An Agent Based Back-End RFID Tag Management System

Evangelos Rekleitis, Panagiotis Rizomiliotis, and Stefanos Gritzalis

Dep. of Inf. and Comm. Syst. Eng., University of the Aegean
Karlovassi, Samos, GR 83200, Greece
{erekl,prizomil,sgritz}@aegean.gr

Abstract. Motivated by the plethora of RFID security protocols and the interoperability problems that this diversity causes, we propose a software agent-based platform that allows an RFID back-end subsystem to integrate and manage heterogeneous tags that are based on non-standardized implementations. In addition, we introduce a new suite of lightweight tag management protocols that support tag authentication, time-based tag delegation and ownership transfer. The protocols can take advantage of the proposed agent-based platform and do satisfy all the standard security and privacy requirements.

Keywords: RFID, software agent, privacy, security, delegation, ownership.

1 Introduction

Radio Frequency Identification (*RFID*) is a sensor-based technology, used, primarily, to identify and track products or living organisms [12]. This is achieved by using devices, called transceivers or readers, to query embedded integrated circuits, called transponders or tags. RFID 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, anti-counterfeiting mechanisms, etc.). To make RFID systems economically viable, strict restrictions have been placed, mainly, on the tag side, whose implementation has to be power, space and time efficient. However, these restrictions cause sever security and privacy problems, since well known and trusted solutions, like public-key cryptography, are no longer applicable, and efficient alternatives are required.

Going through the corpus of published research work on RFID security and privacy, one realizes that, the main focus is in the front-end system communication, i.e. the reader-tag interaction; while, only a limited number of published work studies the back-end part of the system (e.g. [14,10]). This can be partially justified by the general belief that the back-end system, usually, comprises of well known and understood server-based technologies ([4]). Thus, it comes at no surprise that there exists a plethora of cryptographic algorithms, protocols and tag implementations, incompatible with each other, that satisfy different and, at times, contradicting requirements. This babel has caused grave interoperability problems.

Moreover, an even closer look of the published papers, reveals that, while tag authentication is covered by numerous protocols, the issue of tag delegation has not been sufficiently addressed yet. Tag delegation meaning the capability to allow a third party, tag authentication and read access to an owned tag, while maintaining the right to revoke this privilege, under some predefined conditions. Molnar et al. [11] 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 [3]. Fouladgar et al. [5] also used pseudonyms to construct an authentication protocol, where delegation lasts for a predetermined number of queries. A similar protocol, supporting a limited kind of delegation, was proposed in [3]. Other research on ownership transfer include [15,9,13,8,16].

In this paper, motivated by the interoperability problems caused by the diversity of cryptographic protocols, we propose a software agent-based RFID back-end system that simplifies the integration and management of heterogeneous RFID tags. The new system is able to implement different security or communication protocols, by offloading part of the management and communication functions, from the back-end and the tag reader, to a, per tag dedicated, agent, residing in a repository. The concept of using software agents to complement RFID-based systems is not new. In [7] mobile software agents were employed to monitor the transport parameters of tagged items, and in [2] a multi agent-based back-end application subsystem, excluding tag management, is proposed, while Chen et. al. in [1] implement the whole back-end subsystem, including tag management, using a Multi Agent System. However, to the best of our knowledge, no previous work exists, that combines both RFID and agent technologies and specifically addresses security and privacy concerns. We fill this gap by making the agent based back-end security conscious. Due to space constraints we have chosen to focus on the tag managing portion of the data processing subsystem, not dwelling in technicalities.

At the second part of the paper, we complement this architecture with a novel suite of tag management protocols. The proposed suite of protocols covers all the standard, important security and privacy requirements (data confidentiality, backward & forward untraceability, etc) supporting, among other operations, tag authentication, tag ownership transfer and time-based tag delegation. It goes without saying that, while all the protocols fit nicely the proposed agent-based infrastructure, they, none the less, can also be used with a more conventional back-end system without any modifications.

The paper is organized as follows. In Section 2, the software agent-based platform is described and its main characteristics are analyzed. In Section 3, a new suite of lightweight protocols for tag management is introduced and its security is evaluated. Finally, in Section 4, we present some concluding remarks and provide some directions for future research work.

