

Fig. 4. Huang and Ku (2009) online protocol for medication safety of inpatient.

messages received/transmitted by one of the participating tags (e.g. Tag_i):

(1.0) The reader sends to Tag_i the authentication message m_{i-1} computed by Tag_{i-1}.

(2.0) Tag_i first inserts m_{i-1} and PIN_i into its PRNG to generate $r_i = PRNG(m_{i-1})$ and $c_i = PRNG(PIN_i)$, respectively. Second, the tag concatenates the electronic product code (EPC_i) and c_i and computes its *CRC*. Third, the bitwise XOR operation between the above result, c_i and r_i is calculated ($m_i = CRC(EPC \| c_i) \oplus c_i \oplus r_i$). Finally, the tuple $\{EPC_i, m_i\}$ is sent to the reader and the tag updates its secret password ($PIN_i = c_i$).

5.1.1. *Forgery attack on Huang and Ku (2009) protocol*

An attacker can obtain information of the messages passed over the channel due to CRC is not a secure hash function as assumed by the authors. CRC functions are based on polynomial arithmetic in F_2. Computing a CRC value for a given binary stream is essentially performed by dividing the polynomial associated with this stream by another fixed polynomial (generator polynomial) and obtaining a remainder. Due to the linearity, CRCs have the following properties (Peris-Lopez et al., 2009b; Han and Kwon, 2009):

$$CRC(A \oplus B) = CRC(A) \oplus CRC(B) \tag{1}$$

$$CRC(A \| B) = CRC(A \ll n) \oplus CRC(B) \tag{2}$$

where n is the bit-length of B.

An attacker can exploit the above properties to obtain private information linked to the target tag and impersonate this tag in a future grouping-proof. The attacker follows the phases described below.

Phase 1: *Acquiring private information*:

(1.0) The adversary sends to Tag_i a known a value.

(1.1) Tag_i first inserts a and PIN_i into its PRNG to generate $r_i = PRNG(a)$ and $c_i = PRNG(PIN_i)$, respectively. Second, the tag concatenates EPC_i and c_i and computes its *CRC*. Third, the bitwise XOR operation between the above result, c_i and r_i is calculated ($m_i = CRC(EPC \| c_i) \oplus c_i \oplus r_i$). Finally, the pair $\{EPC_i, m_i\}$ is sent to the adversary and the tag updates its password ($PIN_i = c_i$).

(1.2) The adversary is able to know r_i value as only a known seed a takes part in its generation. The static identifier of the tag EPC_i is transmitted in clear over channel and revealed to the adversary. Taking advantage of this knowledge and the properties of CRC functions, the adversary can disclose certain private information linked to the tag:

$$\begin{aligned} m_i &= CRC(EPC_i \| c_i) \oplus c_i \oplus r_i \\ &= CRC(EPC_i \ll n) \oplus CRC(c_i) \oplus c_i \oplus r_i \end{aligned} \tag{3}$$

Simplifying, the adversary obtains $CRC(c_i) \oplus c_i$, which is a value linked to the target tag univocally. According to Gen-2 standard c_i bit-length is 16. Therefore, $n = 16$ in Eq. (3). Reader should note that only public messages are used for this computation.

$$S_i = CRC(c_i) \oplus c_i = m_i \oplus CRC(EPC_i \ll 16) \oplus r_i \tag{4}$$

Phase 2. *Generation of a forged proof*:

(2.0) The legitimate reader sends to the adversary — impersonating Tag_i — the authentication message m'_{i-1} computed by Tag_{i-1}.

(2.1) The adversary inserts m'_{i-1} into its PRNG and generates $r'_i = PRNG(m'_{i-1})$. Then, the message authentication m'_i is computed by means of Eq. (4) and the EPC_i identification number, $m'_i = S_i \oplus r'_i \oplus CRC(EPC_i \ll 16)$.

Thus, the adversary can mislead the reader/verifier that Tag_i is involved in the proof when this tag is absent. It is important to note that S_i is closely related to PIN_i. The strength of this attack resides on when the attack is launched. In the protocol, the target tag updates its PIN just after the interrogation by the adversary. However, the updating is not performed by the verifier as this entity is not aware of the reading of the tag. That is, the reader and the tag have lost their synchronization after the reading of the tag by the adversary. This fact is very advantageous for an adversary. The adversary has thus at one's disposal an indefinite time window to impersonate the tags. After tag impersonation, the corresponding legitimate tag and the verifier are resynchronized, and the whole attack — phases 1 and 2 — would have to be repeated in order to supplant the legitimate tag again. So the updating of the secret information (PIN) seems appropriate as it reduces the consequence of leaking private information on the channel. However, the updating is performed even if there is no confirmation that interrogation comes from a legitimate reader. An adversary can exploit this weakness to conduct a very easy denial-of-service attack. If a fake request is sent to a tag, the tag and the verifier would be out of synchronization as a consequence of this simple attack. Additionally, Chien et al. (2010) show that Huang and Ku (2009) scheme is also vulnerable to replay attacks.

Countermeasures: CRC functions should be confined to detect errors transmission and secure cryptographic primitives should be used instead. On the other hand, when internal secret values are updated in the protocol, the scheme should support authentication and/or data integrity checking mechanisms (i.e. Chien, 2007, protocol). Furthermore, extra protection mechanisms (e.g. store the old and potential value of the updated variables) to combat desynchronization attacks are convenient.

