A Simple Privacy Protecting Scheme Enabling Delegation and Ownership Transfer for RFID Tags

Sepideh Fouladgar, Hossam Afifi
Institut National des Télécommunication, Evry, FRANCE
Email: {sepideh.fouladgar, hossam.afifi}@int-edu.eu

***Abstract*— RFID (Radio frequency identification) technology raises many privacy concerns among which the potential tracking of an RFID tag bearer and the eventuality of an illegitimate reading device (reader) collecting information about him. To solve these issues, many RFID privacy protecting protocols assume that readers have continuous connectivity with a centralised on-line database in charge of the identification of a certain amount of tags. However such centralised models can raise scalability and latency problems. Moreover, they are not suitable in applications where connectivity is intermittent. As RFID tags may often change hands, it is also necessary to guarantee the privacy of a new tag owner. In this paper, we introduce a privacy protecting scheme based on pseudonyms that allows an on-line database to delegate temporarily and in a secure manner the capability to identify tags to selected readers. A reader which receives delegation for a given tag can identify this tag without referring to the on-line database, thus solving possible intermittent connectivity issues and making tag identification more scalable. Our protocol also manages tags ownership transfer without threatening the new owner's privacy.**

***Index Terms*— RFID, privacy, scalabilty, intermittent connectivity, time-limited delegation, ownership transfer**

I. Introduction

RFID technology enables automated identification of objects that are labelled with basic microchips called RFID tags. Thanks to their embedded antenna, tags are able to transmit over the air, information about the object they are attached to.

However, the wireless aspect of the technology raises privacy concerns. Actually, passive tags can broadcast information when powered and queried by a reader, without the tag owner being aware of this action. Most basic passive tags can even transmit a static serial number in response to a readers query, allowing tracking or inventorying of individuals [1]–[6].

A common solution to these privacy issues is to use a pseudonym scheme that relies on a trusted on-line database [7]–[9]. In this approach the tag replies with a freshly generated random value, each time it is queried, avoiding thus its bearer to be tracked. From its part, the on-line database decodes the pseudonyms broadcasted by the tags for authorised and authenticated readers.

However, on-line centralised schemes have many drawbacks in terms of scalability, latency and dependency on network connectivity. As the number of tags may be consequential and network connectivity disruptive, it is necessary to give to readers some controlled autonomy and enable off-line reading operations.

To overcome these limitations, some protocols delegate temporarily the ability to decode tags pseudonyms to selected authorised readers [10], [11]. Molnar, Soppera and Wagner [10], propose a scheme where readers can decode a tags pseudonyms within a certain window of reading operations. Soppera et al. [11], enhance the method described in [10] by distributing the on-line database functions on several local entities named RAT (RFID Acceptor Tag). They both suggest solutions for ownership transfer. Our protocol comes within this line of work, reducing the calculation burden on the tag.

In this paper, we present a new privacy-protecting protocol that enables selected readers to identify RFID tags temporarily without referring to the on-line database. We introduce two possible implementations of this protocol, one using a hash function and another based on a symmetric cryptographic function. This protocol is constituted of three distinct sub-protocols. In Delegation Request sub-protocol a reader demands to the on-line database, the ability to decode the pseudonyms of a tag or a group of tags. The Delegation sub-protocol describes the interactions between a delegated reader and the tags it has delegation for. In Delegation Update sub-protocol, the readers ability to identify tags is updated or if need be, revoked.

In addition, we present two methods for ownership transfer (when a tag changes hands) guaranteeing the privacy of the new owner of the tag. The first one, relies on a hash function and works under the assumption that both old and new owner trust the same on-line database. The second proposed protocol achieves complete ownership transfer, that is to say, the database trusted by the old owner does not maintain control on the tag and its secrets after ownership transfer. This latter solution also takes into account the eventuality of after sales services. The different protocols are then analysed from a security and performance point of view.

This paper is based on "A Simple Delegation Scheme for RFID Systems (SiDeS)," by S. Fouladgar, and H. Afifi, which appeared in the Proceedings of the IEEE International Conference on RFID, Grapevine, USA, March 2007. © 2007 IEEE.

II. Problem Statement

When a reader emits a query, the tags located in its read range, respond without alerting the tags owner. As a consequence, if a tag replies with a constant bit string (static identifier or even cryptographically protected identifier) the person bearing the tag broadcasts this value along its way, enabling clandestine readers to track him. Likewise, if a tag replies with a value that can be related to a particular item, thanks for example to an object naming service, clandestine readers will be able to harvest information about the person carrying the tag [1]–[6].

In order to solve these privacy issues, tags reply at each query with a pseudonym, i.e. a freshly generated random value encoded with a tag-specific secret. Therefore, as the pseudonym changes at each query, the tag cannot be tracked. In addition, only an authorised entity possessing the tag secret can identify it. Thus, clandestine information collection is no more possible. Usually in pseudonym models, tags share their secret key used to compute pseudonyms with a permanently on-line central database. If a reader possesses the appropriate rights and is connected to the on-line database, it can simply act as a relay passing pseudonyms to it. Then, the on-line database identifies the tag from the pseudonym it broadcasts, and replies to the reader with the tags identity.