2 Agent-Based RFID Platform

In this section, we propose a software agent-based RFID platform that addresses security and privacy concerns. In the vast majority of the RFID security research papers, the abstract RFID system consists of 3 main components, namely the **RFID tags**, the **tag readers** and the **back-end subsystem** (Fig. 1).

The back-end subsystem is responsible for managing all information related to the tags. It can be thought as the combination of a *Back-end Database*, which associates tag identifiers to information related to the tagged objects, and a *Back-end Application subsystem* that performs business specific functions ([17]). It is assumed that communication between the back-end components as well as between the back-end subsystem and the readers is secured, by suitable means, while the reader-tag communication is not.

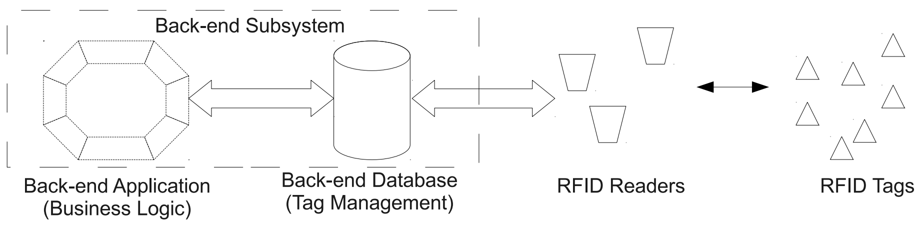
Back-end Subsystem
Back-end Application (Business Logic)
Back-end Database (Tag Management)
RFID Readers
RFID Tags

Fig. 1. Abstract RFID System

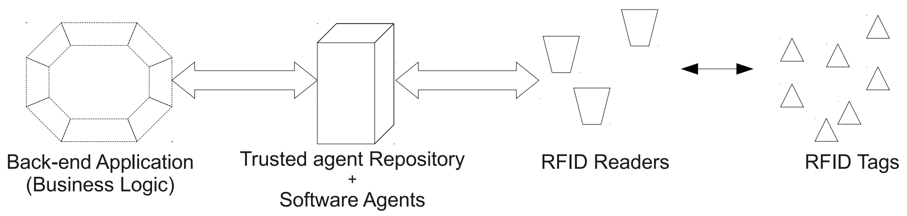
Back-end Application (Business Logic)
Trusted agent Repository + Software Agents
RFID Readers
RFID Tags

Fig. 2. RFID System with Software Agents

The proposed agent-based platform appears in Fig. 2. The Back-end Database and the tag-related records are replaced by an *Agent Repository* and *Tag Agents*, respectively.

The **Agent Repository** is a host platform that provides, to residing or visiting agents, necessary computational resources and services. From a security perspective, we distinguish between trusted and untrusted (alien) repositories. A trusted repository is one complying with our security and privacy (S&P) policy, and acts as a safe haven for the managed agents. That is, it enforces all necessary security means to protect them, prevent unauthorized access and tampering. To each entity that owns and wishes to manage a set of tags, corresponds a trusted repository, thus there is no need for a central 'trusted' agent repository.

A **Tag Agent** is an autonomous, software entity that manages one and only one RFID tag. Each tag agent stores relevant tag information, including all data required to interact with the rest of the back-end and to manage the RFID tag (including tag owner's credential, S&P policy, important business logic etc).

By default, it is assumed that the tag agents reside in the protected environment of a trusted repository and, for security reasons, are not allowed to travel outside of it. However, mobility is a desired characteristic, as there are cases where we would like to send an agent to an alien platform; e.g. when a tagged item changes owner, or when we wish to delegate tag access to an off-line reader or third party. For this, we allow the creation of partial **agent clones** that hold only a subset of the original agent's information. By using suitable security and privacy policies, we can control the amount of information exposed, while maintaining functionality.

2.1 Basic Tag Operations

We will now describe, in brevity, how the main tag-related operations are performed by the agent-based platform.

Tag Initialization: During a tag's initialization (e.g. assignment of a unique identifier (uID), secret key), an agent is created that has all the necessary information (credentials, policies, tag information, communication protocols, etc.) to communicate securely both with the rest of the back-end system, as well as, the tag.

