An Efficient Delegation and Transfer of Ownership Protocol for RFID tags

Sepideh Fouladgar
Institut National des télécommunications
Evry, France
sepideh.fouladgar@int-edu.eu

Hossam Afifi
Institut National des télécommunications
Evry, France
hossam.afifi@int-edu.eu

***Abstract* - RFID technology raises many privacy concerns among which the potential tracking of an RFID tag bearer and the possibility of collecting information about him. 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 new protocol based on pseudonyms that solves these privacy issues while enabling tag identification without the need of a continuous connectivity between reading devices (readers) and a centralised online database.

I. Introduction

RFID technology aims at identifying objects in an automated fashion. For this purpose, objects 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. The main privacy concerns on RFID technology are due to its wireless aspect. In fact, 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 [1–3].

A common solution to these privacy issues is to use a pseudonym scheme relying on a trusted on-line database [4–6]. The role of this database is to decode pseudonyms broadcasted by tags, for authorised readers. However, on-line centralized schemes have many drawbacks in terms of scalability, latency and dependency on network connectivity. To overcome these limitations, other protocols delegate temporarily the ability to decode tags pseudonyms to selected authorised readers [7–9].

In this paper, we present a new protocol that simplifies and enhances the security of the delegation protocol proposed in [9]. It also proposes a simple and efficient method for ownership transfer (when a tag changes hands) guaranteeing the privacy of the new owner of the tag. Our protocol is then analysed from a security and performance point of view.

II. Problem Statement

When a reader emits a query, the tags located in its read range, respond. As a consequence, if a tag replies with a constant bit string, 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 [1–3].

In order to solve these issues, tags can respond 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 reader possessing the tag secret can identify it. Thus, clandestine information collection is no more possible.

Usually in pseudonym models, tags share their specific secret keys (used to compute pseudonyms), with a permanently on-line central database. If a reader possesses the appropriate rights, the on-line database identifies the tag from the pseudonym it broadcasts, and replies to the reader with the tag's 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. However it 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 tag's 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 tag's new owner.

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

III. Preliminaries

3.1 Notations

Table 1 presents the notations used in this paper.

Table 1 - 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
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
$\vert$	Concatenation

3.2 Assumptions and Attacking model

Our protocol works under the assumption that tag T has a symmetric key cryptographic function f, an XOR gate and a random number generator in order to create its pseudonyms and update its secrets. In fact, the tag 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.

In our design, we assume that T is passive and possesses a small re-writable memory to store Kp, Ku and the counter's value.

We expect low cost practical and secure implementations of cryptographic symmetric key functions to exist. In fact, various research works propose AES implementations specifically designed for ultra-low power devices. In [10], authors propose an AES implementation that requires 3400 gates. Likewise, in [11] authors present an AES implementation supporting CBC mode and requiring 4K gates.We also assume that interactions between the database D and each reader R are performed over a suitable secure communications protocol.

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.

3.3 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 legitimate principals.

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

In this section, we describe the sequence of messages exchanged during delegation and ownership transfer between the various principals.

4.1 Initial Setup

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

4.2 Delegation

When a reader R first meets a tag T, it forwards the tag's pseudonym to the on-line database D in order to identify it. As D possesses key Kp used to create pseudonyms, it is able to recognise T from the data it broadcasts. If the reader possesses the suitable credentials for T's identification, D replies with T's identity ID_T.

To prevent limitations due to the on-line database, the idea is to delegate the ability to decode pseudonyms to selected readers by giving them key Kp. If a reader R possesses the convenient credentials for delegation, database D joins in its reply, the tag's key Kp along with ID_T. Once R is delegated for tag T, it is able to identify T on its own, without referring to the database.

The sequence of messages exchanged for delegation is illustrated in figure 1. We describe the detailed procedure for each step.

Step 1: Reader R generates a nonce N_R to prevent replay attacks and queries its surrounding tags.

Step 2: In response to this query, tag T first increments its counter C. Then, T creates a new pseudonym by generating a fresh nonce N_T and encrypting $N_T \oplus N_R$ with key Kp and cryptographic function f. N_T ensures that the tag creates a fresh pseudonym at each query and protects the tag bearer against tracking.