The limitations of this on-line approach are clear. A centralised database is often in charge of a large number of tags and though must compute all the possible tag outputs until it finds a match. This can make scalability difficult. Moreover, each time a reader needs to identify a tag, it has to interact with the centralised database. In many applications, this reading latency can be disqualifying. Finally, if the database becomes unavailable for some reasons such as network connectivity failure, etc., all the reading operations of the tags relying on that database will be stopped.

Temporary delegation is a solution to these drawbacks. The idea of delegation is to enable readers to decode pseudonyms without referring to the on-line database. In fact, if a reader is authenticated and has delegation rights, the database not only gives the reader the tags identity but also the tag-specific secret used to create its pseudonyms, providing the ability to identify the tag to the reader. However, delegation must not be permanent since the delegated reader can be compromised. Moreover, one may not want to put unlimited trust on readers.

The privacy of a tag bearer must be guaranteed during the whole tags life. As the tag may change hands, the old owner should not be able to identify the tag. However, when the new owner buys a warranted tagged item, the old owner should be able to identify the tag in order to supply after sales services. Thus, tags ownership transfer raises other privacy issues for the tags new owner.

Our protocol introduces a new, secure and privacy protecting methods for temporary secret delegation to readers and RFID tags ownership transfer.

III. Preliminaries

A. Notations

Table 1 presents the notations used in this paper.

TABLE I.
Notations

D	On-line database
R	Reader
T	Tag
D_{old}	Database trusted by the tag's old owner
D_{new}	Database trusted by the tag's new owner
ID_T	Tag's identifier
N_x	Nonce generated by principal x
$Cred_x$	Principal x credentials
$H(.)$	Hash function H
f	Symmetric key cryptographic function
$f_K(v)$	Value v encoded with function f and key K
Kp	Pseudonym Key used to create pseudonyms
Ku	Update Key used to renew keys
Kp_{new}	Newly generated pseudonym key
Ku_{new}	Newly generated update key
δ	Random value generated by database D
C	Counter
C_{max}	Counter's maximum value
OT	Ownership transfer flag
$\|$	Concatenation
N	Number of tags D is in charge of
M	Number of tags delegated to R

B. Assumptions

In our protocol, we assume that a Tag T possesses two secret keys. One of the keys, Kp is used to compute pseudonyms. The other, Ku is used to update both keys Kp and Ku. T also embeds a counter incremented at each query.

Our protocol works under the assumption that T is passive and possesses a small re-writable memory to store Kp, Ku and the counter's value. T also needs to embed low cost cryptographic functions in order to create its pseudonyms and update its secrets. In the first considered design, T has a hash function, H, an XOR gate and a random number generator. In the second envisaged design, H is replaced by symmetric key cryptographic function f.

We assume that low cost practical implementations of cryptographic hash functions exist and are sufficiently secure and resistant to collision. Several research works confirm this hypothesis. In [1], Weis suggests different candidate paradigms for low cost RFID hash functions. In [12], Yüksel et al. propose several universal hash functions designed specifically for efficient hardware implementations and ultra-low power devices.

Similarly, we expect that low cost practical and secure implementations of cryptographic symmetric key functions exist. In fact, various research works propose AES (Advanced Encryption Standard) implementations specifically designed for ultra-low power devices [13]–[16]. In [15], authors propose an AES implementation that requires 3400 gates. Likewise, in [16] authors present an AES implementation supporting CBC(Cipher Block Chaining) mode and requiring 4K gates.

Finally, we consider that interactions between the database D and each reader R are performed over a suitable secure communications protocol.

C. Attackers model

To solve the security risks and privacy issues, we consider the following possible attacks against RFID tags, legitimate readers or the on-line database:

- **Replay attacks:** Attackers intercept a valid response emitted by a tag and retransmit it to a legitimate reader.
- **Man-in-the-middle attacks:** An attacker is able to insert or modify messages exchanged by legitimate principals without detection.
- **Eavesdropping:** Attackers listen passively to messages exchanged by legitimate principals and are able to decode them.
- **Denial-of-service (DoS):** The attacker disturbs or impedes communications between principals.

Our protocol, like many others, is not able to face jamming attacks. However, we try to prevent desynchronisation problems that can follow from these attacks. Moreover, we do not consider physical attacks against RFID tags since they are difficult to complete successfully in public or on a wide scale without detection.