Tag's Data Access: Agents enforce access control mechanisms, to allow authorized entities to access, append or modify tag's data, based on the entity's permissions. So, it is possible to regularly and in an automatic way update/modify information (e.g. location, status, etc.) related to a tagged item.

Tag Authentication/Secret Update: Any of the proposed authentication and secret update protocols can be applied. More specifically, the agent is able to instruct the reader on the correct protocol to use, when communicating with the tag, and provide the required information. So, with respect to security, the agent knows which authentication protocol the tag implements and has knowledge of the secret keys used and it is able to update and generate those keys, identifiers, pseudonyms etc., accordingly.

Tag Ownership Transfer: For tag implementations supporting secure ownership transfer, the agent-based back-end can facilitate the process and ease information management. To wit, the previous owner is able to decide the amount of information she wants to pass to the new owner. This is achieved by creating and forwarding, to the new owner, a suitably constructed *agent clone* that supports the tag secret update operation. For example, at a retailer's point-of-sale, the retailer would remove any sensitive data pertaining to the shop's logistics system, as well as, perform a secret update operation (to alter the secret, stored in the tag, to a temporary value). The agent clone would then be populated with the new temporary secret, along with any data required by law (e.g. expiration date, manufacturer, etc.) and data that would add value to the transaction and allow for better after sales services. As a final step, the new owner takes tag-ownership by performing the secret update operation, under a controlled environment (to avoid eavesdropping from the retailer).

Tag Delegation: This operation is supported through the use of clone agents. A suitable clone is created and forwarded to the delegated entity, allowing interaction with the corresponding tag, sans access to the tag owner's repository. For implementations not supporting revocable delegation rights, a tag owner would only use this trivial delegation for trusted entities under his control. On the contrary, when revocation is practicable and desirable, the tag's owner would provide a specially constructed agent clone, according to the protocol instructions. As soon as the delegation expires or is revoked, the clone would stop functioning and the delegated entity wouldn't be able to interact with the tag any longer, thus preserving backward security. The protocol, we present in Section 3, allows for temporal, revocable delegation.

2.2 Advantages

The perceived benefits of the agent-based platform include:

Support for heterogeneous tags: Since reader-tag interaction is supervised by agents, the back-end can be oblivious of the implementation details.

Simplified key management: Accordingly, all relevant actions are offloaded to the agent, minimizing the complexity of handling heterogeneous tags or tags having different security & privacy needs.

Facilitates ownership transfer and tag delegation: For implementations supporting such advanced features, the platform provides suitable agent clones, coupled with appropriate privacy policies, to manage information exposure. On the receiver's side, they offer the advantage of carrying the tag's implementation details; alleviating worries about introducing and integrating a foreign technology to his own platform. (Problems, such as code safety, must be dealt suitably before executing an alien agent.)

Support for fine grained S&P policies: In addition to any generic policies residing in the Repository, each agent carries his own, individual, S&P policy. Thus the owner can choose between enforcing a general one-for-all policy or applying specialized and elaborate policies on individual tags or tag groups. Such policies will also be able to govern reader devices and effectively restrain or permit access to important information.

Support for complex business logic: The proposed infrastructure, can be instructed to manipulate tags' data, according to the organization's needs, introducing automation in data gathering and data handling.

3 A Novel Suite of Lightweight RFID Management Protocols

In this section, we describe a suite of lightweight tag management protocols that can take advantage of the proposed agent-based infrastructure. The proposed protocols support tag operations, like authentication, delegation/revocation and ownership transfer, by satisfying at the same time important security and privacy requirements, such as data confidentiality and untraceability. While all the new protocols nicely fit the proposed agent-based infrastructure, they can, also, be used in more typical RFID systems, consisting of back-end databases, with trivial modifications.

