SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2014; **7**:2669–2683
Published online 9 December 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.400

SPECIAL ISSUE PAPER

How to protect security and privacy in the IoT: a policy-based RFID tag management protocol

Evangelos Rekleitis*, Panagiotis Rizomiliotis and Stefanos Gritzalis

Department of Information and Communication Systems Engineering, University of the Aegean, Karlovassi, Samos 83200, Greece

ABSTRACT

Radio-frequency identification (RFID) technology constitutes an important part of what has become known as the Internet of Things (IoT) that is accessible and interconnected machines and everyday objects that form a dynamic and complex environment. To secure the IoT in a cost-efficient manner, we need to build security and privacy into the design of its components. Moreover, mechanisms should be constructed that will allow both individuals and organizations to actively manage their "things" and information in a highly flux environment. The contributions of this paper are twofold: We first discuss the use of security and privacy policies that can offer fine granularity and context-aware information control in RFID systems. Second, we propose a novel secure and privacy-preserving tag management protocol that can support such policies. Our protocol has a modular design that allows it to support a set of desirable management operations (viz. tag authentication, delegation, and ownership transfer) while imposing minimal hardware and computational requirements on the tag side. Furthermore, inspired by the European Network and Information Security Agency's Flying 2.0 study, we describe a near-future air travel scenario to further explain and demonstrate the inner workings of our proposal. Copyright © 2011 John Wiley & Sons, Ltd.

KEYWORDS
IoT; RFID; policy; security; authentication; delegation

***Correspondence**
Evangelos Rekleitis, Department of Information and Communication Systems Engineering, University of the Aegean, Karlovassi, Samos 83200, Greece.
E-mail: erekl@aegean.gr

1. INTRODUCTION

The term Internet of Things (IoT) describes a vision of a tighter integration between the physical and the virtual world. Building on the rapid growth of the Internet, visionaries imagine a world where everyday objects (things) and machines will be interconnected and networked, revolutionizing our way of life. The increase of available data and the emerging new ways of interacting with and managing everyday objects will bring an unrepresented level of automation.

The IoT is expected to form a dynamic and complex environment, consisting of some billions of networked and interrelated things and machines. This vision raises many security and privacy concerns because today's tools and techniques might prove inadequate to ensure a safe IoT. This comes at no surprise, considering the difficulty at which we provide security and privacy in current systems. It is therefore crucial that IoT components are designed from their inception with a privacy-by-design and security-by-design mindset and comprehensively include user requirements [1]. Already, several technologies exist that are to become the basis of the IoT, such as Internet Protocol version 6 (IPv6), Web services, Service-Oriented Architecture (SOA), and radio-frequency identification (RFID). Especially regarding RFID, which is the focus of this research, there is an on-going effort to provide a viable secure and privacy-respecting system. More precisely, several protocols have been proposed aiming to provide secure tag management operations, such as tag authentication and ownership transfer. However, most of these proposals offer stand-alone security services and do not consider the security and privacy of the tag in a unified way.

In this paper, we propose mechanisms that can achieve usable security and privacy in an RFID system. First, we discuss the application of security and privacy policies to provide both fine-grain access control to the tags' information, stored in a back-end system, and to control tag operations. Furthermore, we introduce a secure authentication and management protocol for low-cost RFIDs. The protocol has a modular design supporting crucial operations, such as tag authentication, secure delegation, and privacy-respecting ownership transfer, and in addition, it implements all the operations in a uniform way, offering significant implementation efficiency. Moreover, the proposed protocol can be coupled with the said policies to offer both security and practical privacy management to the end users. Finally, inspired by the European Network and Information Security Agency's (ENISA) Flying 2.0 study [2], we describe a near-future air travel scenario to further explain and demonstrate the inner workings of our proposal. This paper extends previous work in the field of IoT security, where we proposed robust mechanisms that can achieve usable security and privacy in RFID systems [3].

The paper is organized as follows: In Section 2, we define the notion of RFID and identify a set of important security and operational requirements that a protocol needs to satisfy. Section 3 contains an overview of related research work. In Section 4, we discuss the use of security and privacy policies to provide fine-grain control of both tag operations (tag data) and object information. In Section 5, we describe our proposal for a modular secure tag management protocol. Sections 6 and 7 contain the security and complexity evaluation of the new protocol, respectively. Section 8 provides an air travel scenario that demonstrates the inner workings of our protocol and how it can be coupled with security and privacy policies to strengthen the security and privacy of the tag owner. Finally, in Section 9, we present some concluding remarks and provide directions for future research work.

2. BACKGROUND

Radio-frequency identification is a sensor-based technology used primarily to identify and track products or living organisms [4]. An RFID system can be viewed as consisting of two components: a front-end and a back-end part [5]. The front-end consists of embedded integrated circuit (IC) tags (transponders) that can be queried by reader devices (transceivers), whereas the back end server infrastructure manages the tag/object-related information. In its simplest form, when a reader queries a tag, the tag responds with an ID, thus identifying the tagged object.

In an abstract RFID system (Figure 1), such as the one assumed in most RFID security research papers, we can identify three main components:

- the *RFID tags* or *transporters*: IC microchips that are (usually) affixed to objects and carry identifying data for the said objects,
- the *RFID readers*, *tag readers*, or *transceivers*: devices that can read data from and (usually) write data to the tags, and
- a *data processing subsystem* (aka *back-end subsystem*): an information system (IS) that stores information about managed tags. Often, the subsystem is thought of as a (back-end) database or repository that associates tag identifiers to information related to the tagged objects, in which case we could further distinguish a *back-end application subsystem* that performs business-specific functions—as opposed to tag managing.

Radio-frequency identification tags may either be self-powered (active) or require power from an external source (passive), usually the reader or a hybrid, using both internal and external power sources. The main goal of such a system is to replace and enhance the now ubiquitous barcode, as well as allow new tracking, access management, and security services (e.g., e-passports and anti-counterfeiting mechanisms). Tag-related information can be grouped into tag data and object information. Tag data include data that support tag operations, such as tag secrets (keys) and unique identifiers. On the other hand, object information comprises of data related to the tagged object (e.g., description, owner, manufacturer) or the supported actions and services (e.g., physical access control, inventory management)

To make RFID systems economically viable, we have placed strict restrictions, mainly on the tag side, whose implementation has to be power, space, and time efficient. However, these restrictions cause severe security and privacy problems, because well-known and trusted

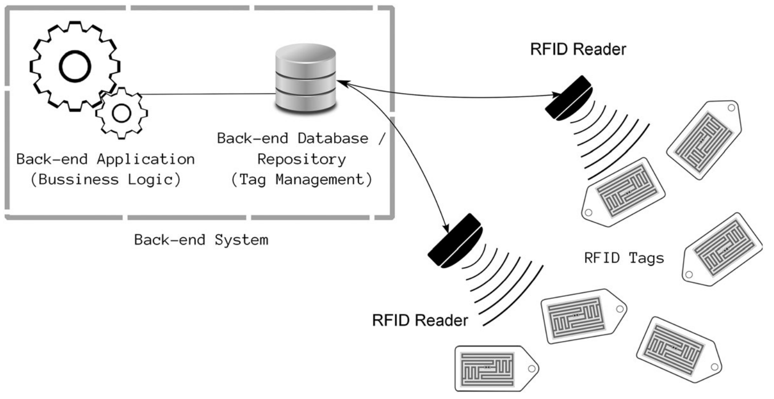

Figure 1. Abstract RFID system.

Table I. Hardware classification of RFID security protocols.

Class	Hardware requirements (cryptographic primitives)
Full-fledged	Conventional cryptographic functions, e.g., symmetric and/or asymmetric encryption algorithms
Simple	Cryptographic one-way hash function
Lightweight	Random number generator and simple functions, e.g., Cyclic Redundancy Code checksum
Ultralightweight	Simple bitwise operations, e.g., XOR, AND, OR

solutions, such as public key cryptography, are no longer applicable and efficient alternatives are required.

Chien [6] proposed a rough classification of RFID authentication protocols based on the computational cost and the operations supported by the tag. As shown in Table I, we can distinguish four protocol classes, namely, "full-fledged," "simple," "lightweight," and "ultralightweight," with diminishing hardware requirements. Hence, tags belonging to the "full-fledged class" implement protocols that require conventional cryptographic functions, for example, symmetric encryption, cryptographic hash function, or even public key algorithms. The second class called "simple" requires the support of a random number generator and/or a one-way hash function on tags. The "lightweight" class refers to protocols that require a random number generator and simple functions such as Cyclic Redundancy Code checksum, *but* not a hash function. The fourth class called "ultralightweight" refers to protocols that only involve simple bitwise operations (such as XOR, AND, OR) on the tag.