D. Security Requirements

Our protocol should fulfil the following security requirements in order to guarantee the tag owner's privacy:

- **Anonymity:** An unauthorised reader should not be able to identify a tag from the pseudonyms it broadcasts.
- **Confidentiality:** Tag's messages should have no signification for illegitimate readers. They should not be able to deduce its private information (e.g. tag's secret key or identity) from its communications.
- **Integrity:** An attacker should not be able to modify surreptitiously messages exchanged between T, legitimate readers and on-line database D.
- **Authentication:** Mutual authentication between the tag and the on-line database, the database and the reader, and finally, a delegated reader and the tag should be provided in order to avoid man-in-the-middle or replay attacks.

IV. Protocol Design

This section presents the sequence of messages exchanged during delegation and ownership transfer between the various principals for both hash and symmetric key cryptographic function designs.

A. Setup

At setup, each tag T shares two secret keys Kp and Ku with the on-line database D. For each tag it manages, D stores these keys along with the tag's identifier ID_T. The tags counter C is initialised to zero and will be incremented at each readers query.

B. Delegation

When a reader R first meets a tag T, it needs to forward the pseudonym received from the tag to the on-line database D in order to identify the tag. As D shares with the tag the pseudonym key Kp, it can decode the tag's pseudonym. If the reader R possesses suitable credentials for tag T's identification, D gives R the tag's identifier ID_T.

To prevent limitations of permanently on-line systems, the idea is to delegate the ability to decode pseudonyms to selected readers by giving them key Kp. "Delegation request" sub-protocol is initiated when R asks D for the ability to decode tags pseudonyms on its own. If R has delegation rights, database D joins in its reply along with ID_T, the tags pseudonym key Kp. Once R is delegated for tag T, it is able to identify T , without referring to the database.
The sequence of messages exchanged for delegation request and delegation is illustrated in figure 1. We describe the detailed procedure for each step.

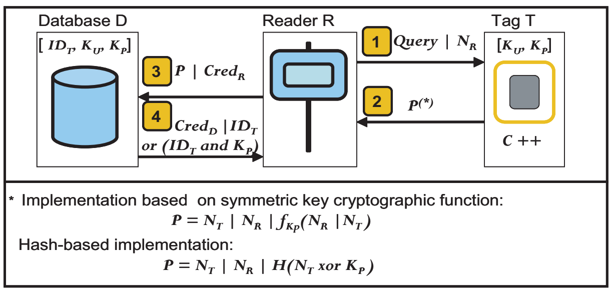

Database D
$[ID_T, K_U, K_P]$
Reader R
Tag T
$[K_U, K_P]$
1 $Query \mid N_R$
2 $P^{(*)}$
3 $P \mid Cred_R$
4 $Cred_D \mid ID_T$ or $(ID_T$ and $K_P)$
$C{++}$
* Implementation based on symmetric key cryptographic function:
$P = N_T \mid N_R \mid f_{Kp}(N_R \mid N_T)$
Hash-based implementation:
$P = N_T \mid N_R \mid H(N_T \text{ xor } K_P)$

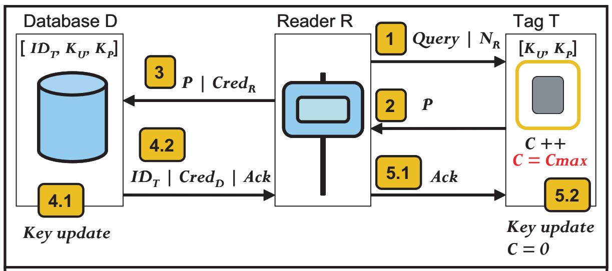
Figure 1. Delegation request and delegation

- **Step 1:** The reader R queries tag T joining to its demand, a freshly generated nonce N_R. This nonce enables R to prevent replay attacks from a fake tag.
- **Step 2:** In response to this query, T increments its counter C, generates a nonce N_T, and computes its pseudonym P. N_T ensures that the tag creates a fresh pseudonym at each query and protects the tag bearer against tracking. In the case of a hash function implementation, pseudonym P is $N_R|N_T|H(N_T \oplus Kp)$. For a symmetric key cryptographic function implementation, T encrypts the value $N_T|N_R$ with function f and key Kp. Pseudonym P is then, $N_R|N_T|f_{Kp}(N_T|N_R)$. The reader is unable to find out ID_T or the tag's secret keys from P since it does not know the tag's pseudonym key Kp.
- **Step 3:** R forwards P along with its credentials $Cred_R$.
- **Step 4:** If $Cred_R$ is not valid, the protocol ends. If the reader has the rights for tag identification, database D decodes the tag's pseudonym P by searching the space of all tag Kp keys it possesses and computing P until the calculated value matches the received pseudonym and returns ID_T. If the reader has delegation rights, the database joins Kp to its reply.

  Once the reader R is authenticated and granted delegation for a given tag T, it is able to decode Ts pseudonyms by itself. Consequently, it does not forward the tag pseudonym to D in step 3, but computes P for all the keys it possesses until it finds the matching key and the corresponding ID_T. Only two messages are exchanged to identify the tag.