3.1 Basic Protocols

The proposed suite supports tag delegation (by using time-based / temporal pseudonyms) and privacy preserving ownership transfer (with secret updating), while imposing limited hardware requirements. More specifically:

The tag must implement a secure one-way function $h(\cdot)$ and a pseudorandom number generator (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 store 3 values, namely an l-bit secret value *secret*, shared between the tag and the corresponding agent, a time-dependent tag identifier (TID) and a time value ($horizon$), which designates a specific point in time and is publicly known. Time is an important concept in the delegation of the tag and it's representation, comforts to the ISO 8601 international standard [6].

Next, we describe the main suite protocols, namely *tag query, delegated tag query, secret updating and time horizon updating*. The delegated tag query protocol is of special significance, as it supports the delegation of a tag, by taking advantage of the new agent-based platform. In addition, this delegation is only temporal and it is automatically revoked after a given time period.

Tag Query

1. The reader sends to the tag: the ID of the reader's repository (Rep_ID), an l-bit random $nonce_A \in_{\mathbb{R}} \{0,1\}^l$ and the current time, c_time.
2. If c_time designates a point in time 'older than' $horizon$, then the tag replaces c_time with $horizon$. It generates a $nonce_B \in_{\mathbb{R}} \{0,1\}^l$ and computes a time-dependent identifier $TID_{c\_time} = h(Rep\_ID, secret, c\_time)$. Then it computes a pseudonym $Pseud = h(nonce_A, TID_{c\_time}, nonce_B)$, which is sent to the reader along with $nonce_B$.
3. The reader forwards the received values, along with $nonce_A$ and c_time to the Repository.
4. At the Repository, the freshness of the received c_time is checked against the clock; if a discrepancy is found, suitable actions, e.g. raising an alarm, take place. Further, each agent compares c_time to the stored $horizon$ value; and

if found older they replace it with *horizon*. Subsequently, every agent computes it's own time-depended ID ($TID'_{c\_time} = h(Rep\_ID, secret, c\_time)$) and then computes the pseudonym $Pseud' = h(nonce_A, TID'_{c\_time}, nonce_B)$ and compares it to the tag's (received) *Pseud*.

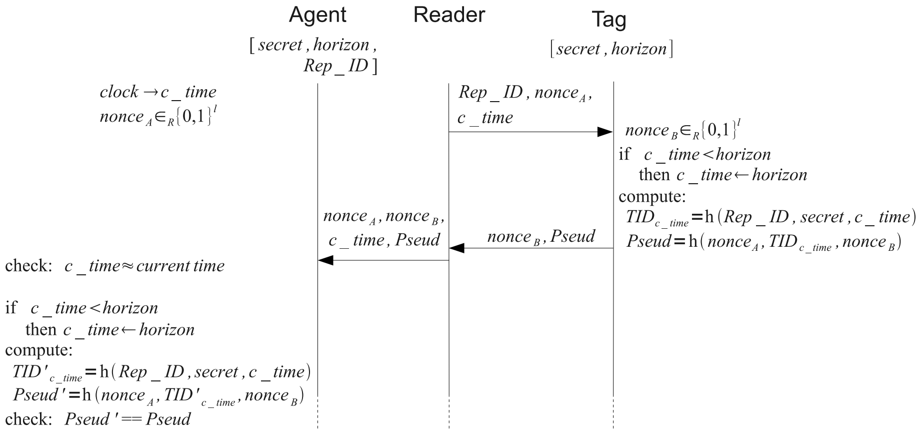
Agent
Reader
Tag
[secret, horizon, Rep_ID]
[secret, horizon]
clock → c_time
nonce_A ∈_R {0,1}^l
Rep_ID, nonce_A, c_time
nonce_B ∈_R {0,1}^l
if c_time < horizon then c_time ← horizon
compute:
TID_c_time = h(Rep_ID, secret, c_time)
Pseud = h(nonce_A, TID_c_time, nonce_B)
nonce_B, Pseud
nonce_A, nonce_B, c_time, Pseud
check: c_time ≈ current time
if c_time < horizon then c_time ← horizon
compute:
TID'_c_time = h(Rep_ID, secret, c_time)
Pseud' = h(nonce_A, TID'_c_time, nonce_B)
check: Pseud' == Pseud

Fig. 3. Compact schematic of Tag Query

A match, signals the discovery of the correct agent. Having found the managing agent, the application can continue with its business process.