In order to protect tag holders' privacy and provide adequate security, it is useful to state clear security goals when discussing security properties of various RFID designs. Tags must not compromise the privacy of their holders. Information should not be leaked to unauthorized readers, nor should it be possible to build long-term tracking associations between tags and holders. To prevent tracking, holders should be able to detect and disable any tags they carry. Publicly available tag output should be randomized or easily modifiable to avoid long-term associations between tags and holders. Private tag content must be protected by access control mechanisms and - whenever interrogation channels are deemed insecure - encryption. Both tags and readers should trust each other. Spoofing either party should be neither trivial nor practical. Session hijacking and replay attacks are also concerns. Fault induction or power interruption should not compromise protocols or open windows to hijack attempts. Both tags and readers should be resistant to replay or man-in-the-middle (MIM) attacks. Thus, we identify five important security requirements that a security protocol should satisfy:

- *Resistance to tag impersonation*: an adversary should not be able to impersonate a legitimate tag to the reader.
- *Resistance to reader impersonation*: an adversary should not be able to impersonate a legitimate back end/reader to the tag.
- *Resistance to Denial of Service (DoS) attacks*: abnormal executions of the protocol and manipulating or blocking communication during a given number of sessions between the tag and the reader should not prevent any future normal interaction between the legitimate reader and tag. This kind of attack is also known as a desynchronization attack.
- *Indistinguishability (tag anonymity)*: tag output must be indistinguishable from truly random values. Moreover, it should be unlinkable to the static ID of the tag. To achieve a stricter notion of tag anonymity, we further define the following:

  - *Forward security/untraceability*. Even if an adversary acquires all the internal states of a target tag at time t, he or she should not be able to ascribe past interactions, which occurred at time $t' < t$, to the said tag.
  - *Backward security/untraceability**. Similarly to forward security, it requires that even if an adversary gains knowledge of a tag's internal state at time t, he or she should not be able to ascribe future/subsequent interactions, which occur at time $t' > t$, to the said tag.

Furthermore, we compile a set of desirable tag management operations, which contains the following:

- *Tag authentication*: an authorized reader/back-end system should be able to authenticate the tag.
- *Revocable access delegation*: (aka tag delegation) the capability to allow a third (delegated) party tag authentication, and read access to an owned tag while maintaining the right to revoke this privilege, under some predefined conditions.
- *Ownership transfer*: the capability to pass ownership of a tag to a third party without compromising backward untraceability for the said party or forward untraceability for the previous owner. The same operation could be used by the owner to re-initialize the tag.
- *Permanent and temporal tag invalidation*: more commonly known as kill and sleep operations, were initially proposed to offer a minimal degree of command over the tag. A legitimate tag owner can issue a command to disallow the tag from emitting any signals; in the case of the sleep operation, this ban

*In some research work, for example, Ref. [7], the terms are interchanged; that is, backward security is called forward security.

of communication can easily be revoked by the owner. Implementing them is trivial, and it is obvious that these operations can also be achieved by physical means, for example, breaking the tag or placing it in a Faraday cage.

3. RELATED WORK

Although an ultralightweight solution would be most welcomed, unfortunately, most such proposed protocols have been shown vulnerable to attacks. Vajda and Buttyán [8] proposed a set of extremely lightweight challenge–response authentication algorithms that by design could be broken by a powerful attacker. Peris-Lopez *et al*. designed a series of very efficient ultralightweight authentication protocols (viz. lightweight mutual authentication protocol [LMAP] [9], minimalist mutual authentication protocol [$M^2 AP$] [10], and efficient mutual authentication protocol [11]) using simple bitwise operations (XOR, OR, AND) and addition mod m. But these schemes were also successfully attacked by Li and Deng [12] and Li and Wang [13], who found that a powerful adversary can mount a desynchronization and a full-disclosure attack against all three protocols and proposed some improvements, and by Barasz *et al*., who described a full-disclosure passive attack (eavesdropping) against LMAP [14] and $M^2 AP$ [15]. Chien and Huang [16] further found weakness in Li–Wang's improved schemes. Li and Wang proposed another ultralightweight mutual authentication protocol, called SLMAP [49], whose traceability was attacked by Hernandez-Castro *et al*. [17] using a metaheuristic-based attack. Similarly, a protocol by Chien [6] was successfully attacked by Phan [18], where it was shown that a passive attacker could track a tag by obtaining information about its static ID. A heavier blow on ultralightweight protocols came from Alomair and Poovendran [19], who conducted a study in which they claimed that "relying only on bitwise operation for authentication cannot lead to secure authentication in the presence of an active adversary" [*sic*].[†] Respectively, in the lightweight camp, protocols have, as well, been notorious for their flaws. A striking example is the series of corrections and counter-corrections proposed on a family of lightweight protocols based on the learning parity with noise problem. Another example is that in Ref. [20], where we demonstrated that a lightweight Song–Mitchell authentication protocol [21] could be successfully attacked by a passive adversary and proposed a simple correction. We therefore maintain a cautious stance as to the security, achievable by ultralightweight and lightweight protocols.

What's more, going through the corpus of published research work on RFID security and privacy, we detected an imbalance between the offered services, even among *simple* RFID protocols. That is, most of the published work proposed tag authentication protocols, whereas other important operations were less explored. Indeed, the bibliography was rather limited; namely, Molnar *et al*. [22] proposed an authentication protocol using pseudonyms and secrets, organized in a tree structure, to offer secure ownership transfer and time-limited, recursive delegation; the tree scheme was compromised in Ref. [23]. Fouladgar and Afifi [24] also used pseudonyms to construct an authentication protocol, where delegation lasts for a predetermined number of queries. And a similar protocol, supporting a limited kind of delegation, was proposed in Ref. [23]. Ownership transfer, by itself, is also addressed in Refs [25–27,7,28–31]. Recently, Song and Mitchell [32] proposed a revised RFID protocol that appears to cover the security and functional requirements mentioned in Section 2, including the time-limited (counter-based) delegation and ownership transfer, while being scalable.

Overall, we believe that a shift of focus is necessary. Instead of offering stand-alone security services, we propose a holistic approach, governed by security and privacy policies that allow secure tag/object management. To this end, first, we describe an abstract framework for using policies to control tag information dissemination and then design a "complete" *simple* protocol that covers all the identified (RFID tag) security and privacy requirements (such as data confidentiality, backward and forward untraceability, etc.), supporting in a unified way operations, such as tag authentication, tag ownership transfer, and time-based tag delegation.

4. PRIVACY AND SECURITY POLICIES

Although an RFID security protocol can help reduce information leakage of tag data, by itself, it does not give to the user control over the disseminated tag/object information. A complete approach should provide the necessary tools to describe how and by whom resources may be used. By resources, we mean the tag data (secret keys, IDs, etc.), the object-related information, and the tag devices.

Traditionally, resources are protected using access control techniques. For data resources, mechanisms such as access control lists (ACL), capability-based access control, mandatory access control, role-based access control (RBAC), and more recently, attribute-based and rule-based access controls (ABAC and RuBAC) have been used in traditional systems [33,34].

Because of the envisioned dynamic and complex nature of RFID and IoT technologies, static approaches such as ACLs and RBAC are deemed unsuitable. Instead, research points out that RuBAC and/or ABAC systems seem a more suitable candidate for such services [35]. RuBAC and ABAC access decisions are based on the evaluation of rules expressed in terms of attributes and obligations of the subject, action, resource and environment. This allows finer granularity and context-aware authorization where required, even when the involved entities do not have

[†]We stress, again, that the hardware constraints refer only to the tag; the reader can satisfy more complex requirements, for example, a random number generator.

predefined relationships (in contrast, an ACL mechanism would require that all entities be known in advanced).

Policies themselves are expressed through the use of policy languages that define specialized grammar, syntax, and enforcement mechanisms, for example, XML-based access control language (XACML) [36]. There is a rich literature on policy languages [37], but a critical review of these is out of scope. In the remainder section, we will provide a high-level description of how a nonmonolithic security protocol can be coupled with privacy and security policies to provide fine-grain control to the end users.

For simplicity, we can assume that an RFID tag has an abstract four-step life cycle from birth (creation) to death (end of life/recycling), as depicted in the following:

(1) *Creation*. A tag is created, initialized (viz. given a [unique] identifier, secret and public data stored on tag, etc.), and bound to a data entry on the managing back-end infrastructure (e.g., a database server, repository, or even an intelligent agent [5])
(2) *Attachment*. The tag is attached to an object (inanimated item or living organization), and the data entry is expanded to include information pertaining to the tagged "thing," possibly in a new back end managed by the object's owner.
(3) *Operation*. The tag's daily usage, where authorized entities acquire access to the tag's operations (viz. tag querying, tag delegation, secret updating, ownership transfer) and information.
(4) *End of life*. The tag is no longer usable and is (hopefully) recycled.