C. Delegation update

As delegation should not be permanent, it is necessary to regularly update key Kp in order to end or update the reader's delegation status. For this purpose, T embeds a counter which is incremented at each query. When the counter reaches its maximum value C_{max}, keys are updated thanks to key Ku. This mechanism permits to limit readers delegation to a number of C_{max} queries for a given tag. The sequence of messages exchanged for key update is illustrated in figure 2. We describe the detailed procedure for each step.



Implementation based on symmetric key cryptographic function:

$P = N_T | N_R | f_{Ku}(N_R | N_T)$, $Ack = N_T | f_{Ku}(N_T | \delta)$

Key update: $K_{Pnew} = K_P \text{ xor } N_T \text{ xor } \delta$, $K_{Unew} = K_U \text{ xor } N_T \text{ xor } \delta$

Hash-based implementation:

$P = N_T | N_R | H(N_T \text{ xor } K_U)$, $Ack = H(\delta) | (\delta \text{ xor } H(K_U))$

Key update: $K_{Pnew} = H(K_P \text{ xor } \delta)$, $K_{Unew} = H(K_U \text{ xor } \delta)$

Figure 2. Delegation update

- **Step 1:** R generates a fresh nonce N_R and queries its surrounding tags.
- **Step 2:** T increments its counter C which reaches the maximum value C_{max}. Then, T generates a fresh nonce N_T and computes a new pseudonym P using key Ku instead of Kp. In the case of a hash function implementation, P is then $N_R|N_T|H(N_T \oplus Ku)$. For a symmetric key cryptographic function implementation, P becomes $N_R|N_T|f_{Ku}(N_T|N_R)$.
- **Step 3:** When the reader receives this pseudonym, it is unable to decode it because R only knows key Kp. As a consequence R forwards the tags pseudonym along with its credentials $Cred_R$ to database D.
- **Step 4.1:** If $Cred_R$ is not valid, the protocol ends. If the reader has the rights for tag identification, database D identifies the tag thanks to key Ku. Then D generates a random value δ and updates Ku and Kp while keeping their old values. In the case of a hash function implementation, Kp and Ku are updated as follows: $Kp_{new} = H(Kp \oplus \delta)$ and $Ku_{new} = H(Ku \oplus \delta)$. For a symmetric key cryptographic function implementation, $Ku_{new} = Ku \oplus N_T \oplus \delta$ and $Kpnew = Kp \oplus N_T \oplus \delta$.
- **Step 4.2:** D replies to R with the tags identity and an acknowledgement of key update which is $H(\delta)|\delta \oplus H(Ku)$ if it is a hash-based implementation or $N_T|f_{Ku}(N_T|\delta)$ if it is a symmetric key cryptographic function implementation.
- **Step 5.1:**R forwards the received message to the tag.
- **Step 5.2:** T decodes the received message with Ku and gets δ. Like database D, T updates Ku and Kp. Once the tag has updated its keys, previously delegated readers have to refer to the database in order to decode tags pseudonyms or to extend their delegation rights.

D. Ownership Transfer

When a tagged object passes from one owner to another, tags attached to this object are no longer the old owner's property. In order to ensure its privacy, the new owner has to revoke the ability of the old owner and any previous reader to access tag information. This section presents two ownership transfer protocols. The first one, relies on a hash function and works under the assumption that both old and new owner trust the same on-line database. The second proposed protocol is based on symmetric key cryptography and achieves complete ownership transfer, independently of the trusted database. Figure 3 and 4 respectively illustrate the sequence of messages exchanged in these two protocols.

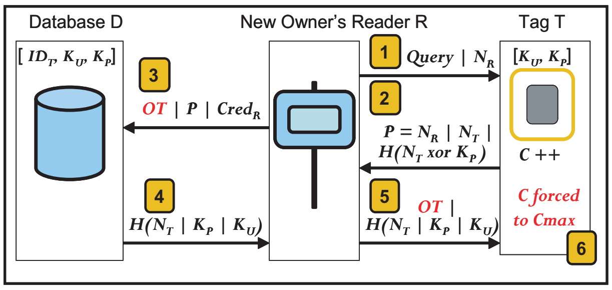


Figure 3. Hash based ownership transfer

1) Hash based implementation:

- **Step 1:** The new owner's reader R interrogates tag T joining a nonce value N_R.
- **Step 2:** In response, T increments its counter C, generates a nonce N_T and computes a new pseudonym $H(N_T \oplus Kp)$.
- **Step 3:** R forwards the tag's reply to D along with an ownership transfer flag OT and its credentials.
- **Step 4:** If R's credentials are valid,D returns the value $H(N_T|Kp|Ku)$.
- **Step** 5: R transmits this value to T along with the ownership transfer flag OT .
- **Step 6:** Tag T computes $H(N_T|Kp|Ku)$ from his side. If the two hash values match, then T forces its counter to C_{max}, initiating the update subprotocol. Kp and Ku authenticate D to the tag and N_T protects T against replay attacks.