Delegated Tag Query: The protocol, depicted in Fig. 4, is identical to the Tag Query protocol, with one notable exception, a clone agent is used. The owner of the tag creates a delegation agent clone, *d_Agent*, and sends it to the 'temporary user' of the tag. The clone, does not store the secret value, but instead stores one fixed, time-dependent identifier $TID_{d\_time} = h(Rep\_ID, secret, d\_time)$ and the corresponding time value (d_time). The identifier is precomputed by the original managing agent, to be used by a specific Repository (Rep_ID), for the predetermined time period. The *d_Agent* can be used to track and locate the tag for as long as the horizon value, stored in the tag, is anterior or equal to d_time. Therefore, the tag owner can revoke the tag delegation, by updating the horizon with a posterior time value. As soon as the tag is updated with the Time Horizon Update protocol, the *d_Agent* clone becomes obsolete.

We deliberately have not enforced any additional checks on the freshness of the current time c_time. In the security analysis section the design choices will be justified in detail.

Tag Secret Update

1. Execute Tag Query protocol and if successful continue.

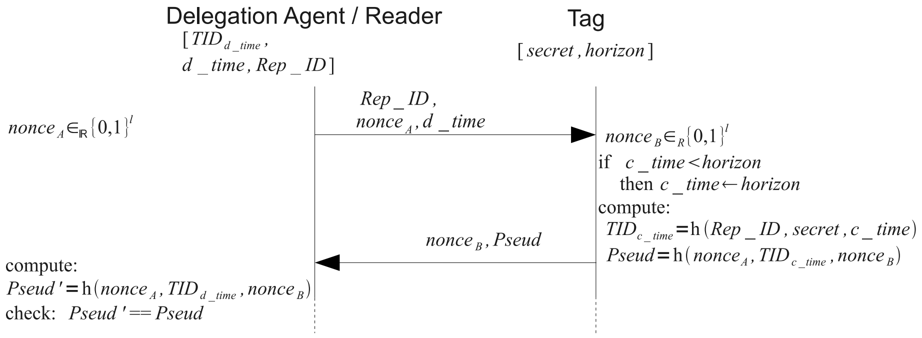
Delegation Agent / Reader
$[TID_{d\_time}, d\_time, Rep\_ID]$
Tag
$[secret, horizon]$
$nonce_A \in_R \{0,1\}^l$
$Rep\_ID, nonce_A, d\_time$
$nonce_B \in_R \{0,1\}^l$
if $c\_time < horizon$ then $c\_time \leftarrow horizon$
compute: $TID_{c\_time} = h(Rep\_ID, secret, c\_time)$
$Pseud = h(nonce_A, TID_{c\_time}, nonce_B)$
$nonce_B, Pseud$
compute: $Pseud' = h(nonce_A, TID_{d\_time}, nonce_B)$
check: $Pseud' == Pseud$

Fig. 4. Compact schematic of Delegated Tag Query

2. The agent chooses a new secret ($secret_{(new)}$), it generates a third $nonce_C \in_{\mathbb{R}} \{0,1\}^l$ and computes a session key

$$session\_key = h(secret_{(old)} \oplus nonce_B, nonce_C).$$

   It then XORs both ($NS = session\_key \oplus secret_{(new)}$) and calculates a checksum value $Scheck = h(secret_{(old)} \oplus nonce_C, secret_{(new)} \oplus nonce_B)$. It sends to the Reader all three values: $nonce_C, NS$ and $Scheck$.
3. The Reader forwards all three values to the Tag.
4. The Tag computes the session key

$$session\_key' = h(secret_{(old)} \oplus nonce_B, nonce_C),$$

   extracts the new secret ($secret'_{(new)} = NS \oplus session\_key'$) and computes $Scheck' = h(secret_{(old)} \oplus nonce_C, secret'_{(new)} \oplus nonce_B)$. If the checksums match then it replaces the old secret with $secret'_{(new)}$.
5. The agent performs a Tag Query operations using $secret\_new$, if successful, the tag was updated and the agent's old secret is replaced with $secret\_new$.

