RFID Tag Ownership Transfer

Boyeon Song

Information Security Group
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK
b.song@rhul.ac.uk

Abstract. In some applications, the bearer of a radio frequency identification (RFID) tag might change, with corresponding changes required in the RFID system infrastructure. We survey the security requirements for RFID tag ownership transfer, and propose novel authentication protocols for tag ownership and authorisation transfer. The proposed protocols satisfy most of the requirements that we present, and have desirable performance characteristics.

1 Introduction

Radio frequency identification (RFID) is an automatic identification and data capture technology that uses radio frequency (RF) to identify objects such as products, animals or persons. An RFID tag consists of an integrated circuit for storing and processing data, and an antenna for communicating via an RF signal with RFID readers. RFID readers are typically connected to a back-end server with a database containing information associated with the RFID tags that it manages.

The main advantage of RFID technology is the capability it offers for automatic, large scale, and contactless data collection. For this reason, RFID systems have been applied in a wide variety of fields, including product management, animal supervision, transportation payments, library book administration, entry access control, and electronic passports [5, 6].

However, security and privacy concerns arise from the use of such tags for the following reasons. An RFID reader and RFID tags communicate via a wireless channel using RF. Thus, interactions between a reader and tags are susceptible to eavesdropping. Also, each RFID tag has a unique value that is used for identification. If a tag emits its fixed value to every reader that queries it, then the location of the tag can be tracked by an attacker, and thus the privacy of the tag holder could be invaded. Moreover, an RFID tag is typically designed to be inexpensive for mass distribution. Such a low-cost tag has limited memory capacity and processing ability, and its memory is typically not tamper-resistant. That is, information stored in an RFID tag is vulnerable to compromise, e.g. by side-channel attacks.

A considerable volume of papers have been published providing solutions to these RFID security and privacy challenges. The following properties for a secure

RFID system have been proposed. To protect tag location privacy, an RFID tag should give an anonymous response to each reader query; to achieve such anonymity, many RFID protocols use pseudonyms based on random numbers [3, 4, 8, 10, 11, 14, 15]. To resist eavesdropping, messages between a reader and a tag should be cryptographically protected. Public key cryptography is too computationally intensive to be used in current RFID tags, and instead hash functions and symmetric cryptographic schemes are commonly applied. RFID protocols also need to address security threats arising from the compromise of a tag's internal state, including stored identifers and keys, because low-cost tags are not tamper-proof; even if an adversary compromises a tag, it should not be able to trace previous or future tag transactions using knowledge of the tag's internal state.

Another possible requirement for RFID systems is secure tag ownership transfer. In some applications, an RFID tag may change its owner a number of times during its lifetime. Ownership transfer means that the server of the new owner takes over tag authorisation, and so needs to be given the necessary private information to securely interact with and identify the tag. Thus all information associated with the tag will need to be passed from the old to the new owner. However, at the moment of tag ownership transfer, both the old and new owners have the information necessary to authenticate a tag, and this fact may cause an infringement of tag owner privacy. More specifically, if the previous owner is malicious, it may still be able to read the tag using retained tag information after transfer, and/or trace the new owner's transactions with the tag. That is, the privacy of the new owner might be compromised by the previous owner. Conversely, if the new owner is malicious, then it might be able to trace the previous owner's transactions with the tag. That is, the privacy of the previous owner might be compromised by the new owner.

In this paper, we examine the requirements for secure tag ownership transfer, and propose RFID authentication protocols satisfying such requirements. These protocols are combined with the scheme given in [15].

Section 2 introduces related work, and section 3 describes the general security requirements for RFID protocols as well as the specific requirements for tag ownership transfer. Section 4 proposes new protocols for secure tag ownership transfer, designed to provide the identified properties. In the following section, the privacy and security properties of the protocols are analysed, together with their performance characteristics, and the protocols for tag ownership transfer are also compared with the prior art. Section 6 concludes the paper.