5.2. *Chien et al. (2010) online protocol to enhance inpatient medication safety*

In an attempt to correct the security weakness of a previous scheme by Huang and Ku (2009), Chien et al. (2010) proposed two grouping-proofs protocols. First for online mode and the second for offline mode. The analysis of the online scheme is subject of this section.

The authors proposed an authentication protocol conforming to the standard EPC Class-1 Generation-2 (2008). The operations on the tags are limited to 16-bit PRNG and bitwise XOR operation. Additionally, the verifier and each tag share a secret PIN_i and tags also store an static identifier (EPC_i) into its memory. As the verifier (reader) is online, any RFID authentication protocol may be used. Specifically, the following scheme was proposed (see Fig. 5):

(0.0) The reader starts the timer.
(1.0) The reader generates a random number N_R as challenge to all the tags in its range.
The following procedure is repeated for all the tags:
(2.0) The tag generates a random number N_i, and computes a pseudo-random message authentication code:

$$MAC_i = PRNG(EPC_i \oplus PRNG(PIN_i) \oplus PRNG(N_R) \oplus PRNG(N_i)) \tag{5}$$

The tag sends to the reader the tuple $\{EPC_i, N_i, MAC_i\}$.
(2.1) The reader stops the timer and checks the correctness of MAC_i for each tag. If it passes, the tags $\{Tag_1, Tag_2, \ldots, Tag_n, Tag_{Pallet}\}$ are associated. Otherwise the protocol is aborted. Finally, the reader verifies that all the tags' answers are within a predefined time window.

5.2.1. *Forgery attack on Chien et al. (2010) protocol*

We now show how the above protocol results vulnerable to a passive attack. Basically, an adversary after eavesdropping several grouping-proof sessions can impersonate a target tag indefinitely. We focus our analysis on a specific Tag_i but it is straightforward to perform the attack in parallel for a set of tags. Suppose that the adversary is listening the messages exchanged between this tag and a legitimate reader. If the adversary detects that the random number generated by these two entities are equal (i.e. $N_R = N_i$), then she eavesdrops the corresponding MAC_i and stores it for future use. In such a case MAC_i value is independent of any random number:

$$\begin{aligned} S_i &= PRNG(EPC_i \oplus PRNG(PIN_i) \oplus PRNG(N_R) \oplus PRNG(N_i)) \\ &= PRNG(EPC_i \oplus PRNG(PIN_i) \oplus PRNG(N_R) \oplus PRNG(N_R)) \\ &= PRNG(EPC_i \oplus PRNG(PIN_i)) \end{aligned}$$

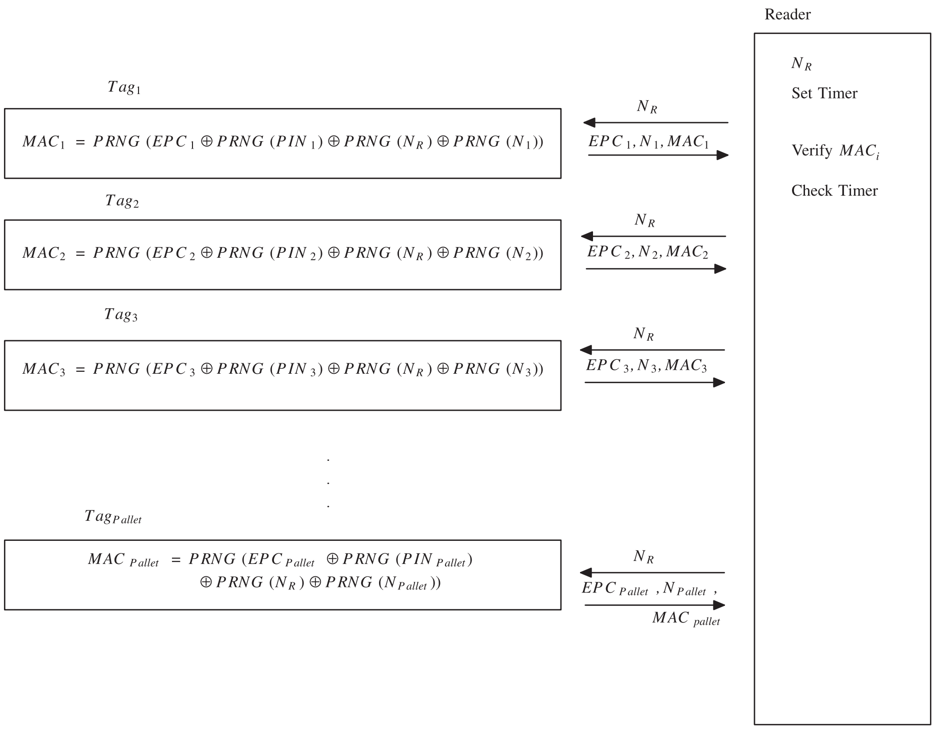
Reader
N_R
Set Timer
Verify MAC_i
Check Timer
Tag_1
$MAC_1 = PRNG(EPC_1 \oplus PRNG(PIN_1) \oplus PRNG(N_R) \oplus PRNG(N_1))$
N_R
EPC_1, N_1, MAC_1
Tag_2
$MAC_2 = PRNG(EPC_2 \oplus PRNG(PIN_2) \oplus PRNG(N_R) \oplus PRNG(N_2))$
N_R
EPC_2, N_2, MAC_2
Tag_3
$MAC_3 = PRNG(EPC_3 \oplus PRNG(PIN_3) \oplus PRNG(N_R) \oplus PRNG(N_3))$
N_R
EPC_3, N_3, MAC_3
Tag_{Pallet}
$MAC_{Pallet} = PRNG(EPC_{Pallet} \oplus PRNG(PIN_{Pallet}) \oplus PRNG(N_R) \oplus PRNG(N_{Pallet}))$
N_R
$EPC_{Pallet}, N_{Pallet}, MAC_{pallet}$

Fig. 5. Chien et al. (2010) online protocol to enhance inpatient medication safety.