The governing policies come as a natural extension of the tag information stored in the back end. Each tag, from its creation, may be bound to a policy that defines the attributes that an entity must hold, the obligation he/she must make, and the conditions under which tag operations are allowed. When a tag is attached to an object, along with the object information, suitable policies will be created to control access to this data.

Assuming a generic RFID system that uses an RFID authentication protocol (e.g., the one described in Section 5), we have the following scenario:

- When a tag query request first arrives to the managing back end, a first-layer policy will define whether the user/reader (requester) is allowed access to the back end's services. If the user holds the needed attributes, his query is forwarded to the back-end storage module that holds tag-related information (viz. tag data, object information and privacy policies). Otherwise, access is denied.
- At the tag information entry, a second-layer security policy will be consulted to check if the requester is authorized to perform the specific operation (in this case, tag authentication/query). If yes, the operation proceeds. Otherwise, access is denied.
- If the back end does not have an entry for the queried tag, a relevant message is returned. The contents of the message depend on the requester's trust level, because a policy may define that certain entities are not entitled to learn whether a tag is not managed by the back end.
- If the correct tag is found, a policy should define how much of the object information will be released to the requester.
- Tag protocols may support extra operations beyond simple tag authentication/query. Whether the requester is allowed to perform these depends again on tag policy. In essence, because these operations require that the back end returns the result of certain computations/data (e.g., decrypting a ciphertext), the policies allow or disallow them by controlling access to these computations/data.

Although the use of privacy policies might prove beneficial, there are problems that need to be addressed first. Such include the following:

- *Efficiency issues*. This includes policy evaluation at the infrastructure, storage costs, and the like [35].
- *Policy and rule construction and revision*. Although many policies use the XML to provide a form that not only is machine readable, but can also be reviewed by human users, this may become a barrier for non-technical users.
- *Access control complexity*. When moving from a closed well-managed RFID system to a highly dynamic, interconnected, and complex system such as the IoT, there is a considerable amount of complexity that will need to be expressed into the policies. A good balance between fine-grain control, usability, manageability, and cost will need to be reached.
- *Privacy issues regarding use of attributes*. Attributes hold information about entities. Releasing more attributes than necessary to gain access to a resource could lead to sensitive information disclosure.
- *Interoperability*. To achieve a unified IoT, not only heterogeneous RFID hardware, RFID management protocols, and back-end infrastructures but also policies will need to be able to communicate and operate with each other.

Although fine-grain control is required, it is nonetheless assumed that in the general case, policies would not differ in excess. A user will most probably group her items according to her privacy, security, and usability needs. Thus, the labor of writing individual policies for each and every tag is greatly reduced.

In addition, the literature provides research on efforts made to construct machine-readable policies using "natural" language rule editors [38], allowing not only easier policy creation but also policy revision from the user. It is thus easy to envision interested groups, such as privacy rights organizations, providing ready-made rules and policies for everyday use. Tweaking item grouping and generic policies will both provide the required level of control and abstraction needed.

Another challenging task is providing a privacy-preserving trust negotiation mechanism. Trust negotiation, simply put, is the bilateral exchange of digital credentials to establish trust gradually. When entities set up access policies and try to satisfy them, by exchanging proofs that they hold the necessary attributes, they release sensitive information (e.g., credentials) about themselves. Over the years, researchers have proposed several mechanisms that try to build trust and at the same time preserve users privacy, including trust managing systems and attribute release strategies [39,40].

Standardization efforts have been made to provide an interoperable environment, both in the hardware and software level. For example, the OASIS consortium has standardized the XACML, but more research is needed on the interoperability (bridging services) of existing mainstream languages.

In Section 5, we present an RFID management protocol with a modular design that can support security and privacy policies.

5. A NOVEL PROTOCOL FOR SECURE RADIO-FREQUENCY IDENTIFICATION MANAGEMENT

In this section, we describe a tag management protocol that supports, in a compact and uniform way, all basic tag operations, such as mutual authentication, tag delegation, and ownership transfer while satisfying the security and privacy requirements identified in Section 2. The protocol falls into the "simple" protocol class, because it imposes limited hardware requirements on the tag side. More specifically, the tag is required to implement only a secure one-way function $h(\cdot)$ and a pseudorandom number generator (the random selection of an element from a finite set using a uniform probability distribution is denoted as $\in_{\mathbb{R}}$). In addition, the tag needs to share only two values with the back-end system, namely, an l-bit secret value *secret* and a t-bit time value *horizon*, which designates a specific point in time and is publicly known. Time is an important concept for the delegation of the tag, and we assume that its representation comforts to the ISO 8601 international standard [41]. It is important to note that, unlike, for example, the timed efficient stream loss-tolerant authentication (TESLA) protocol [42], our proposal does not require synchronized clocks between the tag and the back-end system/reader.

Figure 2 provides a concise schematic diagram of the proposed protocol, which presents all the supported operations. To declutter the schematic diagram, we chose to depict the reader and the back end as one entity and skip the command signals that the reader sends to the tag; in practice, the reader would act as a middleman forwarding messages and might also be given the capability to generate certain data items, such as random nonces or timestamps.

Motivated by the observation that, practically, all the protocols supporting stand-alone tag operations begin with the authentication of the tag by the back end/reader, our protocol is divided into two phases denoted by the dashed line. In the first phase, the tag is authenticated by the (owner's or delegated entity's) back end/reader. In the second phase, initially, the back end/reader is authenticated by the tag, and then, depending on the performed tag management operation, some of the tag's data are updated. The protocol description follows:

Phase I: Tag authentication

- *Back end/reader* $\rightarrow$ *tag*: Forwards the identity *Rep_ID* of the reader's repository, an l-bit random *nonce* and the t-bit current time *c_time*.
- *Tag* $\rightarrow$ *back end/reader*: If *c_time* designates a point in time "older than" *horizon*, then the tag "updates" *c_time* to the horizon value. Following this check the tag generates a random l-bit $nonce_A$ and computes a time-dependent secret key, using the key update process $secret' \leftarrow chainedHash(secret,\ c_{time},\ horizon)$. Furthermore, it computes the corresponding k-bit identifier $TID \leftarrow h(Rep\_ID,\ secret')$ and a k-bit pseudonym $Pseud_T \leftarrow h(nonce_B, TID \oplus nonce_A)$. Then the tag forwards pseudonym $Pseud_T$ along with $nonce_A$. The function $chainedHash(s, t_1, t_2)$ is just the hashing of s repeated $t_1 - t_2$ times (Figure 3).
- *Back end (tag authentication phase)*: For each tag entry, in its storage, the back end computes an individual time-dependent key ($secret'$), an identifier ($TID \leftarrow h(Rep\_ID,\ secret')$), and subsequently, a pseudonym ($Pseud \leftarrow h(nonce_B, TID \oplus nonce_A)$). If the computed pseudonym is equal to the received one, then the tag has been successfully identified and authenticated. Note that if the tag is not authenticated, this step has to be computed twice using the old values of the [secret, horizon] pair stored for each tag. This is carried out to prevent desynchronization attacks.

Phase II: Tag data update

- *Back end/reader* $\rightarrow$ *Tag*: Chooses the desirable operation, "A" or "B," and forwards it along with the new t-bit horizon value $time_{new}$.
- *Tag* $\rightarrow$ *Back end/reader*: Generates and forwards an l-bit random $nonce_C$.
- *Back end/reader* $\rightarrow$ *Tag*: Computes a checksum value for the update ($checkV \leftarrow h(Oper, nonce_C, secret' \oplus time_{new})$). Forwards the k-bit value *checkV*. The back-end system stores both the new and the old values of the [secret, horizon] pair for the tag.
- *Tag (data update phase)*: Checks if the received *checkV* is equal to $h(Oper, nonce_C, secret' \oplus time_{new})$; if yes, based on the value of *Oper*, it updates the time-dependent secret, either by using the secret update process $chainedHash(secret, time_{new}, horizon)$ or by setting it to $secret' \oplus checkV$. The new *horizon* value is set to $time_{new}$.