Step 3: R forwards T's pseudonym 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 has to identify the tag. To do so, D computes $f_{Kp}(N_T \oplus N_R)$ searching the space of all tag Kp keys it possesses until it matches the received pseudonym. If the reader has delegation rights, the database replies with both tag's key Kp and ID_T. Otherwise, D only returns ID_T. Once the reader R is authenticated and granted delegation for a given tag T, it is able to decode T's pseudonyms on its own. Thus it does not forward the tag pseudonym to D in step 3, but it computes $f_{Kp}(N_T \oplus N_R)$ for all the keys it possesses until it finds the right key and the corresponding ID_T. Only two messages are exchanged.

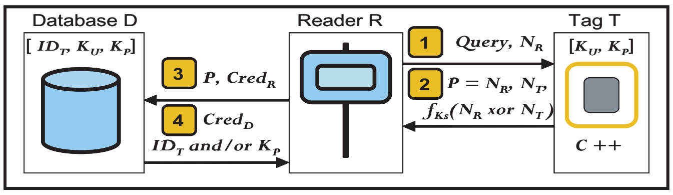

FIGURE 1 - DELEGATION

However, as delegation should not be permanent, it is necessary to regularly update key Kp to end R's delegation. For this purpose, T also 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.

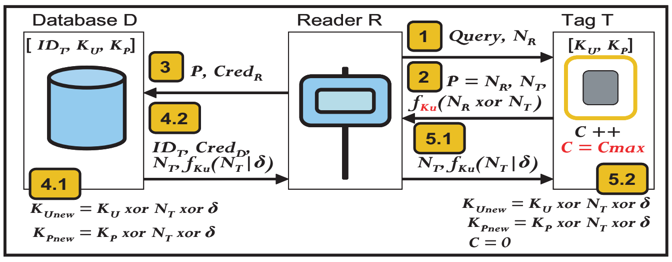
Database D
Reader R
Tag T
[ID_T, K_U, K_P]
1
Query, N_R
[K_U, K_P]
3
P, $Cred_R$
2
$P = N_R, N_T,$
$f_{Ku}(N_R$ xor $N_T)$
4.2
ID_T, $Cred_D$,
N_T, $f_{Ku}(N_T | \delta)$
5.1
N_T, $f_{Ku}(N_T | \delta)$
C ++
C = Cmax
5.2
4.1
$K_{Unew} = K_U$ xor N_T xor δ
$K_{Pnew} = K_P$ xor N_T xor δ
$K_{Unew} = K_U$ xor N_T xor δ
$K_{Pnew} = K_P$ xor N_T xor δ
C = 0

FIGURE 2 - DELEGATION UPDATE

Step 1: Reader R generates N_R and queries its surrounding tags.

Step 2: T increments its counter C but this latter reaches its maximum value C_{max}. Then, T creates a new pseudonym by generating a fresh nonce N_T and encrypting $N_T \oplus N_R$ with cryptographic function f however using this time key Ku.

Step 3: Even a delegated reader is not able to decode this pseudonym because it only knows key Kp. As a consequence R forwards the tag's 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 it generates a random value δ and updates Ku and Kp where $Ku_{new} = Ku \oplus N_T \oplus \delta$ and $Kp_{new} = Kp \oplus N_T \oplus \delta$ while keeping old values of Kp and Ku.

Step 4.2: D replies to R with the tag's identity and $N_T, f_{Ku}(N_T|\delta)$, using the Ku's old value.

Step 5.1: R forwards the received message to the tag.

Step 5.2: T decodes the received message with Ku and gets δ. T updates Ku and Kp in the same way than database D. Once the tag has updated its keys, previously delegated readers have to refer to the database in order to decode tag's pseudonyms or to extend their delegation rights.

4.3 Ownership transfer

A very similar method is used to change the tag's keys when T passes from an old owner to a new owner. The sequence of messages exchanged for delegation is illustrated in figure 3.