As a consequence, the adversary can impersonate the target tag as described below:

(1.0) The reader generates a random number N'_R as challenge to the adversary.
(2.0) The adversary sends to the reader the tuple $\{EPC_i, N'_R, S_i\}$.

The adversary — impersonating Tag_i — is thus authenticated by the reader and the attack succeeds. The adversary exploits the linearity of bitwise operations to perform this attack. It is important to note that this attack can be launched at any time.

The remaining question is how many grouping-proof sessions have to be eavesdropped by the adversary to detect a session where $N_R = N_i$. Reader should note that random challenges are 16-bit length as required by Gen-2 standard. Therefore, and due to the birthday paradox (Blomm, 1973), the adversary has to eavesdrop approximately $\sqrt{(\pi/2)2^{16}} \simeq 286$ sessions to find a collision. In summary, the adversary needs to eavesdrop a small number of sessions to impersonate the target tag and generate a forged grouping-proof.

6. Subset replay attacks

In this section we describe a new type of replay attack on grouping-proofs — compared to those put forward against (Juels, 2004; Saito and Sakurai, 2005) — that allows the generation of fake proofs. Specifically, a rogue reader is able to generate a proof that links a subset of simultaneously read legitimate tags to any other legitimate tag. Chien et al. (2010) and Saito et al. (2004) are protocols that fall into this flaw. Guideline 2 discusses this problem. Following we explain the attack focusing on Chien et al. (2010). First we begin with the protocol description, then we explain the attack.

6.1. Chien et al. (2010) offline protocol to enhance inpatient medication safety

This protocol is focused on proving that specific group of drugs are indeed given to specific inpatients. Accordingly, the offline verifier knows in advance what drugs correspond to each inpatient (i.e. the prescription). Thus, each drug is associated with a tag and a special tag refers to inpatient (*Pallets*). In their notation, EPC_i is the tag identifier of Tag_i and EPC_{Pallet} is the tag identifier of the *Pallet*. The steps of the protocol, represented in Fig. 6, are:

(1) $Verifier \rightarrow Reader: t = E_{K_V}(timestamp)$
First, the reader gets an encrypted timestamp $t = E_{K_V}(timestamp)$ from the verifier, where $E_{K_V}(timestamp)$ denotes an encryption of the current timestamp using verifier's secret key K_V.
(2) $Reader \rightarrow Tag_1, Pallet: t$
The reader sends the encrypted timestamp to Tag_1 and *Pallet*.
(3) For $i = 1, \ldots, n-1$
 (3.1) $Tag_i \rightarrow Reader: EPC_i, m_i$ If $i = 1$, then let $m_0 = t$; Tag_i computes $m_i = PRNG(EPC_i \oplus PRNG(m_{i-1}) \oplus PRNG(PIN_i))$ and sends it to the reader with EPC_i.
 (3.2) $Reader \rightarrow Tag_{i+1}: m_i$
 The reader forwards m_i to the next tag Tag_{i+1}.
(4) Tag_n and *Pallet*
 (4.1) $Tag_n \rightarrow Reader: EPC_n, m_n$
 Tag_n computes $m_n = PRNG\ (EPC_n \oplus PRNG(m_{n-1}) \oplus PRNG(PIN_n))$ and sends it to the reader with EPC_n.
 (4.2) $Reader \rightarrow Pallet: m_n$
 The reader forwards m_n to Pallet.
 (4.3) $Pallet \rightarrow Reader: EPC_{Pallet}, m_{Pallet}$
 Upon receiving m_n, *Pallet* computes $m_{Pallet} = PRNG(EPC_{Pallet} \oplus PRNG(m_n) \oplus PRNG(PIN_{Pallet}))$ and sends both EPC_{Pallet} and m_{Pallet} to the reader.
(5) $Reader \rightarrow Verifier: (t, EPC_1, m_1, \ldots, EPC_n, m_n, EPC_{Pallet}, m_{Pallet})$
Reader collects the evidence $(t, EPC_1, m_1, \ldots, EPC_n, m_n, EPC_{Pallet}, m_{Pallet})$ and forwards it to the verifier.
(6) The verifier checks:
 (6.1) Whether the association $(EPC_1, \ldots, EPC_n, EPC_{Pallet})$ holds for the prescription.
 (6.2) Whether the evidence $(m_1, \ldots, m_n, m_{Pallet})$ holds.
 (6.3) That the decrypted timestamp $D_{K_V}(t)$ is within a reasonable time span. If so, the grouping-proof succeeds.