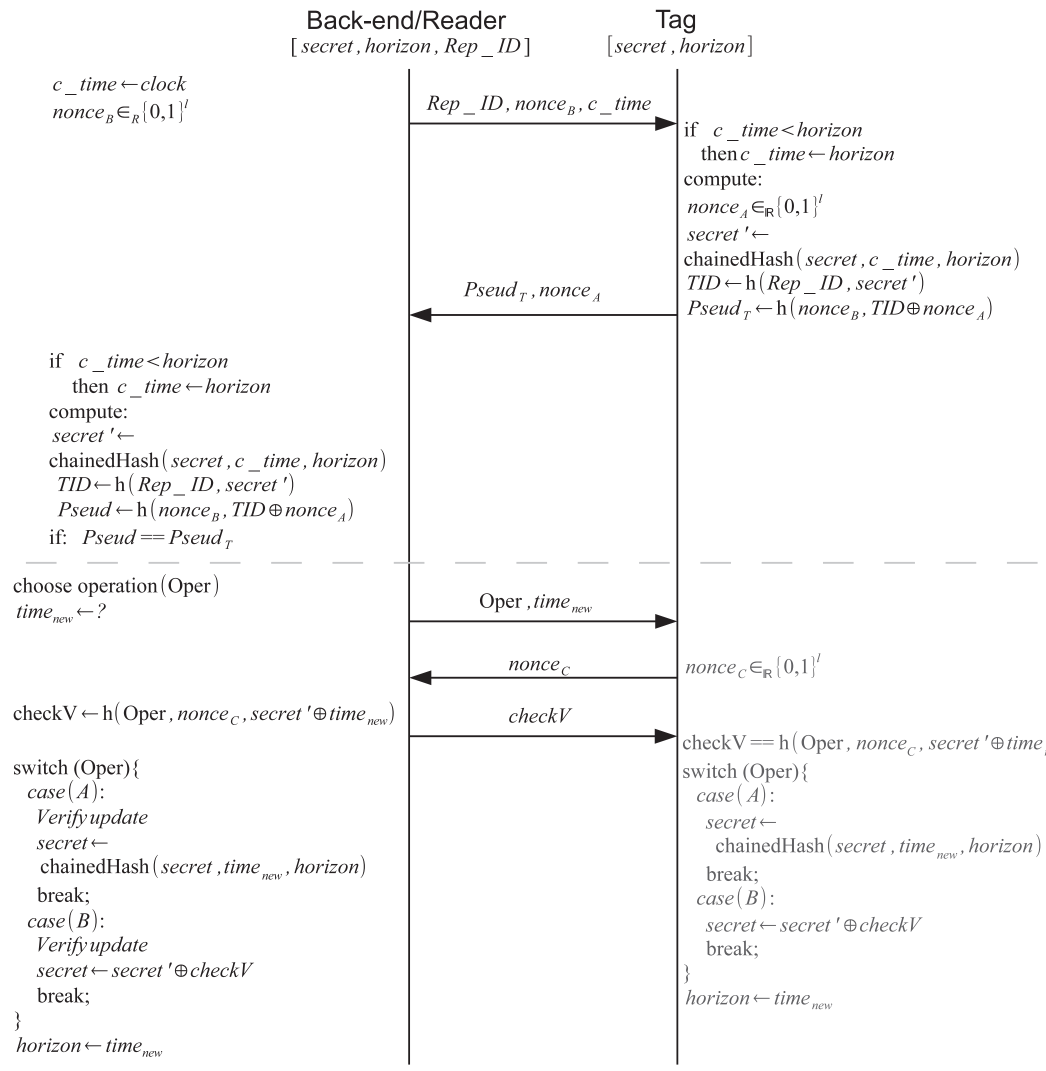
Back-end/Reader
$[secret, horizon, Rep\_ID]$
Tag
$[secret, horizon]$
$c\_time \leftarrow clock$
$nonce_B \in_R \{0,1\}^l$
$Rep\_ID, nonce_B, c\_time$
if $c\_time < horizon$
then $c\_time \leftarrow horizon$
compute:
$nonce_A \in_R \{0,1\}^l$
$secret' \leftarrow$
$\mathrm{chainedHash}(secret, c\_time, horizon)$
$TID \leftarrow \mathrm{h}(Rep\_ID, secret')$
$Pseud_T \leftarrow \mathrm{h}(nonce_B, TID \oplus nonce_A)$
$Pseud_T, nonce_A$
if $c\_time < horizon$
then $c\_time \leftarrow horizon$
compute:
$secret' \leftarrow$
$\mathrm{chainedHash}(secret, c\_time, horizon)$
$TID \leftarrow \mathrm{h}(Rep\_ID, secret')$
$Pseud \leftarrow \mathrm{h}(nonce_B, TID \oplus nonce_A)$
if: $Pseud == Pseud_T$
choose operation(Oper)
$time_{new} \leftarrow ?$
Oper, $time_{new}$
$nonce_C$
$nonce_C \in_R \{0,1\}^l$
$\mathrm{checkV} \leftarrow \mathrm{h}(\mathrm{Oper}, nonce_C, secret' \oplus time_{new})$
$checkV$
$\mathrm{checkV} == \mathrm{h}(\mathrm{Oper}, nonce_C, secret' \oplus time_{new})$
switch (Oper){
$case(A)$:
Verify update
$secret \leftarrow$
$\mathrm{chainedHash}(secret, time_{new}, horizon)$
break;
$case(B)$:
Verify update
$secret \leftarrow secret' \oplus checkV$
break;
}
$horizon \leftarrow time_{new}$
switch (Oper){
$case(A)$:
$secret \leftarrow$
$\mathrm{chainedHash}(secret, time_{new}, horizon)$
break;
$case(B)$:
$secret \leftarrow secret' \oplus checkV$
break;
}
$horizon \leftarrow time_{new}$

Figure 2. Compact schematic diagram of the proposed protocol.

```
Function chainedHash($s, t1, t2$)
{
  $temp \leftarrow s$
  for($i$ from $t2$ to $t1$)
    $temp \leftarrow \mathrm{h}(temp)$
  return $temp$
}
```

Figure 3. Chained hash function pseudocode.

In phase II, the protocol supports two different operations for the secret key update, namely, operation "A" and operation "B." With operation "A," the secret key is updated according to a predefined chained hash function (Figure 3), whereas with operation "B," the secret key is reloaded with a specific value. During each execution of phase II, the publicly known value *horizon* is reset. More precisely, in day-to-day operations, the tag's *horizon* time value is expected to be set to the current time. However, to revoke a granted delegation, the owner may use a specific horizon value, different from the current time.

Note that both operations are supported without influencing the number or the length of the exchanged messages between the tag and the back-end system. In total, the supported tag management functionalities include the following:

- *Tag authentication/identification*. The authentication/identification is achieved in phase I using the *Rep_ID* identity value of the "owning" back end. The protocol may terminate here, with no further data sent to the tag, if authentication is the only desired operation.
- *Tag re-initialization*. There are cases where the owner wants to disrupt the linkability between the subsequent secret values. It might be the case where we suspect that, at some point in the past, an adversary tampered with the tag and gained access to its stored data (i.e., secret and horizon values). The owner can easily achieve this by using the secret renewal process in operation "B." We assume that these steps are performed during a "safe slot," that is, a period during

which no adversary with knowledge of the current *secret* value eavesdrops the communication.

- *Delegated tag authentication*. The owner of the tag can delegate to another entity the right to successfully authenticate the tag for a given period c_{time}. To achieve this, it produces a time-dependent tag identifier $TID = h(Rep\_ID, secret')$, where $secret' = chainedHash(secret, c_{time}, horizon)$. This identifier is unique for each system with identity *Rep_ID* for the given period c_{time}. Although $horizon \leq c_{time}$, the *Rep_ID* system can authenticate the tag using phase I of the protocol.
- *Revocation of tag delegation*. By selecting a new value for *horizon* greater than c_{time}, the delegation is revoked. The "owning" back-end system can use the protocol with operation "A" (Figure 2) to update the value of *horizon* with the new value $time_{new}$.
- *Ownership transfer*. By executing phase II with operation "B," first, the previous and, subsequently, the new tag owner can alter the value of the tag *secret* so as to ensure forward and backward untraceability. We assume that these steps are performed during a "safe slot," that is, a period during which no adversary, with knowledge of the current *secret* value, eavesdrops the communication.

6. PROTOCOL SECURITY EVALUATION

First, we define an adversary model matrix, according to the available attack actions/capabilities [43–45]. More precisely, attackers can be distinguished into those that can tamper the tag (*corruptive*), that is, can take the IC apart and extract, delete, or alter data and those that cannot (*weak*). Furthermore, an attacker is characterized *wide* if he has access to (side channel) information about the outcome of the protocol (e.g., whether tag identification process was successful or not). Table II details the different adversarial types and highlights their capabilities. For all defined adversary models, we assume the existence of a secure communication channel between the reader and the back-end system. In addition, we expect that both reader devices and the back-end system use suitable security mechanisms to protect against attacks toward them (infrastructure hardening).

To give a characteristic example, according to Table II, a *wide-Passive* adversary is one that can only eavesdrop on the unencrypted communication between the tag and the reader and has knowledge of whether the tag authentication was successful or not. A *wide-Strong* adversary, on the other hand, is the one that not only can manipulate the communication channel—according to the Dolev–Yao threat model, that is, eavesdrop, corrupt, insert, etc. messages, mount MIM, and replay attacks—but also can corrupt the tag (altering and/or reading the data stored in the tag) whenever he sees fit.