In this protocol, we assume that both old and new tag owners trust the same database D. If the new owner does not trust the database D_{old} trusted by the old owner, the tag keys should be changed without D_{old} knowing the new values of tag keys. For example, if a customer buys an item from a retailer and wants to use the RFID tag attached to this item in its smart home system, the retailers database should not be able to identify the tags when used in the customers home.The symmetric key implementation deals with this issue.

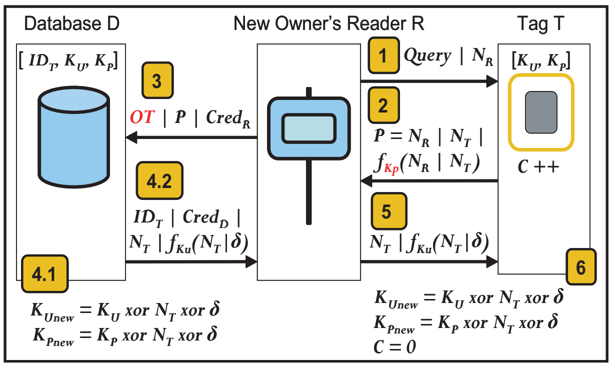

Database D
New Owner's Reader R
Tag T
[ID_T, K_U, K_P]
3
OT | P | $Cred_R$
4.2
ID_T | $Cred_D$ | N_T | $f_{Ku}(N_T|\delta)$
4.1
$K_{Unew} = K_U$ xor N_T xor δ
$K_{Pnew} = K_P$ xor N_T xor δ
1 $Query$ | N_R
2
$P = N_R$ | N_T | $f_{Kp}(N_R$ | $N_T)$
5
N_T | $f_{Ku}(N_T|\delta)$
[K_U, K_P]
C ++
6
$K_{Unew} = K_U$ xor N_T xor δ
$K_{Pnew} = K_P$ xor N_T xor δ
$C = 0$

Figure 4. Ownership transfer based on a symmetric key cryptographic function

2) Symmetric key cryptographic function implementation: If both old and new owners trust the same on-line database D, when the new owners reader forwards to D the tags pseudonym and asks for tags ownership transfer (step 3), D generates a random value δ, updates keys for this tag with δ and N_T (step 4), encodes δ with f and old key Ku, before transferring it to the tag (step 5). When T gets δ, it updates its keys (step 6). Then, the old owners reader is unable to identify T without referring to database D.

If the new owner relies on its own back-end database D_{new}, the database of the previous owner D_{old} must transfer the current tag key values to the D_{new} (e.g. the database in charge of a customer smart home readers). D_{old} may keep these key values if the item is warranted for after sales services. When a reader of the customers smart home forwards to D_{new} the tags pseudonym and asks for tags ownership transfer (step 3), D_{new} generates a random value δ, updates keys for this tag with δ and N_T (step 4), encodes δ with f and old key Ku, before transferring it to the tag (step 5). When T gets δ, it updates its keys (step 6).

D_{new} and T update tag keys thanks to δ and the random value N_T (first generated by the tag to compute the current pseudonym). N_T is used to compute new keys because this random value can be received only by readers located in the tag emission range. Once the key values changed, the retailers reader and D_{old} are not able to identify T anymore. If the customer needs to return a warranted item to the retailer, D_{new} can compute a δ value forcing the tag to change its keys to the old key values memorized by D_{old}. Thus, the retailer will once again be able to identify the item and ensure after sales services.

V. Security Analysis

In this section we evaluate our protocol in the view point of the security requirements presented above.

A. Normal operation

In order to thwart malicious traceability, the tag replies, each time it is queried by readers, with a fresh pseudonym ($H(N_T \oplus K)$ or $f_K(N_T|N_R)$) where K is Kp for delegation and ownership transfer and Ku for delegation update. N_T which is a random number value generated at each query, guarantees the freshness of the pseudonym while shared keys restrict the ability to identify the tag to authorised principals.

To ensure confidentiality and integrity, sensitive data is hashed or encoded with a symmetric key cryptographic function since illegitimate readers should not be able to determine secret keys or ID_T from exchanged messages and consequently attempts to alterate communications can be detected. Moreover, using a symmetric cryptographic function to encode pseudonyms (e.g AES algorithm), makes very difficult the compromise of the shared secrets from the cryptanalysis of exchanged messages.

In hash-based ownership transfer protocol, when the new owners reader asks for ownership transfer, D returns $H(N_T|Kp|Ku)$. As Ku is only known by D and T , it authenticates D to the tag. N_T ensures the freshness of the message. The hash function guarantees its confidentiality and integrity.

Mutual authentication is achieved between the tag and the backend database thanks to the shared secrets Kp and Ku. Kp also enables authentication between the tag and a delegated reader. The backend database and the readers authenticate each other thanks to specific credentials (e.g certificates).