6.1.1. Subset replay attacks on Chien et al. (2010) offline protocol

Chien et al. (2010) protocol assumes that the verifier knows in advance what the prescription for each inpatient is. Let us suppose that the prescription of inpatient *A* is a subset of the prescription of inpatient *B*. In these conditions, it is possible to

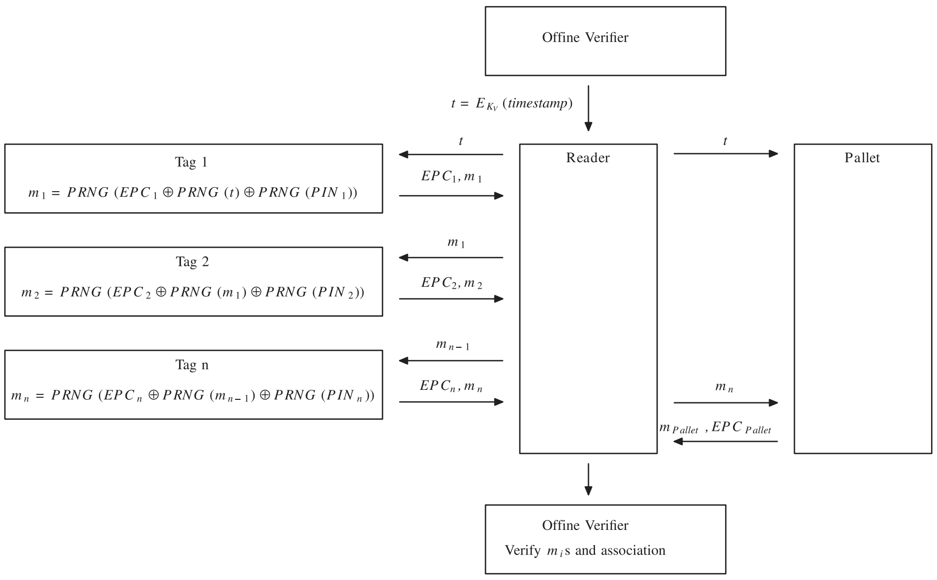
Offline Verifier
$t = E_{K_V}(timestamp)$
Tag 1
$m_1 = PRNG(EPC_1 \oplus PRNG(t) \oplus PRNG(PIN_1))$
Tag 2
$m_2 = PRNG(EPC_2 \oplus PRNG(m_1) \oplus PRNG(PIN_2))$
Tag n
$m_n = PRNG(EPC_n \oplus PRNG(m_{n-1}) \oplus PRNG(PIN_n))$
t
EPC_1, m_1
m_1
EPC_2, m_2
m_{n-1}
EPC_n, m_n
Reader
t
m_n
m_{Pallet}, EPC_{Pallet}
Pallet
Offline Verifier
Verify m_is and association

Fig. 6. Chien et al. (2010) offline protocol to enhance inpatient medication safety.

generate a proof that inpatient *A* has received her prescription just eavesdropping the messages exchanged when generating the proof for inpatient *B*. For instance, let inpatient *A* ($Pallet_A$) prescription be ibuprofen (Tag_1) and paracetamol (Tag_2) and let inpatient *B* ($Pallet_B$) prescription be ibuprofen (Tag_1), paracetamol (Tag_2) and morphine (Tag_3). Once the messages corresponding to the grouping-proof for inpatient *B* (t, EPC_1, m_1, EPC_2, m_2, EPC_{Pallet_B}, EPC_3, m_3, m_{Pallet_B}) have been eavesdropped, a rogue reader can replay m_2 to $Pallet_A$ and generate the corresponding fake proof with the response (t, EPC_1, m_1, EPC_2, m_2, EPC_{Pallet_A}, m_{Pallet_A}).

In addition, it is important to note that Chien et al. (2010) grouping-proofs could not be applied to other scenarios such as a supply chain. In such scenario the number and type of tags are not known in advance. Therefore, it is straightforward according to the previous analysis how a rogue reader can generate a grouping-proof stating that just a subset of legitimate tags were on any pallet (when for instance the complete set was on a specific one). To this end, it is necessary that an adversary eavesdrops the channel to capture t, EPC_i and the corresponding m_i (i from 1 to x with $x \leq n$). Then, the adversary via a rogue reader forwards m_x to any pallet and uses the response ($m_{anyPallet}$) to generate a fake proof. In such a case, the proof (t, EPC_1, m_1,..., EPC_x, m_x, $EPC_{anyPallet}$, $m_{anyPallet}$) would state that x legitimate tags were simultaneously read with *anyPallet*.

The implications of the aforementioned attack are determined by the usefulness of using the subsets of a sequence of tags. That is, the adversary cannot select a sequence of tags but she can associated a subset — in the same positions — of this sequence with a different inpatient (*anyPallet*). On the other hand, the verifier does not set in advance a correspondence between an encrypted timestamp t and a specific sequence of tags. So the adversary can eavesdrop the messages associated with a sequence of tags — including the t value — and then she can use these values to generate an evidence for a subset of these tags.

In this context, Saito et al. (2004) protocol is also vulnerable to this kind of attack but in a more severe way. It is possible to generate a grouping proof stating that any subset of legitimate tags (not necessarily ordered now) were simultaneously read with any *Pallet*. We could think that Lien et al. (2008) suffers from a similar problem but the Chaining-Proof they propose is not a real offline protocol but online. Thus, they assume the reader is connected to a trusted security database (TSD) that can be consulted in each phase of the protocol.

Countermeasure: The computation of a Tag_i should be dependant on the values computed by its predecessors {Tag_1,..., Tag_{i-2}, Tag_{i-1}} instead of being exclusively dependant on a value generated by Tag_{i-1}. The reader is urged to consult Guideline 2 in Section 7 for details.

Finally, Table 2 summarizes up to date attacks on state-of-the-art grouping-proofs. First four columns represent the attacks described in this paper.