It is important to clearly define actions 3–5 of Table II to avoid any misconception. According to the forward privacy model, a *forward* attacker is allowed to corrupt the data stored in the tag, but only at the end of the attack, so that no further active action happens after corruption, whereas action 4 defines that an (*destructive*) attacker may corrupt the tag, whenever he sees fit, but, after that, the tag is destroyed. Of course the adversary may continue his attack, for example, by simulating the tag. The fifth action allows the attacker to access and manipulate the tag at his convenience, without further limitations.

For every identified security requirement, we will describe how it is satisfied by our protocol for the strongest possible adversary model:

- *Resistance to tag and reader impersonation*. This requirement is studied under the weak adversary models; to prevent stronger (*corruptive*) attackers, one would need to use hardware anti-tampering techniques, which are out of scope. For an *active* attacker, the protocol can prevent malicious manipulation of the tag data. Any changes to the tag data or to the relevant data stored to the back end are carried out after authenticating the received input and verifying its integrity. Replay attacks are thwarted by using random nonces. A MIM attack on the unprotected front channel would not yield anything for the attacker because all secret information is enciphered.
- *Resistance to DoS*. In order to avoid tag desynchronization (intentional or unintentional), the last two pairs of tag data (i.e., the current and previous secret and *horizon* values) are stored at the back-end system for each tag.
- *Indistinguishability (tag anonymity)*. The tags always reply using pseudonyms, which depend on the current secret and the exchanged random nonces. Even when the secret is not updated, the tag's reply will seem random to those that do not have access to the current secret or temporal ID. Thus, the protocol can defend

Table II. Adversary matrix.

	Weak		Corruptive		
Actions	Passive	Active	Forward	Destructive	Strong
1. Eavesdrops	✓	●	●	●	●
2. Full control of network operations	–	✓	✓	✓	✓
3. Tag corruption at the end of the attack	–	–	✓	●	●
4. Destructive tag corruption	–	–	–	✓	●
5. Arbitrary tag corruption	–	–	–	–	✓
6. Side channel knowledge	Wide	Wide	Wide	Wide	Wide

–, the action is not available; ✓, the action is available; ●, a more powerful action is available.

itself against active attackers. (For corruptive attackers, see further on this paper.)

- *Forward security/untraceability*. The protocol provides forward security, even under the strong attacker model. If a corruptive attacker gains access to the tag data in time *t*, he or she cannot correlate past interactions to the tag (which were carried out using older keys), thanks to the "one-way" property of the secret update process.
- *Backward security/untraceability*. As soon as a corruptive adversary gains access to the tag data in time *t*, he becomes able to trace all subsequent tag interactions—for the destructive adversary, this type of attack is not applicable, because the tag is destroyed and no further interaction is possible. The only way to regain untraceability is by exploiting a safe slot to disrupt the chained hash update process and change the secret to a new unrelated value (i.e., operation "B" of phase II).

7. COMPLEXITY CONSIDERATIONS

In this section, we will analyze the time, communication, and memory complexity of our protocol, and we will investigate some optimization trade-offs that do not downgrade its security. We compare our protocol with the revised SM-2 protocol [32], which, according to its authors, provides similar functionality and security features. To ease comparison, we have made the necessary conversions and assumptions; namely, we have assumed that the hash functions in both our protocol and the SM-2 protocol have a *k*-bit output, nonces have an *l*-bit length, and so on. Table III summarizes the analysis.

7.1. Memory complexity

As seen in Table III, storage requirements are minimal, especially for the tag. More precisely, each tag needs to store one *k*-bit *secret* key and one *t*-bit *horizon* value. The repository/back end needs to store for each tag two pairs of *secret* and *horizon* values, the current pair and the previous pair; that is, $2k+2t$ bits per tag. In the case of delegated tags, the delegated third party's repository needs to store, per tag, at least one *k*-bit temporary tag identifier (*TID*) and the corresponding *t*-bit time value (*time*). The number of (*TID*, *time*) pairs that must be stored per tag depends on the quantization of time that is used and the duration of the delegation. See Section 7.4 for more details.

7.2. Communication complexity

Phase I requires the exchange of two messages: one from the (delegated) reader containing an *r*-bit repository identity value, an *l*-bit nonce value, and a *t*-bit time value; and one message from the tag consisting of a *k*-bit pseudonym value and an *l*-bit nonce value. That is, in total, $2l+k+r+t$ bits are exchanged.

In phase II, the reader needs to send two messages; one with a *t*-bit time value and the second one with a *k*-bit checksum value. On the other hand, the tag needs to send only one message, containing an *l*-bit nonce value. In total, $l+k+t$ bits are exchanged. By making some conservative assumptions about the bit length of the variable in use, viz. the implemented hash function has an output of 128 bits, identifiers and nonces have a length of 96 bits, and

Table III. Comparison of memory, communication, and computation complexity.

	Tag	Repository	Delegated repository
Storage cost (bits)			
Our protocol	$k+t$	$2n\cdot(k+t)$	At least $\hat{n}\cdot(k+t)$
SM-2	$2\cdot k+q$	$n\cdot(2^q+3)\cdot k$	at least $\hat{n}\cdot\hat{q}\cdot k$
Communication cost for phase I			
Our protocol	$k+l$	$l+r+t$ bits	
	1 message	1 message	
SM-2	$2\cdot k+l$ bits	l bits	
	1 message	1 message	
Communication cost for phase II			
Our protocol	l bits	$k+t$ bits	–
	1 message	2 messages	–
SM-2		$k+l$ bits	–
		1 message	–
Processing cost of tag authentication			
Our protocol	$2+m$ hashes	$n\cdot(2+m)$ hashes	$\hat{n}$ hashes
SM-2	1 hash	1 OR n hashes	1 hash
Processing cost of tag data update			
Our protocol	$1+m$ hashes	$1+m$ hashes	–
SM-2	2 hashes	$2+q$ hashes	–
Processing cost of tag data renewal			
Our protocol	1 hash	1 hash	–

n is the number of managed owned tags.
$\hat{n}$ is the number of managed "delegated" tags.
$m = max(0, c_{time} - horizon)$.

that we need at least 64 bits to represent time, one can calculate that a tag authentication requires the exchange of 480 bits, whereas the update process needs 288 bits.

7.3. Computation complexity

For ease of presentation, we distinguish between the processing cost of an authentication, a secret/horizon updating, and a secret renewal round. The most demanding computation is hashing.

For *tag authentication* (phase I), the number of hashing computations that the tag and the back-end system perform depends on the difference between the value of c_{time} and *horizon*. Let $m = max(0, c_{time} - horizon)$. The tag computes $2 + m$ hashes, whereas the back-end system, in the worst-case scenario, computes $n \cdot (2 + m)$, where n is the number of tags that are managed by this system.

According to the terminology used by Alomair and Poovendram [46], our proposal is a stateless linear-time identification protocol, because tags are able to randomize their responses internally. Although this ability enhances tags privacy, it introduces an identification inefficiency. Owing to the "one-way" property of the cryptographic hash function, the authorized back-end system has no means of identifying the tag other than to perform an exhaustive search through its tag space. Given that for n tags, the system is expected to perform an average of $(n/2) \cdot (2 + m)$ hashes before it can identify the tag, the complexity of the identification process is $O(n)$.

This scalability problem plagues all (working) challenge–response-based protocols we are aware of. To counteract it, some authors opt for precomputed look-up tables that increase memory consumption [32] or for tag clustering techniques, for example, the tree structure technique proposed by Molnar *et al.* [47]. These solutions either degrade privacy or use time-memory trade-offs, as shown in Ref. [48].

In case of a temporary desynchronization between a tag and the back-end system, the whole procedure must be repeated using the stored old values. In which case, the back-end system performs $2n \cdot (2 + m)$ hashes (worst-case scenario) in total. On the other hand, a delegated third party needs to compute only one hash for every delegated tag using the corresponding *TID*. If $\hat{n}$ is the number of delegated tags that the back-end system is temporarily authorized to authenticate, then, in maximum, it has to compute $\hat{n}$ hashes.

For *tag data updating*, that is, phase II and operation "A," both the tag and the repository compute $1 + m$ hashes. Finally, for tag data renewal, that is, phase II and operation "B," both the tag and the repository compute only one hash.

7.4. Optimization considerations

The computational cost of the protocol strongly depends on the number of iterations performed by the chained hash function (Figure 3). This number depends on the difference between c_{time} and *horizon* on one hand and on the quantization of time that has been selected on the other hand. Although we cannot manipulate the first, we can optimize the choice of the second.

More precisely, the ISO 8601 standard supports up to fractions of a second without limiting the number of decimal places for the decimal fraction. Just to give an example, let us assume that the difference between c_{time} and *horizon* is two hours (2 h). If the basic quantum or unit of time is an *hour*, then only two hashes will be performed by the chained hash function. If, on the other hand, the quantization was in *seconds*, the number of hashes skyrockets to 3600.