If both old and new owners trust the same on-line database D, when the new owner's reader forwards to D the tag's pseudonym and asks for tag's ownership transfer (step 3), D generates a random value δ, updates keys for this tag with δ and N_T (step 4.1), encodes δ with f and old key Ku, before transferring it to the tag (step 4.2). When T gets δ, it updates its keys (step 5.2). Then, the old owner's reader is unable to identify T without referring to 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 retailer's database should not be able to identify the tags when used in the customer's home.

In our protocol, when the customer buys a tagged item, the retailer's database D_{old} must transfer the current tag key values to the database D_{new} trusted by the customer (e.g. the database in charge of the 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 customer' smart home forwards to D_{new} the tag's pseudonym and asks for tag's ownership transfer (step 3), D_{new} generates a random value δ, updates keys for this tag with δ and N_T (step 4.1), encodes δ with f and old key Ku, before transferring it to the tag (step 4.2). When T gets δ, it updates its keys (step 5.2). D_{new} and T use δ and the random value N_T (first generated by the tag to compute the current pseudonym), in order to update the tag keys. We use N_T 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 retailer's 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.

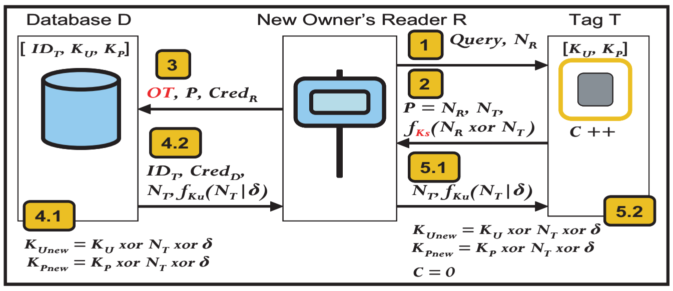
Database D
New Owner's Reader R
Tag T
[ID_T, K_U, K_P]
1
Query, N_R
[K_U, K_P]
3
OT, P, $Cred_R$
2
$P = N_R, N_T,$
$f_{Ks}(N_R$ xor $N_T)$
C ++
4.2
ID_T, $Cred_D$,
$N_T, f_{Ku}(N_T|\delta)$
5.1
$N_T, f_{Ku}(N_T|\delta)$
4.1
5.2
$K_{Unew} = K_U$ xor N_T xor δ
$K_{Pnew} = K_P$ xor N_T xor δ
$K_{Unew} = K_U$ xor N_T xor δ
$K_{Pnew} = K_P$ xor N_T xor δ
$C = 0$

FIGURE 3 - OWNERSHIP TRANSFER

V. SECURITY ANALYSIS

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

5.1 Normal operation

To thwart malicious traceability and ensure its anonymity, the tag replies each time it is queried by a reader, with a fresh pseudonym $f_K(N_T \oplus N_R)$ where K is Kp for delegation and ownership transfer and Ku for delegation update. For this purpose, it generates at each reader's query a fresh random number N_T and encodes it thanks to the shared key only known by the tag and principals which have the right to identify the tag.

Random value N_R generated by the reader when it queries a tag, enables to detect a replay attack from a fake tag broadcasting a pseudonym previously generated by a legitimate tag.

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

In the case of delegation update and ownership transfer, the secret key Ku shared by the tag and the back-end database protects the confidentiality of the value δ used to update the keys.

Finally, 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.

5.2 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 tags 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, 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 generates pseudonym $f_{Ku}(N_T \oplus N_R)$ and keeps on sending it 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.

Similarly in the case of ownership transfer, if message 5.1 is lost, the tag remains with keys known by the old owner until message 5.1 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

6.1 Computational aspect

In order to create a new pseudonym, the tag requires a nonce generation, a symmetric encryption operation and an XOR computation. T also needs to increment its counter. When it receives a message from the reader, the tag needs to perform a symmetric decryption operation and two XOR operations to update its keys.

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 a maximum of 2N pseudonym calculations to identify the tag, where N is the number of tags relying on D. 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. To update keys Kp and Ku, D requires a random number generation, two XOR operations and a symmetric key encryption to send δ to the tag.

When a reader R is delegated, it takes M calculations for the 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. When the tag delegation comes to an end and the pseudonym is encoded with key Ku, it takes M calculations for R to find out that it cannot decode the tag's pseudonym.