7. Guidelines for securing RFID yoking/grouping-proofs

In previous sections, we show several attacks to recent schemes that fail in their attempt to design a secure yoking/grouping-proof. Indeed, every protocol published to date result vulnerable to attacks of major or minor relevance. We now provide a list of practical tips that should be followed by protocol designer to preclude past errors:

(1) *Computing capabilities*: RFID tags are devices which a computing power severely restricted. The designer of a grouping-proof should be aware of this condition. Tags should use just pseudo-random authentication messages for tags conforming to EPC Class-1 Generation-2 or message authentication codes (MACs) for more powerful tags.
Low-cost RFID tags conforming to Gen-2 use cover-coding to obscure passwords and information written to or read from a tag. This technique is only useful when adversaries are expected to be at greater distance from the tags than readers. The 16-bit PRNG supported on-board by Gen-2 tags may be used to build a kind of pseudo-random authentication message in that cases in which the eavesdropper can capture messages transmitted on the backward and forward channels. In this context, bitwise operations generally appear to make up certain messages because these operators can be easily implemented in hardware. However, some protocols abuse of the usage of these functions which facilitates their cryptanalysis. An attacker may exploit their vulnerability against active attacks (Alomair and Poovendran, 2008) and even linear cryptanalysis techniques, as a consequence of being triangular-functions (Klimov and Shamir, 2005). Note that a combination of triangular functions results in another triangular function being still possible a simple analysis. A triangular-function has the property that output bits only depend on the leftmost input bits, instead of every input bit. This undesirable characteristic (lack of diffusion) greatly facilitates the analysis of the messages. We recom-

Table 2
Summary of up to date attacks on yoking/grouping-proofs.

	Traceability	Impersonation	Forge proof	Subset replay	Anonymity	Replay	Multi-proof (Peris-Lopez et al., 2007)	Useless proofs (DoS) (Burmester et al., 2008)
Juels (2004)	×	×	–	–	×	×	–	×
Saito and Sakurai (2005)	–	×	–	×	–	×	–	×
Bolotnyy and Robins (2006)	–	–	–	×	–	–	×	×
Piramuthu (2006)	×	–	–	–	×	–	×	×
Peris-Lopez et al. (2007)	–	–	–	–	–	–	–	×
Cho et al. (2008)	×	–	–	–	×	–	×	×
Lien et al. (2008)	×	–	–	–	×	–	–	×
Burmester et al. (2008)	–	×	–	–	–	–	–	–
Chien and Liu (2009)	×	–	–	–	–	–	–	–
Huang and Ku (2009)	×	–	×	–	×	–	–	×
Chien et al. (2010)	×	–	×	–	×	–	–	×
Lin et al. (2007)[a]	×	×	–	–	×	–	–	×
Chien et al. (2010)[a]	×	–	–	×	×	–	–	×

[a] Offline version.

mend the combination of triangular (e.g. bitwise exclusive-OR operation) and non-triangular-functions (e.g. circular shift or bit rotation) for the composition of the messages or the input parameters of the pseudo-random functions.

Apart from tags conforming Gen-2, it is commonly assumed that tags are able to compute a MAC, like a HMAC function. The verifier and the tag share a common secret key that can be used in combination with a hash algorithm to provide one-way or mutual authentication between these two entities. The integrity of data in messages is also ensured. Furthermore, tags should include fresh random numbers as one of the inputs of MAC function to prune the probability of launching a replay attack successfully.

(2) *Dependence*: Every input to a given tag (except the first one) should be derived from computations that can only be carried out by fellow tags participating in the proof. This guarantees the causality of the tokens generated during the proof. Furthermore, the proof should be ended in a time window defined by the verifier. When just two tags are involved in the proof, the causality and the completion of the proof in a limited time window certify that they have been scanned simultaneously. On the contrary, if the proof is made up of three or more tags, causality only guarantees the sequential scanning of the tags but simultaneity cannot be verified. To prove simultaneity, the input of a Tag_i should be a combination of the values computed by its predecessors $\{Tag_1, \ldots, Tag_{i-2}, Tag_{i-1}\}$ instead of being exclusively dependant on a value generated by Tag_{i-1}. The requirement of completing the proof in a time window defined by the verifier is also demanded in these cases.

(3) *Identification*: Privacy protection is one of the fundamental concerns linked with the deployment of RFID systems. Nevertheless, in the vast majority of existing proposals, tags transmit their static identifier uncovered over the insecure radio channel. Thus, a simple eavesdropper can capture these values, compromising the privacy of tags' holder. An encrypted version of the identifier may be used to protect privacy, but traceability is still at risk because the answer provided by each tag is a constant value. Therefore, static identifiers should be disguised (anonymized) and different (non-traceable) at each protocol execution. The inclusion of random numbers looks convenient to build privacy-protected identifiers but by itself does not guarantee that the proposed protocol is immune to privacy attacks (Peris-Lopez et al., 2009a).

(4) *Matching*: Burmester et al. (2008) introduced the interesting idea of using an identifier (ID_{group}) and a key (K_{group}) to prove membership to a group. At the start of the proof, tags can check each other's computation and make sure that only tags in the group participate in the proof. If verification success the proof continues; otherwise it is aborted. If tags do not verify their belonging to the group, unrelated tags can be participants of a proof, and the failure would only be detected when the proof is sent to the verifier. We recommend the use of groups to avoid wasting resources by filtering out useless proofs. Certain applications may require greater accuracy in the identification process and groups could be divided into subgroups.