The main use of the time values (*horizon*, c_{time}) is to support revocable delegation so the chosen accuracy will be a trade-off between fine-grain delegation management and processing/storage cost. Too small a period translates to more hash operations, whereas too big a period leads to less control over delegation, because the minimum delegation period is one time quantum/unit. It is thus crucial to choose a suitable time unit for the purpose at hand.

Interested readers can refer to Refs [48] and [46] for a review of the scalability problem imposed by privacy-preserving RFID protocols.

7.5. Comparison with related protocols

As mentioned in Section 3, Song and Mitchell [32] recently proposed a new RFID protocol (SM-2). According to its description, this protocol appears to offer comparable security and functionality features with our own. SM-2 supports tag authentication, tag ownership transfer, and tag delegation. The protocol makes use of precomputed look-up tables to improve scalability. This leads to a complexity of $O(log_2 n)$, under normal operation (see further on the SM-2 paper), but increases storage requirements at the back end. Per tag, the back end needs to store $2^q - 1$ precomputed pseudonyms, where q is the bit length of an on-tag counter. Table III compares the resources requirements between our protocol and SM-2.

On the hardware side, the protocol requires tags to implement a counter, a pseudorandom number generator, three keyed hash functions, and a hash function; one of the keyed hash functions has an output of $2 \cdot k + q$; the rest have a k-bit output. Each tag stores a k-bit key, a k-bit pseudonym, and the q-bits long state of its counter, which is initially set to $2^q - 1$. The back end stores per tag a k-bit secret value, the current k-bit key, and a $2^q - 1$ long hash chain consisting of k-bit pseudonyms. In addition, the back end also stores the previous values of s and k to protect against accidental desynchronizations.

The protocol defines three states/cases: Case 1 corresponds to our phase I and allows tag authentication under normal operation, that is, as long as the tag counter is greater than zero (the counter decreases with every attempted authentication). Case 1 is the only process permissible to delegated entities. Case 2 corresponds to our phase II, operation "B," and is executed whenever the back end runs out of precomputed pseudonyms, namely, $counter = 1$. Lastly, case 3 (non-normal) is executed when the tag counter is zeroed ($counter = 0$); in this state, the back end is forced to go through every tag entry in its storage until it identifies the tag and then re-synchronizes it. Case 3 has a complexity of $O(n)$ and is considered abnormal, because it is expected to occur only after desynchronization and failed updates.

8. AIR TRAVEL SCENARIO

To better explain the inner workings of the described protocol and to demonstrate how it can be coupled with security and privacy policies, we will describe an air travel scenario (Figure 4) inspired by ENISA's Flying 2.0 study [2]. For this scenario, we assume that the time unit is 1 day, because this seems a reasonable compromise between fine-grain delegation control and computation load.

Listing 1: Delegation Policy

```
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
   PolicyId="TagDelegationPolicy"
   RuleCombiningAlgId="rule-combining-algorithm:first-
   applicable">
<Description>
Allow luggages' tag delegation to airline company 's check-in
   counter employees, for the next 5 days
</Description>
 <Target>
   <Subjects><AnySubject/></Subjects>
   <Resources>
    <Resource>
     <ResourceMatch MatchId="function:string-equal">
      <AttributeValue DataType="string">
              type=<application>,
              application=Alice-GenerateLuggageTIDs
      </AttributeValue>
     </ResourceMatch>
    </Resource>
    ...
   </Resources>
   <Actions>
    <Action>
     <ActionMatch MatchId="function:string-equal">
     <AttributeValue DataType="string">Generate</AttributeValue>
       <ActionAttributeDesignator DataType="string"
       AttributeId="action:action-id"/>
     </ActionMatch>
    </Action>
    ...
   </Actions>
 </Target>
 <Rule RuleId="GenerateTIDRule" Effect="Permit">
  <Condition FunctionId="function:and">
   <Apply FunctionId="function:string—one—and—only">
    <SubjectAttributeDesignator DataType="string"
                                AttributeId="Airgroup"/>
     <AttributeValue DataType="string">
        check-in counter
     </AttributeValue>
     </Apply>
    <Apply FunctionId="function:dateTime-less—than—or—equal">
     <Apply FunctionId="function:dateTime—add—dayTimeDuration">
       <Apply FunctionId="function:dateTime—one—and—only">
        <ResourceAttributeDesignator DataType="dateTime"
           AttributeId="environment:current-time"/>
     </Apply>
     <AttributeValue DataType="dayTimeDuration">
        P5D
     </AttributeValue>
    </Apply>
    <Apply FunctionId="function:dateTime—one—and—only">
     <EnvironmentAttributeSelector DataType="datetime"
        AttributeId="TID DateTime"/>
    </Apply>
   </Apply>
   ...
  </Condition>
 ...
 </Rule>
 <Rule RuleId="deny—rule"Effect="Deny"/>
</Policy>
```

8.1. Background settings

Alice is an IoT aficionado who uses RFID tags to manage her belongings, both in-house and outside. At her house, a PC hosts a tag managing back end/repository (*Home_Rep*) that "communicates" with her tags through RFID reader devices, her smart phone being one such reader device. Alice is packing her suitcases because she is flying to a security conference for a presentation. Thanks to IoT technology, packing is made easier because she can compile a list of necessary items to pack, she knows the current availability of items, and so on.

8.2. Step 1: initialization

While Alice is in her house, she feels safe and has minimal security and privacy concerns. With regard to tag managing, her reader devices are authorized to access the repository and query tags. Authorization is granted through the use of suitably formed policies in the repository that evaluate reader credentials; as described in Section 4. Tag queries are made using the main protocol procedure "*Phase I: Tag Authentication*," but to speed up operations, her repository has precomputed the TIDs of related items, in this case, clothing articles, accessories, and other articles she might need to take with her. Of course, the privacy policies in place that regulate access to the tag-related information in the repository allow unrestricted access to Alice. Moreover, the repository intelligently uses the readers' location information (e.g., room) to limit the tag search space. Once Alice has finished packing, a final check is made to ensure that she has not forgotten anything. At that point, she chooses to minimize *tag noise* (i.e., tag replies to unauthorized readers) by putting all packed items/tags to sleep.

When she electronically checked in, Alice had not decided yet how much she will pack, so she opted to perform luggage registration at the airport's counter. Thus, she requests the *Home_Rep* to transfer the required information of the luggage tag to her smart phone; at minimum, this contains the *secret*, the *horizon* value, and the object's (luggage's) ID and/or description. Along with this information, Alice constructs a suitable policy that will allow designated (by role or attribute) employees of the airline company to request and access the said data. An abstract example of such a policy, written in a XACML-like syntax, is given in listing 1. In it, Alice describes an application "Alice-GenerateLuggageTIDs," which runs in her smart phone device and can produce suitable TIDs for her luggage. The application provides a "generate" service that can be accessed only by employees of the airline company who work in the check-in counter and for a given duration (5 days). The actual implementation and the authentication process (e.g., how can Alice know that the employee belongs to the specified group) is beyond the scope of this paper,

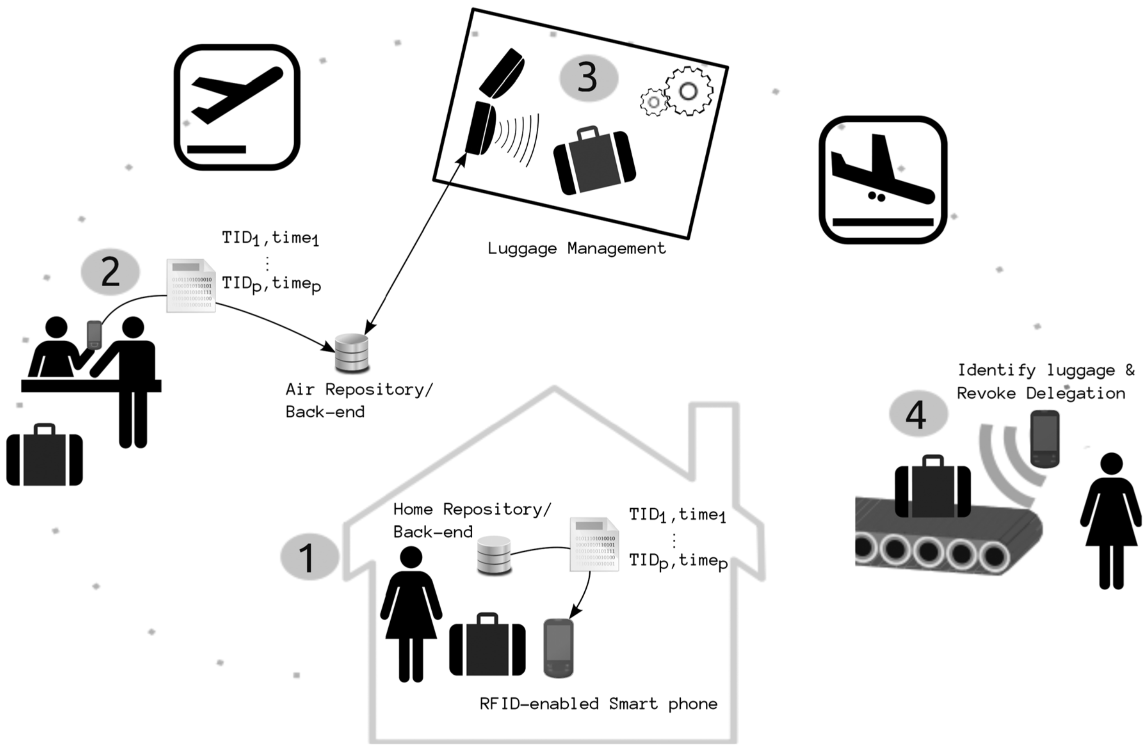

Figure 4. Air travel scenario.

but one can easily imagine the use of (attribute) certificates.