In terms of memory, the tag needs to store two 128-bit secret keys Kp and Ku and holds a 20-bit counter. It needs also some buffer memory for cryptographic operations.

In terms of 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. Five messages are exchanged for key update and ownership transfer, which is mostly acceptable.

VII. Related Work

Few research works has been done to ensure privacy of the tag owners when it changes hand. Among these solutions, many combine delegation protocols and ownership transfer. 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 [12], 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.

Molnar et al. [7], propose a delegation protocol based on a tree of secrets that enables ownership transfer. In this scheme, each node except the root is associated to a secret. Readers that are given a node of the tree of secrets are then able to derive keys corresponding to descendant nodes and decode the tag's pseudonyms computed with these keys autonomously within a certain window of reading operations. The number of reading operations is set to the maximum value C_{max} of a counter C. In order to achieve ownership transfer, the new owner can read the tag until it reaches C_{max}. It can also increase the tag counter's value by performing mutual authentication thanks to the secrets shared with the tag. While reducing the workload for the back-end database this protocol raises key compromise problems due to intersections between key sets [13]. Moreover the tag needs to make an important amount of computations in order to derive secrets and encode pseudonyms in the delegation sub-protocol.

Soppera et al. [8], enhance the scheme proposed in [7] by introducing a local physical device that performs delegation and ownership transfer similarly to and in place of the back-end database.

In [9], authors introduce a delegation scheme where the database and the tag share two keys. One key is used to create pseudonyms and the other to update secrets. It also enables ownership transfer by forcing the tag to update its secrets.

None of these protocols consider the case where old and new owners do not trust the same database. Thus, the old owner's database still maintains control on the tag even after ownership transfer.

In [14] 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. However 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.

VIII. Conclusion

In this paper we propose a new protocol for delegation and ownership transfer. It fulfils the main security requirements among which tag privacy and confidentiality of tag's private information (e.g. tag's secret keys or identity). An attacker cannot modify surreptitiously messages exchanged between legitimate principals. Also, mutual authentication between the different principals is achieved preventing man-in-the-middle or replay attacks. In our protocol, the tag only needs to possess a small re-writable memory to store Kp, Ku and the counter's value. Moreover, as low cost practical and secure implementations of cryptographic symmetric key functions exist, its implementation remains sufficiently low cost.

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 *IEEE PerCom'03*, March 2003.

[3] A. Juels. RFID Security and privacy: A research survey. *IEEE JSAC*, 24(2):381–394, February 2006.

[4] D. Henrici and P. Muller. Hash-based Enhancement of Location Privacy for Radio Frequency Identification Devices using Varying Identifiers. In *IEEE PerSec'04*, March 2004.

[5] M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic Approach to Privacy-Friendly Tags. In *RFID Privacy Workshop*, 2003.

[6] G. Tsudik. YA-TRAP: Yet Another Trivial RFID Authentication Protocol. In *IEEE PerCom'06*, March 2006.

[7] D. Molnar, A. Soppera, and D. Wagner. A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags. In *SAC'05*, August 2005.

[8] A. Soppera and T. Burbridge. Secure by default: The RFID Acceptor Tag (RAT). In *RFIDSec'06*, July 2006.

[9] S. Fouladgar and H. Afifi. A Simple Delegation Protocol for RFID Systems (SiDeS). In *IEEE RFID*, March 2007.

[10] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. AES Implementation on a grain of sand. *Information Security, IEE proceedings*, 152(1):13–20, October 2005.

[11] P. Kaps and B. Sunar. Energy comparison of AES and SHA-1 for ubiquitous computing. In *EUC'06*, December.

[12] K. Osaka, T. Takagi, K. Yamazaki, and O. Takahashi. An Efficient and Secure RFID Security Method with Ownership Transfer. In *CIS'06*, November 2006.

[13] G. Avoine, E. Dysli, and P. Oeschlin. Reducing Time Complexity in RFID Systems. In *SAC'05*, August 2005.

[14] C. H. Lim and T. Kwon. Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer. In *ICICS'06*, December 2006.