(5) *Verification*: Saito and Sakurai (2005) proposed the inclusion of timestamps to thwart replay attacks. Nevertheless, replay attacks are not prevented as future proofs can be generated in advance due to timestamps — transmitted in clear — are predictable values (Piramuthu, 2006). We suggest the usage of an encrypted version of the timestamps. The verifier computes this protected timestamps, using its long-term secret key (i.e. $t = E_{K_V}(Timestamp_i)$). Finally, she stores these values and also annotates a time window (Δ_i) in which each of these values is valid. Hundreds of these values may be computed in advance by the verifier and sent out to the reader at regular times.

6. *Performance*: Inspired by Chien and Liu (2009), Table 3 shows a performance comparison of every grouping-proofs proposed up to date. The performance is evaluated according four main criteria. First, the utilization of the radio channel is quantified by counting the number of messages and packets transmitted on the channel. Second, we estimate the cost of the whole computations performed by one — the worst case — of the tags participating in the proof. More precisely, we sum the number of invocations to a hash function, a PRNG function and bitwise operators by each tag involved in the proof. The maximum of the values obtained is finally annotated in the table. Third, as tags are the most restricted devices, we count the number of variables stored in their memory. Finally, we show the computation cost of identifying a tag in the back-end database.

The number of computations as well as the messages transmitted on the channel by tags should be minimized, without compromising the security of the scheme proposed. According to memory requirements, protocol designers

Table 3
Performance comparison of grouping proofs.

	♯ Messages	♯ Packets	♯ Hashing operation	♯ Random number generation	♯ Bitwise invocation	Storage of the tag	Cost of computing of back-end server
Juels (2004)	6	9	1	1	–	2	$O(N)$
Saito and Sakurai (2005)	5	5	1	–	–	2	$O(N^2)$
Bolotnyy and Robins (2006)[a]	5	10	2	–	–	3	$O(N)$
Piramuthu (2006)	6	12	2	1	–	2	$O(N)$
Peris-Lopez et al. (2007)	6	9	2	–	1[b]	3	$O(N)$
Cho et al. (2008)	6	7	2	1	–	2	$O(N)$
Lien et al. (2008)[a]	8	18	2	1	1	2	$O(N)$
Burmester et al. (2008)[c]	11	14	2	–	–	4	$O(M)$
Chien and Liu (2009)	11	23	5	1	2	5	$O(1)$
Huang and Ku (2009)[a]	5	8	–	3	3	1	$O(N^2)$
Lin et al. (2007)[d]	4	9	1	–	–	2	$O(N^2)$
Chien et al. (2010)[da]	5	7	–	3	2	2	$O(1)$[e]

[a] We consider that the proof only involves two tags.
[b] Invocation of *Nun* function.
[c] Robust grouping proof, where *M* symbolizes the number of tags in the group.
[d] Offline version.
[e] Privacy is not a concern issue: identifiers are passed in clear over the channel and are included in the proof.

should limit the use of non-volatile tags' memory in which identifiers and secret keys are stored. Furthermore, the length of variables used in a proposed scheme should be carefully checked when these are combined. Sometimes this simple point is overlooked as in a recent proposal that claims the Gen-2 specification conformance (Chien et al., 2010).

7. *Forward security*: An attacker may disclose secret keys stored on tag's memory as these devices are susceptible to physical manipulation. In certain scenarios, privacy of past communications has to be guaranteed even if a tag is compromised sometime later. This property is commonly named as forward security. Indeed, there are some RFID authentication protocols that claim to satisfy this property (e.g. Conti et al., 2007; Le et al., 2007). Nevertheless, the design of RFID offline grouping protocol with forward security is an open problem. The protocol designers have to sort out two important restrictions: (1) the verifier is offline — in the interesting case — in contrast to the online verifier in authentication protocol; (2) tags and verifier have to keep on their synchronization state. In a RFID authentication protocol, a tag and a reader (verifier) are the only entities involved. However, in a grouping-proof several tags and a (untrusted) reader participate in proof generation and the verifier takes part sometime later. In addition, only a subgroup of tags in the group may be involved in a proof, which makes difficult the updating of the key in the group.

8. Kazahaya: an RFID yoking proof for low-cost RFID tags

In this section, we present an RFID yoking proof for tags conforming to the Gen-2 standard and the Guidelines defined in the previous section. Tags operations are limited to the invocation of a PRNG function and the bitwise XOR operator as dictated by Guideline 1. The number of invocations, as recommended by Guideline 6, is minimized but the security of our proposed protocol is not put at risk.

Tags are divided into groups, which are identified by a group identifier ID_{group}. By using this technique, we prevent the participation of unrelated tags in the proof (Guideline 4). Tags have an unique identifier ID_{T_i} and store two private keys. A group key K_{group} proves its membership to a specific group, and a secret key K_{T_i} facilitates the authentication of tags. For each tag, the back-end database stores the tuple $\{ID_{T_i}, ID_{group}, K_{T_i}, K_{group}\}$.

Following Guideline 3, the static identifiers $\{ID_{T_i}, ID_{group}\}$ of a tag are never sent in clear on the channel to guarantee privacy protection. Furthermore, in the generation of messages transmitted on the radio channel by each participating tag, two random numbers $\{r_{T_i}, r'_{T_i}\}$ are used in each session to prevent the traceability of tags' answer. The composition of messages has been analyzed in depth to avoid straightforward attacks such as those that exploit the involution property of the XOR operator.