In order to detect replay attacks from a fake tag broadcasting a pseudonym previously generated by a legitimate tag, a random value N_R is generated by the reader, each time it queries the tag.

B. Abnormal operations and attacks

In this subsection we explain the consequences of jamming attacks, message loss and other abnormal operations in our protocol.

The loss or blocking of the readers request and tag's reply messages is a denial-of-service attack preventing tag identification. This kind of attack is not inherent to the proposed scheme but an issue in any wireless system. However, these attacks cannot remain undetected for a long time. Detection of jamming attacks and protection from these attacks are out of the scope of this paper.

In the case of delegation update sub-protocol, messages of step 2 or 5.1 may be lost, intercepted or blocked. Consequently, tag T does not change its keys. While T has not received δ from the back-end database D, it keeps on sending pseudonym $H(N_T \oplus Ku)$ or $f_{Ku}(N_T|N_R)$ without incrementing its counter until a reader transmits successfully step 2 or 5.1 messages. As D keeps the old key values, T can still be identified. This procedure ensures the synchronisation between D and T .

Similarly in the case of ownership transfer, if message 5 is lost, the tag remains with keys known by the old owner until message 5 is received by the tag. However, as noted above, this kind of attacks can be detected. Thus, ownership transfer may be achieved in a protected area, far from the jamming area of the attacking device.

Another feasible attack is that of an illegitimate reader incrementing the tags counter through rapid-fire interrogation, until it reaches cmax. This kind of DoS attack can be detected if delegation update is frequently requested for a given tag.

VI. Performance analysis

A. Computational aspect

We assume that most of the time readers are delegated, i.e. delegation sub-protocol is the usual and commonly employed procedure. This subsection presents the number of computations made by the different entities during the various stages of our protocol.

1) Delegation and delegation request sub-protocols: To create a new pseudonym, T requires a nonce generation, a symmetric encryption operation or a hash calculation and an XOR computation . T also needs to increment its counter.

When readers are delegated, the back-end database D makes no computations. On the other hand, when a reader first asks for delegation it requires an average of N calculations (N times one symmetric encryption or one hash and one XOR) to identify the tag from the pseudonym forwarded by the reader, where N is the number of tags relying on D. The maximum number of operation is $2N$. In fact, when D receives a delegation request from a familiar reader R, it first considers that R asks for delegation update as it may occur more frequently than delegation request. It then searches the space of all Ku keys to identify the tag, before searching the space of all Kp keys. A solution to reduce the maximum complexity of this search (from $2N$ to N) can be to add a flag to the tags response in order to distinguish delegation request and delegation update. This complexity can also be globally reduced if D keeps for each reader an updated table of the tags which are in its corresponding read range.

If a it is delegated, it takes an average $M/2$ calculations for a reader to decode the pseudonym, where M is the number of tags the reader has delegation for. We assume that M is much smaller than N.

2) Update delegation sub-protocol: Similarly to delegation and delegation request sub-protocols, T requires a nonce generation, a symmetric encryption operation or a hash calculation and an XOR computation for pseudonym generation. T also makes a counter incrementation. To check D's acknowledgement and update its keys, T executes two hash calculations and two XOR operations for hash-based implementation. In the case of a symmetric-key implementation, the tag needs to perform a symmetric decryption operation and two XOR operations to update its keys.

For the reader, when the tag delegation comes to an end and the pseudonym is encoded with key Ku, it takes an average of $M/2$ calculations for R to find out that it cannot decode the tags pseudonym.

In a hash-based implementation, D requires a random number generation, two hash and two XOR operations, to update keys Ks and Ku. In the case of a symmetric-key implementation, D requires a random number generation, two XOR operations, to update keys Kp and Ku,and a symmetric key encryption to send δ to the tag.

3) Ownership transfer protocols: In hash-based ownership transfer protocol T requires one additional hash function to check D's reply, $H(N_T|Ks|Ku)$. The database needs one additional hash function to generate its reply $H(N_T|Ks|Ku)$.

B. Storage

In low cost RFID applications the size of tags memory is critical compared to other principals memory. In our protocol, the tag needs to store two 128-bit secret keys Kp and Ku (in hash-based implementation, key size can be of 64-bit) and holds a 20-bit counter. It needs also some buffer memory for cryptographic operations.

C. Communications

One of the advantages of delegation protocols is that they minimise the number of messages exchanged between tags and the related back-end database. Communications are mainly done between tags and readers and only two messages are exchanged. Four to Five messages are exchanged for key update and ownership transfer, which is mostly acceptable.

VII. Related Work

Few research works has been done to delegate to readers the ability to identify tags. The proposed protocols try to reduce the burden on the central online database often used in privacy protecting schemes, by delivering to authenticated readers, temporary secrets that enable tag identification. These solutions are usually combined with ownership transfer protocols as transitory secrets may simplify the transfer of a tag from one owner to another.