Time Horizon Update

1. Execute Tag Query protocol and if successful continue.
2. The agent chooses the new horizon value, $horizon_{(new)}$, and computes a checksum value

$$NH = h(horizon_{(new)}, secret, nonce_B).$$

   It sends both to the reader.
3. The Reader forwards both values to the Tag.
4. The Tag computes $NH' = h(horizon_{(new)}, secret, nonce_B)$ and checks if it is equal to the received NH. If yes, it replaces the existing $horizon_{(old)}$ value with $horizon_{(new)}$.

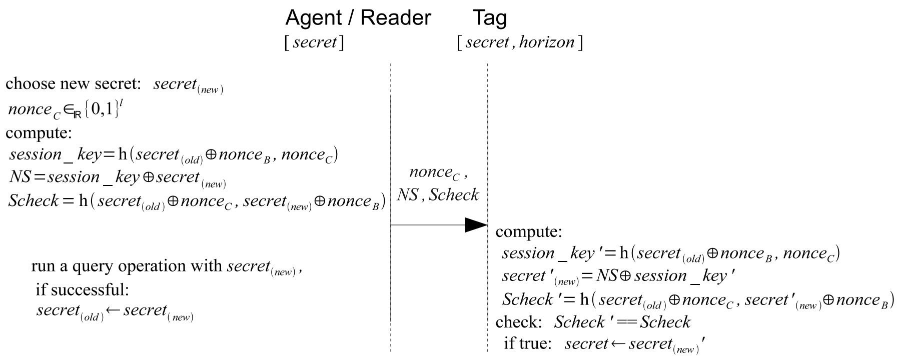
Agent / Reader
[secret]
Tag
[secret, horizon]
choose new secret: secret(new)
nonceC ∈R {0,1}^l
compute:
session_key = h(secret(old) ⊕ nonceB, nonceC)
NS = session_key ⊕ secret(new)
Scheck = h(secret(old) ⊕ nonceC, secret(new) ⊕ nonceB)
nonceC, NS, Scheck
compute:
session_key' = h(secret(old) ⊕ nonceB, nonceC)
secret'(new) = NS ⊕ session_key'
Scheck' = h(secret(old) ⊕ nonceC, secret'(new) ⊕ nonceB)
check: Scheck' == Scheck
if true: secret ← secret(new)'
run a query operation with secret(new),
if successful:
secret(old) ← secret(new)

Fig. 5. Compact schematic of Tag's Secret Update

5. The agent performs a subsequent Tag Query operation, using a *c_time* value older than $horizon_{(new)}$. If the tag replies with a pseudonym generated using the value $TID_{horizon_{(new)}}$, we can be assured the horizon value was updated correctly.

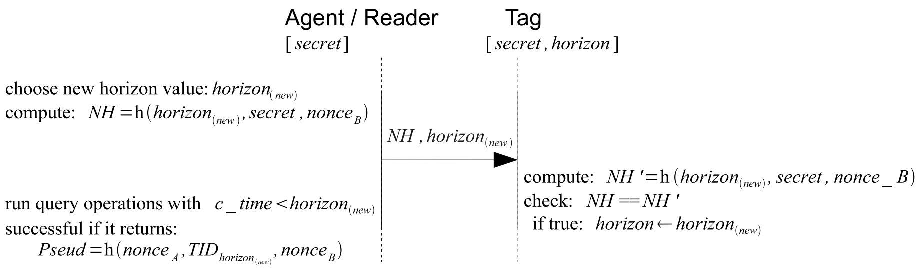
Agent / Reader
[secret]
Tag
[secret, horizon]
choose new horizon value: horizon(new)
compute: NH = h(horizon(new), secret, nonceB)
NH, horizon(new)
compute: NH' = h(horizon(new), secret, nonce_B)
check: NH == NH'
if true: horizon ← horizon(new)
run query operations with c_time < horizon(new)
successful if it returns:
Pseud = h(nonceA, TID horizon(new), nonceB)

Fig. 6. Compact schematic of time Horizon Update

3.2 Security Analysis

Due to space constraints, we will concentrate on the delegated tag query protocol, since it is the one that mainly takes advantage of the new agent-based platform and, at the same time, is the most challenging one. The analysis of the other protocols is much easier and in most the cases the same arguments apply.