In the initialization phase, the verifier computes encrypted timestamps $t_n = E_{K_V}(Timestamp_n)$, where K_V represents the secret key of the verifier. Each of these values are not valid indefinitely but during a limited time window Δ_n (Guideline 5). Finally, the verifier stores the tuples $\{t_n, \Delta_n\}$ computed. The protocol starts once the reader receives an encrypted timestamp t_n.

For tag_A and tag_A belonging to the same group, the messages exchanged are described below. Basically, the computations of each tag — except the first one — derive from results previously computed by the other tag participating in the proof. By this condition and assuming that the proof is delivered to the verifier in the defined time window (Guideline 2), we can assert that the two tags have been read simultaneously.

Step 1: The reader queries Tag_A by sending the timestamp $\{t_n\}$.

Step 2: The Tag_A generates two random number $\{r_{T_A}, r'_{T_A}\}$ and computes:

$$M^1_{group} = PRNG(ID_{group} \oplus r_{T_A} \oplus PRNG(K_{group}) \oplus PRNG(t_n))$$

$$M_{T_A} = PRNG(ID_{T_A} \oplus r'_{T_A} \oplus PRNG(K_{T_A}) \oplus PRNG(t_n+1))$$

The tag sends $\{r_{T_A}, r'_{T_A}, M^1_{group}, M_{T_A}\}$ to the reader.

Step 3: The reader stores r'_{T_A} and submits $\{t_n, r_{T_A}, M^1_{group}, M_{T_A}\}$ to the tag_B.

Step 4: The tag_B checks if the other participant in the proof belongs to the same group. More precisely, she computes a local version of M^1_{group} using its internal values:

$$M^{1*}_{group} = PRNG(ID_{group} \oplus r_{T_A} \oplus PRNG(K_{group}) \oplus PRNG(t_i))$$

If $M^{1*}_{group} \overset{?}{=} M^1_{group}$, the Tag_B generates two random number $\{r_{T_B}, r'_{T_B}\}$ and computes:

$$M^2_{group} = PRNG(ID_{group} \oplus r_{T_B} \oplus PRNG(K_{group}) \oplus PRNG(M^1_{group}))$$

$$M_{T_B} = PRNG(ID_{T_B} \oplus r'_{T_B} \oplus PRNG(K_{T_B}) \oplus PRNG(M_{T_A}))$$

The tag sends $\{r_{T_B}, r'_{T_B}, M^2_{group}, M_{T_B}\}$ to the reader.

Step 5: The reader stores r'_{T_B} and submits $\{r_{T_B}, M^2_{group}, M_{T_B}\}$ to the tag_A.

Step 6: The tag_A checks if the other participant belongs to the same group. More precisely, she computes a local version of M^2_{group} using its internal values:

$$M^{2*}_{group} = PRNG(ID_{group} \oplus r_{T_B} \oplus PRNG(K_{group}) \oplus PRNG(M^1_{group}))$$

If $M^{2*}_{group} \overset{?}{=} M^2_{group}$, the Tag_A computes the final message and sends the result to the reader:

$$M_{T_{AB}} = PRNG(ID_{T_A} \oplus M_{T_A} \oplus PRNG(M_{T_B}) \oplus PRNG(K_{T_A}+1))$$

Step 7: The reader generates the evidence, $e^{T_{AB}}_n = \{ID_{T_A}, ID_{T_B}, t_n, r'_{T_A}, r'_{T_B}, M_{T_{AB}}\}$ and submits it to the verifier.

9. Conclusions

Since the introduction of the concept of yoking-proofs by Juels (2004), more than a dozen of new schemes have been proposed. In general, the methodology followed by authors has been, first, the identification of security vulnerabilities in a specific protocol and, then, the proposal of an enhancement scheme that claims to be immune to the flaws of its predecessor. Nevertheless, we analyze the security of grouping-proofs from a global perspective. Only after a complete revision and analysis of the literature we identify the guidelines that should be followed by protocol designers. Tables 2 and 3 give the reader a quick and all-embracing overview of the state-of-the-art regarding to performance and security properties. From a security perspective, Peris-Lopez et al. (2007) and Chien and Liu (2009) are the most secure schemes but certain flaws are still present. Efficiency is superior on the first scheme but the complexity in the back-end database is lower in the second one. Other interesting schemes are (Burmester et al., 2008; Lien et al., 2008) which introduces the great idea of using groups to avoid the generation of useless proofs. In summary, the design of secure and efficient RFID grouping-proofs is not closed yet. The Kazahaya protocol represents a step further to achieve this objective. Nevertheless, nowadays the challenge of including forward-security on RFID grouping-proofs is an open problem in RFID offline grouping-proofs.

References

18006-C ISO/IEC. Information technology — radio frequency identification for item management — part 6: parameters for air interface communications at 860 MHz to 960 MHz, 2005 〈http://www.iso.org〉.

Alomair B, Poovendran R. On the authentication of RFID systems with bitwise operations. In: Proceedings of the second IFIP conference on new technologies, mobility and security — NTMS'08, 2008. p. 1–6.

Blomm D. A birthday problem. American Mathematical Monthly 1973;80:1141–2.

Bolotnyy L, Robins G. Generalized ''yoking-proofs'' for a group of RFID tags. In: International conference on mobile and ubiquitous systems: networking and services, MobiQuitous. San Jose, CA, USA: IEEE Computer Society; July 2006. p. 1–4.