Other solutions only consider ownership transfer issue. However in these approaches, ownership transfer is often incomplete in the sense that the back-end database still possesses the tag secrets even if the new owner does not trust this database and relies on its own personal database. Moreover, the eventuality to return a tagged object for after sales services was never dealt with to our knowledge.

In [10], authors propose a delegation protocol based on a tree of secrets of depth $d = d_1 + d_2$. In this scheme, each node except the root, is associated with a k-bits secret. Secrets until depth d_1 are chosen by random, those from depth d_1 to depth d can be derived from the ascendant node. To each node at depth d_1 corresponds a tag, thus tags are roots of sub-trees of depth d_2. The leaves of a tag sub-tree stand for the counter values of the tag. A tag receives all the keys of the path from the tree root to its node. Therefore each tag needs to store d_1 secrets. The next d_2 levels of the tree contain secrets that can be derived from ascendant nodes by applying a pseudorandom generator (PRG). When queried, the tag responds by encoding a random value with successively all of the d secret keys of the path. The tag will use each time a different path of secrets since its counter is incremented at each query. If a reader is not delegated, it forwards the pseudonym received from the tag to the central database. This latter possesses the whole tree of secrets. The first d_1 encoded random values of the pseudonym allow the database to identify the tag. If the readers credentials are valid, the trusted center transmits to the reader the decoded identity. If a reader has the associated credentials, the trusted center can decide to delegate the ability to decode pseudonyms to the reader and thus delivers to this latter, a set of nodes under depth d_1, corresponding to a given interval of leaves. Thanks to the last d_2 encoded random values of the tags pseudonyms and the sub-trees of secrets arising from these nodes, the delegated reader will be able to recognise tags which counter values are situated in this interval of leaves, while they are situated in this interval. However, the reader will have to make a brute force search among keys it possesses, in order to identify the tags. In [10], thanks to the tree architecture of secrets, the trusted center can make a depth first search and cut the branches it does not need to explore. Compared to brute force, tree architecture decreases greatly tag identification complexity, reducing it to a maximum of $O(blog_bN)$ where b is the branching factor. However, as mentioned above we can reduce the key search complexity of our protocol by limiting it to the tags of a reader which are in its corresponding read range. With this mechanism, it becomes possible to reach a maximum complexity of $O(M)$ for tag identification. The main inconvenient of [10] comes from the fact that the tree structure implies intersections between the key sets, thus compromise of secrets in one tag can lead to compromise of secrets in other tags. Avoine, Dysli and Oechslin analyse the consequences of this drawback in [17] and show that for some acceptable security requirements, the branching factor b of the tree of secrets should be important and the number of tags delegated to a reader should be limited. This problem is not raised in our protocol since all tag secrets are independent. Compared to [10], another benefit of our protocol is the reduction of computations in the delegation sub-protocol. In fact in [10], the tag not only makes d_2 PRG operations to derive the last d_2 secrets but also executes d pseudo-random function computations in order to generate a pseudonym. The reader in his side, has to do the same amount of calculations for each tag it has delegation for. Soppera et al. [11] enhance the scheme proposed in [10], by introducing a local physical powered device, named RAT (RFID Acceptor tag) that performs delegation of secrets and ownership transfer similarly to and in place of the trusted center, though the two solutions can work together. In order to associate their tags to a local RAT, tag owners should authenticate against credentials stored in the RAT, by the RAT owner. Once authenticated, tag owners can load credentials for further readers.

In [18], authors supplement their authentication protocol with ownership transfer. In this solution, the new owner uses a mobile reader to get the tag's secrets from the back-end database through the check-out reader. Then, when the mobile reader reads the tag, it changes its secrets staying outside the communication range of the check-out reader in order to prevent this latter from being able to compute the new secrets. This solution does not consider discontinuous connectivity nor after sales services. Moreover the authentication part of the protocol requires a great amount of computations and storage capacities from the tag and the back-end database. In fact the tag needs to embed three pseudo-random functions and the database has to compute many tag's identification data and key chains.

In [19], authors achieve ownership transfer by changing the key used by the tag to compute its pseudonyms. In this solution the tag embeds a hash function, an XOR gate and an encrypted value $E_K(ID)$ which is the tag's identifier ID, encrypted with a symmetric key K generated by the tag's owner. In order to prevent invasion of its own privacy the new owner broadcasts a new symmetric key K' to the database. The next time a reader queries the tag, the database forces the tag to change saved data $E_K(ID)$ to $E_{K'}(ID)$. However it does not consider discontinuous connectivity or after sales services.

VIII. Conclusion

In this paper we propose a new lightweight and scalable protocol for delegation and ownership transfer. It fulfils the main security requirements among which tag privacy and confidentiality of tags private information (e.g. tags secret keys or identity), independently of the number of tags delegated to a reader. Messages exchanged between legitimate principals cannot be modify surreptitiously and mutual authentication is achieved preventing man-in-themiddle or replay attacks.