8.3. Step 2: delegating tag querying rights

The airline company (service provider) requires query rights for every checked article (e.g., bags, sports equipment) throughout the travel duration. So it requests passengers to submit per article enough *TID*s for 5 days.[‡] Submission may take place during the e-checking or at the airport, provided that the user has a suitable (handheld) device that can store and transmit such information. The service provider stores in its repository (*Air_Rep*) per passenger, per submitted item the TIDs. Their validity can be verified by using them to query the said items at the check-in counter. If the tag's reply is incomprehensible, then something has gone amiss, and the passenger will be asked to resubmit the correct information. Passengers are requested to put to *sleep* or *kill* any other contained tag. Also, active devices that might tamper or conflict with the system are prohibited, unless they are packed inside a Faraday cage.

[‡]Actually, any suitably long period defined by the service provider and taking into consideration the existing legal framework under operation and the criticality of the offered service.

Listing 2: Delegation Request

```
<Request>
<Description>
Bob a check-in counter employee for the airline company requests
   from Alice's device to generate TIDs
</Description>
 <Subject>
  <Attribute AttributeId="subject:subject-id"
              DataType="data-type:rfc822Name">
   <AttributeValue>Bob@air.com</AttributeValue>
  </Attribute>
  <Attribute AttributeId="Airgroup
                        " DataType="string"
                         Issuer="admin@air.com">
   <AttributeValue>check-in counter</AttributeValue>
  </Attribute>
  ...
 </Subject>
<Resource>
  <Attribute>
   <AttributeValue DataType="string"">
                 <AttributeId="resource:resource-ancestor">
                 type=<application>,
                 application=Alice-GenerateLuggageTIDs,
   </AttributeValue>
  </Attribute>
  ...
 </Resource>
 <Action>
  <Attribute AttributeId="action:action-id"
            DataType="string">
   <AttributeValue>Generate </AttributeValue>
  </Attribute>
  ...
 </Action>
</Request>
```

At the airport, Alice, who has already put to *sleep* her packed tags, is asked by Bob, who is working at the check-in counter, to generate the required number of *TID*s for her two pieces of baggage. Bob submits a request for the required *TID*s, where he states his name, position, and other necessary attributes (listing 2). Because this request satisfies the privacy policies in place, a positive response is returned (listing 3). The response can itself include further *obligations* that Bob must satisfy. Obligations can represent formal requirements that cannot be easily expressed by access control rules; for example, the obligation to notify Alice by a text message that he has accessed the TIDs. Through an intuitive friendly interface, which guides her through the whole process, Alice instructs her smart phone to use *Air_Rep* as the *Rep_ID* to compute for every luggage, per day:

$$secret' \leftarrow chainedHash(secret, time, horizon)$$

and

$$TID \leftarrow h\left(Air\_Rep, secret'\right)$$

where *time* takes five discreet values; namely, today, tomorrow, ..., the fifth day. She then passes the [*TID, time*] pairs to the airline company over a secure channel. From this point onward, Alice has no actual control over the dissemination of the TIDs, but of course, she can always revoke access to the tag once she physically re-acquires her luggage. The airline will pass part of the TIDs to the ground handling service, the airport authorities, and others to provide the required services.

Listing 3: Delegation Response

```
<Response>
 <Result>
  <Decision>Permit</Decision>
  <Status><StatusCode Value="status:ok"/></Status>
  <Obligations>
   <Obligation FulfillOn="Permit"
    <ObligationId="Roles">
       <AttributeAssignment DataType="string"
                         AttributeId="role">check-in counter
       </AttributeAssignment>
      </Obligation>
      ...
     </Obligations>
 </Result>
</Response>
```

8.4. Step 3: tag querying by delegates

During its routine operation, the handling service will query the managed items using "*Phase I: Tag Authentication*" of the protocol with the "daily" TIDs, that is, *time* =current day. Moreover, the provider can himself outsource the operation to a third party by disclosing "today's" TID. The repository (*Air_Rep*) controls access to stored information by means of a suitable security policy (as discussed in Section 4).

Every time a piece of luggage goes through a control point, where a reader device is present (e.g., check-in counter, pressure/X-ray chambers, loaded on-board, unloaded, baggage carousel), its status in the back end (*Air_Rep*) is updated so that subsequent queries in that checkpoint or earlier do not perform the described cryptographic operations on it (location awareness).

8.5. Step 4: revoking delegation

Once Alice lands, she goes to collect her luggage at the baggage carousel. Using "*Phase I: Tag Authentication*" she can easily identify and retrieve her luggage. She then revokes the delegated rights by updating the horizon value to a date beyond the "fifth day," using "*Phase II: Tag Data update—Operation 'A'*." From that point onward, the airline company can no longer "recognize" Alice's articles.

9. CONCLUSIONS

In this paper, we have discussed the use of security and policy languages to control access to tag information and tag operations to allow for finer granularity and context-aware authorization in RFID systems. We believe that this is an interesting topic that needs more research, especially in integrating the so far proposed systems and mechanisms and transforming them into a suitable tool for use with RFID-related operations. In the second part of the paper, we described a novel unified tag management protocol that supports, among others, secure and privacy-preserving tag authentication, delegation, and ownership transfer. The protocol has minimal requirements on the tag side and follows a clear modular design. To showcase the delegation capabilities of the protocol, we provided a near-future air travel scenario inspired by the Flying 2.0 study.

REFERENCES

1. Communication to the European parliament, the council, the EESC and the committee of the regions: Internet of Things—an action plan for Europe. *Technical Report*, Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions Jun 2009.
2. Flying 2.0—enabling automated air travel by identifying and addressing the challenges of IoT & RFID technology. *Technical Report*, European Network and Information Security Agency (ENISA) Apr 2010.
3. Rekleitis E, Rizomiliotis P, Gritzalis S. A holistic approach to RFID security and privacy. Tokyo, Japan, 2010.
4. OECD. Radio-frequency identification (RFID): drivers, challenges and public policy considerations. *Technical Report DSTI/ICCP(2005)19/FINAL*, Organisation for Economic Co-operation and Development (OECD), Paris Mar 2006.
5. Rekleitis E, Rizomiliotis P, Gritzalis S. An agent based back-end RFID tag management system. In *Lecture*