We have no illusions, about the security of the delegation agent. As soon as it leaves the trusted Repository, the new holder can manipulate it, in any way she wants, e.g. by stripping the *d_Agent*'s logic and keeping only the data and TID_{d_time}. Therefore, implementing, in the clones' logic, freshness or any other kind of checks, beyond the one enforced by the TID, cannot increase security. The security of the protocol depends only on the security of the one-way function, the randomness of random generator and the size (l bits) of the secret values.

We enumerate some of the main privacy and security properties that the protocol satisfies, below:

1. **Tag Information Privacy:** The delegated agent clone contains only the minimum possible information concerning the tag, viz. all the information the temporary owner needs to know. The most crucial data concerning the tag, of course, is the secret key. However, by construction, only the values TID_{d_time} and d_time are stored in the delegated agent clone. Thus, even if the attacker acquires the clone, her gain (for instance by inverse engineering the clone agent's code) is limited to this values. Since the one-way function used is considered secure, the attacker cannot learn any extra information.
2. **Tag Location Privacy:** The responses of the tag are anonymous, since the tag emits, only, two messages, $nonce_B$ and $Pseud$, that differ, in a random way, every time a new query arrives. Thus, these messages cannot be linked to any particular tag.
3. **Tag Impersonation attack:** this attack is possible only if the tag is cloned. Otherwise, as analyzed above, the secret key can be computed only by inversing the hash function, which is computationally infeasible.
4. **Resilience to Replay attack:** The protocol is a typical challenge-response authentication protocol using random numbers to resist replay attacks.
5. **Resilience to locating a tag, using a revoked delegation agent clone:** this attack is considered successful when the attacker is able to locate a tag that has been previously delegated, even after the revocation of this delegation. There are two main concerns. The first has to do with the accuracy of the current time and the other with the temporary value TID_{d_time}.

   We have chosen to design the protocol in such a way that the accuracy of the current time is not critical. In other words, it is not expected that all entities will be truthful —to the extent of their clock accuracy— on what the current date and time is. To wit, lying about time doesn't affect the security of our protocol. We will now see why this holds:

   Assuming that the attacker has access to an old (revoked) agent clone. If she chooses to send any time value, older than the tag's *horizon*, the tag will reply with a random nonce and a pseudonym that depends on the *horizon* and the nonce. Given the security of the one-way function $h(\cdot)$, she won't be able to link it to the clone's TID_{d_time} value or distinguish it from a random value. If she sends a contemporary time value (equal or newer than *horizon*) the tag's reply will again appear random; even though now the pseudonym will depend on the nonce and the time value sent. Whats more, even if the attacker retrieves the value TID_{d_time} from the delegated agent, she won't be able to extract any further information concerning the tag.

4 Conclusions and Future Work

In this paper, we have briefly introduced an RFID system, based on software agents and agent cloning, in order to cope with interoperability problems, caused by the diversity of tag management protocols. In addition, we complemented, the said infrastructure, by describing a suite of novel lightweight tag management protocols that support, among others, secure and privacy preserving tag authentication, delegation and ownership transfer.

Future research work includes interesting issues, such as identifying suitable access control model and S&P policy language to enforce privacy on the agents and their clones. Since agents carry in their payload not only data, but also executable code, known solutions for safe code, such as signed code, code decoupling, runtime checking etc. need to be evaluated for suitability. Finally, the scalability of the proposed platform with increasing number of heterogeneous tags must be investigated.