Burmester M, de Medeiros B, Motta R. Provably secure grouping-proofs for RFID Tags. In: Proceeding of the 8th smart card research and advanced applications — CARDIS 2008. Lecture notes in computer science. UK: Springer, Royal Holloway University of London; September 2008.

Chien H-Y. SASI: a new ultralightweight RFID authentication protocol providing strong authentication and strong integrity. IEEE Transactions on Dependable and Secure Computing 2007;4(4):337–40.

Chien H-Y, Liu S-B. Tree-based RFID yoking proof. In: International conference on networks security, wireless communications and trusted computing — NSWCTC 09. Wuhan, China: IEEE Computer Society; April 2009. p. 550–3.

Chien H-Y, Yang C-C, Wu T-C, Lee C-F. Two RFID-based solutions to enhance inpatient medication safety. Journal of Medical Systems 2010. 〈http://www.springerlink.com/content/x2n7x062637g6k74/〉.

Cho J-S, Yeo S-S, Hwang S, Rhee S-Y, Kim SK. Enhanced yoking proof protocols for RFID tags and tag groups. In: International conference on advanced information networking and applications — workshops — AINAW 2008. Okinawa, Japan: IEEE Computer Society; March 2008. p. 1591–6.

Conti M, Di Pietro R, Mancini LV, Spognardi A. FastRIPP: RFID privacy preserving protocol with forward secrecy and fast resynchronization. In: 33th annual conference of the IEEE industrial electronics society (IEEE IECON 07), Taipei, Taiwan, November 2007. p. 52–57.

Generation-2 EPCGlobal, 2008. Class-1 Generation 2 UHF Air Interface Protocol Standard Version 1.2.0: ''Gen-2'' 〈http://www.epcglobalinc.org/standards/〉.

Han D, Kwon D. Vulnerability of an rfid authentication protocol conforming to epc class 1 generation 2 standards. Computer Standards & Interfaces 2009;31(4): 648–52.

Huang H-H, Ku C-Y. A RFID grouping proof protocol for medication safety of inpatient. Journal of Medical Systems 2009;33(6):467–74.

Juels A. ''Yoking-proofs'' for RFID tags. In: Sandhu R, Thomas R, editors. International Workshop on Pervasive Computing and Communication Security — PerSec 2004. Orlando, FL, USA: IEEE, IEEE Computer Society; March 2004. p. 138–43.

Juels A. RFID security and privacy: a research survey. IEEE Journal on Selected Areas in Communications 2006;24(2):381–94.

Klimov A, Shamir A. New applications of t-functions in block ciphers and hash functions. In: Proceedings of 12th international workshop, fast software encryption. Lecture notes in computer science, vol. 3557. ENSTA, Paris, France, Springer: 2005; p. 18–31.

Le Tv, Burmester M, Medeiros Bd. Forward-secure RFID authentication and key exchange. Cryptology ePrint Archive. Report 2007/051; 2007.

Leng X, Lien Y, Mayes K, Markantonakis K, Chiu J-H. Select-response grouping proof for RFID tags. Dong Hoi City, Vietnam: IEEE Computer Society; April 2009. p. 73–77.

Lien Y, Leng X, Mayes K, Chiu J-H. Reading order independent grouping proof for RFID tags. In: IEEE international conference on intelligence and security informatics, ISI 2008. Taipei, Taiwan: IEEE; June 2008. p. 128–36.

Lin C-C, Lai Y-C, Tygar JD, Yang C-K, Chiang C-L. Coexistence proof using chain of timestamps for multiple RFID tags. In: Chang KC-C, Wang W, Chen L, Ellis CA, Hsu C-H, Tsoi AC, Wang H, editors. International workshop on database management and application over networks — DBMAN 2007. Lecture notes in computer science, vol. 4537. Huang Shan, China: Springer-Verlag; June 2007. p. 634–43.

Peris-Lopez P, Hernandez-Castro JC, Estevez-Tapiador JM, Ribagorda A. Solving the simultaneous scanning problem anonymously: clumping proofs for RFID tags. In: IEEE international conference on pervasive services, workshop on security, privacy and trust in pervasive and ubiquitous computing— SecPerU 2007. Istanbul, Turkey: IEEE, IEEE Computer Society Press; July 2007. p. 55–60.

Peris-Lopez P, Hernandez-Castro JC, Estevez-Tapiador JM, Li T, van der Lubbe JC. Weaknesses in two recent lightweight RFID authentication Protocols. In: Workshop on RFID Security — RFIDSec'09. Leuven, Belgium, July 2009a.

Peris-Lopez P, Hernandez-Castro JC, Estevez-Tapiador JM, Ribagorda A. Cryptanalysis of a novel authentication protocol conforming to epc-c1g2 standard. Computer Standards & Interfaces 2009b;31(2):372–80.

Piramuthu S. 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: IEEE, IEEE Computer Society Press; June 2006.

Saito J, Ryou J-C, Sakurai K. Enhancing privacy of universal Re-encryption scheme for RFID tags. In: Jang L, Guo M, Gao G, Jha N, editors. Embedded and ubiquitous computing — EUC 2004. Lecture notes in computer science, vol. 3207. Aizu-Wakamatsu City, Japan: Springer-Verlag; August 2004. p. 879–90.

Saito J, Sakurai K. Grouping proof for RFID tags. In: Conference on advanced information networking and applications — AINA, vol. 2. Taiwan: IEEE; March 2005. p. 621–4.

van Deursen T, Radomirović S. Attacks on RFID protocols. Cryptology ePrint Archive. Report 2008/310; July 2008.