Our protocol allows to strongly reduce interactions between tags and the on-line databases in pseudonym based privacy protecting schemes, enabling off-line reading operations. As low cost practical and secure implementations of hash and cryptographic symmetric key functions exist, its implementation remains sufficiently low cost.The tag only needs a hash function or symmetric cryptographic function, a nonce generator and a small memory to store the two shared key and the delegation counter.

REFERENCES

[1] S. A. Weis, ”Security and Privacy in Radio-Frequency Identification,” Master’s thesis, 2003

[2] S. A. Weis, S. Sarma, R. L. Rivest and D. W. Engels, ”Security and privacy aspects of low-cost radio frequency identification systems,” *in proceedings of IEEE PerCom’03*, March 2003

[3] D. Molnar, and D. Wagner, ”Privacy and Security in library RFID: Issues, practices, and architectures,” *in proceedings of ACM Conference on Communications an Computer Security*, 2004

[4] G. Avoine and P. Oeschlin, ”RFID traceability: A multilayer problem,”*Financial Cryptography 2005*, LNCS 3570, Springer-Verlag, pp.125-140, 2005

[5] A. Juels, S. Garfinkel and R. Pappu, ”RFID privacy: An overview of problems and proposed solutions,” *in proceedings of IEEE Security an Privacy*, 2005

[6] A. Juels, ”RFID Security and privacy: A research survey,” *IEEE JSAC*, vol. 24, no. 2, pp. 381-394, February 2006

[7] D. Henrici and P. Muller, ”Hash-based Enhancement of Location Privacy for Radio Frequency Identification Devices using Varying Identifiers,” *in proceedings of IEEE PerSec’04*, March 2004

[8] M. Ohkubo, K. Suzuki and S. Kinoshita, ”Cryptographic Approach to Privacy-Friendly Tags,” *in proceedings of RFID Privacy Workshop*, 2003

[9] G. Tsudik, ”YA-TRAP: Yet Another Trivial RFID Authentication Protocol,” *in proceedings of IEEE PerCom’06*, March 2006

[10] D. Molnar, A. Soppera and D. Wagner, ”A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags,” *in proceedings of SAC’05*, August 2005

[11] A. Soppera and T. Burbridge, ”Secure by default: The RFID Acceptor Tag (RAT),” *in proceedings of RFIDSec’06*, July 2006

[12] K. Yüksel, J.P. Kaps and B. Sunar, ”Universal Hashing for Ultra-Low-Power Cryptographic Hardware Applications,” *in proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, January 2004

[13] M. Feldhofer, S. Dominikus and J. Wolkerstorfer and V. Rijmen, ”Strong authentication for RFID systems using the AES algorithm,” *in proceedings of CHES’04*, August 2004

[14] M. Aigner and M. Feldhofer, ”Secure Symmetric Authentication for RFID Tags,” i*n proceedings of TCMC’05*, March 2005

[15] M. Feldhofer, J. Wolkerstorfer and V. Rijmen, ”AES Implementation on a grain of sand,” *Information Security, IEE proceedings*, vol. 152, no. 1, pp. 13-20, October 2005

[16] P. Kaps and B. Sunar, ”Energy comparison of AES and SHA-1 for ubiquitous computing,” *in proceedings of EUC’06*, December 2006

[17] G. Avoine, E. Dysli and P. Oeschlin, ”Reducing Time Complexity in RFID Systems,” *in proceedings of SAC’05*, August 2005

[18] C. H. Lim and T. Kwon, ”Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer,” *in proceedings of ICICS’06*, December 2006

[19] K. Osaka, T. Takagi, K. Yamazaki and O. Takahashi, ”An Efficient and Secure RFID Security Method with Ownership Transfer,” *in proceedings of CIS’06*, November 2006

Sepideh Fouladgar is currently a Ph.D. candidate at the National Institute of Telecommunications (Institut National des Télécommunications), Evry, France. She received her MS in network and computer science from the Pierre et Marie Curie University of Paris, France, in 2004. Her research interests include security, low power devices (RFID systems and sensor networks) and cryptography.

Hossam Afifi received his Ph.D. degree in Computer Science from INRIA, Sophia Antipolis. In 1993, he pursued research work at the Washington University in St. Louis and joined ENST-Bretagne at the end of 1993 and worked on computer and telecommunications networks, Internet and public cellular networks. He joined the Institut National des Télécommunications in 2000 where as a Professor he conducts research on applications and services for wireless networks and wireless LANs. His activities include security mechanisms and protocols for applications and services access, mobility management combined with security paradigms and secure communications based on smart cards and networks intelligence distribution. He also holds an adjunct Research Position at INRIA. Prof. Afifi is a member of the IEEE Computer Society.