Notes in Computer Science LNCS, Katsikas S, Lopez J (eds). Springer: Bilbao, 2010.
6. Chien HY. SASI: a new ultralightweight RFID authentication protocol providing strong authentication and strong integrity. *IEEE Transactions on Dependable and Secure Computing* 2007; **4**:337–340. doi: 10.1109/TDSC.2007.70226
7. Lim CH, Kwon T. Strong and robust RFID authentication enabling perfect ownership transfer. In *ICICS*, *LNCS*, Vol. **4307**, Ning P, Qing S, Li N (eds). Springer: Raleigh, NC, 2006; 1–20. doi: 10.1007/11935308_1
8. Vajda I, Buttyán L. Lightweight authentication protocols for low-cost RFID tags. *Second Workshop on Security in Ubiquitous Computing—Ubicomp 2003*. Seattle, WA, 2003.
9. Peris-Lopez P, Hernandez-Castro JC, Estevez-Tapiador J, Ribagorda A. LMAP: a real lightweight mutual authentication protocol for low-cost RFID tags. *Workshop on RFID Security—RFIDSec'06*. Ecrypt: Graz, 2006.
10. Peris-Lopez P, Hernandez-Castro JC, Estevez-Tapiador J, Ribagorda A. M2AP: a minimalist mutual-authentication protocol for low-cost RFID tags. In *International Conference on Ubiquitous Intelligence and Computing—UIC'06, Lecture Notes in Computer Science*, Vol. **4159**. Springer-Verlag: Wuhan and Three Gorges, 2006; 912–923.
11. Peris-Lopez P, Hernandez-Castro JC, Estevez-Tapiador J, Ribagorda A. EMAP: an efficient mutual authentication protocol for low-cost RFID tags. *OTM Federated Conferences and Workshop: IS Workshop—IS'06, Lecture Notes in Computer Science*, Vol. **4277**. Springer-Verlag: Montpellier, 2006; 352–361.
12. Li T, Deng RH. Vulnerability analysis of EMAP—an efficient RFID mutual authentication protocol. *Second International Conference on Availability, Reliability and Security—AReS 2007*. IEEE: Vienna, Austria, 2007.
13. Li T, Wang G. Security analysis of two ultra-lightweight RFID authentication protocols. *IFIP SEC 2007*. IFIP: Sandton, Gauteng, 2007.
14. Barasz M, Boros B, Ligeti P, Loja K, Nagy D. Breaking LMAP. *Conference on RFID Security*, Malaga, Spain, 2007.
15. Barasz M, Boros B, Ligeti P, Lója K, Nagy D. Passive attack against the M2AP mutual authentication protocol for RFID tags. *First International EURASIP Workshop on RFID Technology*, Vienna, Austria, 2007.
16. Chien HY, Huang CW. Security of ultra-lightweight RFID authentication protocols and its improvements. *SIGOPS Operating System Review* 2007; **41**(4): 83–86. doi: 10.1145/1278901.1278916
17. Hernandez-Castro JC, Estevez-Tapiador J, Peris-Lopez P, Clark JA, Talbi EG. Metaheuristic traceability attack against SLMAP, an RFID lightweight authentication protocol. Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium—IPDPS 2009, Rome, Italy, 2009.
18. Phan RCW. Cryptanalysis of a new ultralightweight RFID authentication protocol—SASI. *IEEE Transactions on Dependable and Secure Computing* 2009; **6**: 316–320. doi: 10.1109/TDSC.2008.33
19. Alomair B, Poovendran R. On the authentication of RFID systems with bitwise operations. *New Technologies, Mobility and Security NTMS'08*. IEEE: Tangier, 2008; 1–6.
20. Rizomiliotis P, Rekleitis E, Gritzalis S. Security analysis of the Song–Mitchell authentication protocol for low-cost RFID tags. *Communications Letters* April 2009; **13**: 274–276. doi: 10.1109/LCOMM.2009.082117
21. Song B, Mitchell CJ. RFID authentication protocol for low-cost tags. In *ACM Conference on Wireless Network Security, WiSec'08*, Gligor VD, Hubaux J, Poovendran R (eds). ACM Press: Alexandria, VA, 2008; 140–147.
22. Molnar D, Soppera A, Wagner D. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In *SAC, LNCS*, Vol. **3897**, Preneel B, Tavares SE (eds). Springer: Kingston, 2006; 276–290. doi: 10.1007/11693383_19
23. Dimitriou T. RFIDDOT: RFID delegation and ownership transfer made simple. *Proceedings of the 4th International Conference on Security and Privacy in Communication Networks*, SecureComm '08, ACM: New York, NY, 2008; 34:1–34:8. doi: 10.1145/1460877.1460921
24. Fouladgar S, Afifi H. An efficient delegation and transfer of ownership protocol for RFID tags. *First International EURASIP Workshop on RFID Technology*, Vienna, Austria, 2007.
25. Jin Y, Sun H, Chen Z. Hash-based tag ownership transfer protocol against traceability. *IEEE International Conference on E-Business Engineering* 2009; **0**: 487–492. doi: 10.1109/ICEBE.2009.77
26. Zuo Y. Changing hands together: a secure group ownership transfer protocol for rfid tags. *Hawaii International Conference on System Sciences* 1899; **0**: 1–10. doi: 10.1109/HICSS.2010.100
27. Saito J, Imamoto K, Sakurai K. Reassignment scheme of an RFID tags key for owner transfer. In *EUC Workshops, LNCS*, Vol. **3823**, Enokido T, Yan L, Xiao B, Kim D, Dai YS, Yang LT (eds). Springer: Berlin, Heidelberg, 2005; 1303–1312. doi: 10.1007/11596042
28. Osaka K, Takagi T, Yamazaki K, Takahashi O. An efficient and secure RFID security method with ownership transfer. *Computational Intelligence and Security: International Conference, CIS 2006, Guangzhou, China, November 3–6, 2006, Revised Selected Papers*. Springer, 2007; 778–787.

29. Koralalage KH, Reza SM, Miura J, Goto Y, Cheng J. POP method: an approach to enhance the security and privacy of RFID systems used in product lifecycle with an anonymous ownership transferring mechanism. *Proceedings of the 2007 ACM Symposium on Applied Computing SAC'07*, ACM: Seoul, Korea, 2007; 270–275. doi: 10.1145/1244002.1244069
30. Song B. RFID tag ownership transfer. *Conference on RFID Security*, Budapest, Hungary, 2008.
31. van Deursen T, Mauw S, RadomiroviÄ‡ S, Vullers P. Secure ownership and ownership transfer in RFID systems. *Proceedings 14th European Symposium on Research in Computer Security (ESORICS'09), Lecture Notes in Computer Science*, Vol. 5789, Springer, 2009; 637–654.
32. Song B, Mitchell CJ. Scalable RFID security protocols supporting tag ownership transfer. *Computer Communications* Apr 2011; **34**(4): 556–566. doi: 10.1016/j.comcom.2010.02.027
33. Bishop M. *Computer Security: Art and Science*. Addison-Wesley Professional: Boston MA, USA, 2002.
34. Ferraiolo DF, Kuhn DR, Chandramouli R. *Role-based Access Control, Second Edition*, (2nd edn). Artech House: Norwood MA, USA, 2007.
35. Grummt E, Müller M. Fine-grained access control for EPC information services. *Proceedings of the 1st International Conference on the Internet of Things*, Springer-Verlag: Zurich, 2008; 35–49.
36. OASIS eXtensible access control markup language (XACML) TC. URL www.oasis-open.org/committees/xacml, last retrieved March 2011.
37. Kumaraguru P, Cranor FL, Lobo J, Calo SB. A survey of privacy policy languages. *In SOUPS '07: Proceedings of the 3rd Symposium on Usable Privacy and Security*, ACM: New York, NY, 2007.
38. Stepien B, Felty A, Matwin S. A non-technical user-oriented display notation for XACML conditions. In *E-Technologies: Innovation in an Open World, Lecture Notes in Business Information Processing*, Vol. **26**, Aalst W, Mylopoulos J, Sadeh NM, Shaw MJ, Szyperski C, Babin G, Kropf P, Weiss M (eds). Springer: Berlin, Heidelberg, 2009; 53–64. doi: 10.1007/978-3-642-01187-0_5
39. Seamons KE, Winslett M, Yu T, Yu L, Jarvis R. Protecting privacy during on-line trust negotiation. *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, Springer-Verlag: San Francisco, CA, 2003; 129–143.
40. Winslett M. An introduction to trust negotiation. *Proceedings of the 1st International Conference on Trust Management*, Springer-Verlag: Heraklion, Crete, 2003; 275–283.
41. ISO 8601:2004. *Data elements and interchange formats—information interchange—representation of dates and times*. ISO: Geneva, 2004.
42. Timed efficient stream loss-tolerant authentication (TESLA): multicast source authentication transform introduction. http://tools.ietf.org/html/rfc4082 Jun 2005. URL http://tools.ietf.org/html/rfc4082, last retrieved March 2011.
43. Dolev D, Yao AC. On the security of public key protocols. *Foundations of Computer Science, Annual IEEE Symposium on*, Vol. 0. IEEE Computer Society: Los Alamitos, CA, 1981; 350–357. doi: 10.1109/SFCS.1981.32
44. Avoine G. Adversary model for radio frequency identification. *Technical Report LASEC-REPORT-2005-001*, EPFL, Lausanne, Switzerland Sep 2005.
45. Vaudenay S. On privacy models for RFID. *Advances in Cryptology—Asiacrypt 2007*, Lecture Notes in Computer Science. Springer-Verlag: Berlin, Heidelberg, 2007; 68–87.
46. Alomair B, Poovendran R. Privacy versus scalability in radio frequency identification systems. *Computer Communications* Dec 2010; **33**(18): 2155–2163. doi: 10.1016/j.comcom.2010.08.006
47. Molnar D, Soppera A, Wagner D. Privacy for rfid through trusted computing. *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*. WPES '05, ACM: New York, NY, 2005; 31–34. doi: 10.1145/1102199.1102206
48. Avoine G. Scalability issues in privacy-compliant RFID protocols. *RFID Security: Techniques, Protocols and System-On-Chip Design*, Kitsos P, Zhang Y (eds). Springer: US, 2008; 191–228.
49. Li T, Wang G. SLMAP - A Secure ultra-Lightweight RFID mutual authentication protocols. *Proceedings of Chinacrypt'07*. Science Press: Cheng Du, China, 2007; 19–22.