References

1. Chen, R.S., Tu, M.A.: Development of an agent-based system for manufacturing control and coordination with ontology and rfid technology. Expert Syst. Appl. 36, 7581–7593 (2009), `http://portal.acm.org/citation.cfm?id=1508324.1508647`
2. Chow, H.K.H., Choy, K.L., Lee, W.B.: A dynamic logistics process knowledgebased system - an rfid multi-agent approach. Know.-Based Syst. 20(4), 357–372 (2007)
3. Dimitriou, T.: rfiddot: Rfid delegation and ownership transfer made simple. In: Proceedings of the 4th international conference on Security and privacy in communication networks, SecureComm '08, pp. 34:1–34:8. ACM, New York (2008) `http://doi.acm.org/10.1145/1460877.1460921`
4. Fischer-Hbner, S., Hedbom, H.: WP12: A holistic privacy framework for RFID applications: Future of IDentity in the information society. Deliverable D12.3, FIDIS consortium: Future of Identity in the Information Society (April 2008), `http://www.fidis.net/resources/deliverables/hightechid/d123-a-holistic-privacy-framework-for-rfid-applications/doc/13/`
5. Fouladgar, S., Afifi, H.: An efficient delegation and transfer of ownership protocol for RFID tags. In: First International EURASIP Workshop on RFID Technology, Vienna, Austria (September 2007)
6. ISO 8601:2004: Data elements and interchange formats – Information interchange – Representation of dates and times. ISO, Geneva, Switzerland (2004)
7. Jedermann, R., Behrens, C., Westphal, D., Lang, W.: Applying autonomous sensor systems in logistics–Combining sensor networks, RFIDs and software agents. Sensors and Actuators A: Physical 132(1), 370–375 (2006), `http://www.sciencedirect.com/science/article/B6THG-4JG5FBT-4/2/2dd9c816f409137409e604c48b68db05`
8. Koralalage, K.H., Reza, S.M., 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. In: Proceedings of the 2007 ACM symposium on Applied computing SAC'07, pp. 270–275. ACM, Seoul (2007), `http://doi.acm.org/10.1145/1244002.1244069`
9. Lim, C.H., Kwon, T.: Strong and robust RFID authentication enabling perfect ownership transfer. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 1–20. Springer, Heidelberg (2006)
10. Molnar, D., Soppera, A., Wagner, D.: Privacy for rfid through trusted computing. In: Proceedings of the 2005 ACM workshop on Privacy in the electronic society, WPES '05, pp. 31–34. ACM, New York (2005), `http://doi.acm.org/10.1145/1102199.1102206`

11. Molnar, D., Soppera, A., Wagner, D.: A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 276–290. Springer, Heidelberg (2006), http://www.springerlink.com/content/27344v3647u75803
12. OECD: Radio-Frequency identification (RFID): drivers, challenges and public policy considerations. Tech. Rep. DSTI/ICCP(2005)19/FINAL, Organisation for EconomicCo-operation and Development (OECD), Paris (March 2006), http://www.oecd.org/dataoecd/57/43/36323191.pdf
13. Osaka, K., Takagi, T., Yamazaki, K., Takahashi, O.: An efficient and secure RFID security method with ownership transfer. In: Wang, Y., Cheung, Y.-m., Liu, H. (eds.) CIS 2006. LNCS (LNAI), vol. 4456, pp. 778–787. Springer, Heidelberg (2007), http://portal.acm.org/citation.cfm?id=1417774&coll=&dl=
14. Rieback, M.R., Crispo, B., Tanenbaum, A.S.: Is your cat infected with a computer virus? In: PERCOM '06: Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications, March 2006, pp. 169–179. IEEE Computer Society Press, Pisa (2006)
15. Saito, J., Imamoto, K., Sakurai, K.: Reassignment scheme of an RFID tags key for owner transfer. In: Enokido, T., Yan, L., Xiao, B., Kim, D.Y., Dai, Y.-S., Yang, L.T. (eds.) EUC-WS 2005. LNCS, vol. 3823, pp. 1303–1312. Springer, Heidelberg (2005), http://dx.doi.org/10.1007/11596042_132
16. Song, B.: RFID tag ownership transfer. In: Conference on RFID Security. Budaperst, Hungary (July 2008)
17. Weis, S., Sarma, S., Rivest, R., Engels, D.: Security and privacy aspects of Low-Cost radio frequency identification systems. In: Hutter, D., Müller, G., Stephan, W., Ullmann, M. (eds.) Security in Pervasive Computing. LNCS, vol. 2802, pp. 454–469. Springer, Heidelberg (2004), http://www.springerlink.com/content/yvmfpkwc9nq6hqdw