2 Related Work

Molnar et al. [10] propose a pseudonym protocol (the MSW scheme) enabling ownership transfer of RFID tags. This appears to be the first paper explicitly dealing with ownership transfer. In this scheme, a tag responds with a different pseudonym, generated using a pseudorandom function, whenever it is queried by a reader. The set of tag secrets is organised in a tree, and a tag stores the secrets

corresponding to the path from the root to the tag. The tree structure enables time-limited delegation and ownership transfer of tags. If a reader has the secrets for a node in the tree of secrets, then it can compute the keys corresponding to all descendant nodes, and decode tag pseudonyms computed using these keys.

However, the ownership transfer procedure of this scheme is rather restrictive, in the sense that the old and new owners must trust the same Trusted Centre (TC), and the TC's database controls all the secret tag information. A reader that has received partial information from the TC can read the tag only a limited number of times without on-line connectivity to the TC. Thus, the scheme more closely resembles a time-limited access delegation scheme than a system for ownership transfer [8]. In addition, each tag must store $\lceil \log N \rceil$ secrets, and, if a tag is compromised, secret keys corresponding to descendant nodes can be compromised by exploiting intersections between key sets. Finally, the tag has to perform a significant number of computations in order to derive secrets and encode pseudonyms.

Saito et al. [14] suggest two approaches for reassigning an RFID tag's key for ownership transfer. The schemes protect the privacy of the new owner from the old owner by updating the tag key using a symmetric cryptosystem. As these schemes are only a process for key change, they need to be combined with an RFID authentication scheme to form a complete RFID system. The first scheme (the SIS-1 scheme) employs a three-party model using a Trusted Third Party (TTP). The previous owner first passes the secret key used for encrypting the tag identifier to the new owner. The new owner then generates a new encryption key, and asks the TTP to encrypt both the old key provided by the previous owner and the newly generated encryption key to send to the tag, so that the tag can update its identifier using the new key.

However, this scheme has usability and security issues, because the new owner must communicate with a TTP, and, if a tag is compromised, the secret key shared between the TTP and the tags will be exposed [14].

The second scheme (the SIS-2 scheme) uses a two-party model. It is based on the premise that the backward channel (i.e. the communication channel from the tag to the reader) is more secure against eavesdropping than the forward channel (i.e. the communication channel from the reader to the tag), because the range of the backward signal is shorter. In this scheme, the new owner of a tag encrypts both the old encryption key provided by the previous owner and the new key which the new owner has created, using a nonce received from the tag over the backward channel.

However, the assumption that intercepting the nonce sent from the tag is difficult is questionable, because, as pointed out in [14], an attacker could be located close to the tag and thereby successfully eavesdrop on the backward channel, despite its short range. Also, a tag always stores a fixed identifier, and hence the tag's past interactions could be traced if the tag is compromised.

An RFID authentication protocol (the LK scheme) proposed by Lim and Kwon [8] supports tag ownership transfer. They assert that forward untraceability (i.e. untraceability of the future interactions of a tag) is a fundamentally

important security property for an RFID protocol, and that complete transfer of tag ownership is possible only if some degree of forward untraceability is provided. In this scheme, each tag stores a tag secret, a server validator, and a counter, and makes use of three pseudorandom functions. In every authentication session, the tag secret is evolved using a one-way key chain in one of two different ways to achieve both forward and backward untraceability (i.e. untraceability of the past interactions of a tag); if the authentication succeeds, then the tag and the server refresh the tag secret probabilistically using the exchanged random numbers; if the protocol fails, then the tag updates its secret deterministically. For tag ownership transfer, the server of the new owner of a tag securely communicates with the server of the present owner, and receives all the relevant information for the tag. The new owner's server then communicates with the tag outside the reading range of the previous owner's server. As a result, the tag refreshes its secrets using random values shared only with the new server, and so no other party can communicate with the tag from this point onwards.

However, this scheme does not protect the privacy of the old owner. In addition, implementing the scheme is not easy, because the authentication and key updating processes of the protocol are both computation and storage intensive.

Osaka et al. [11] present an RFID security method (the OTYT scheme) that achieves ownership transferability with high efficiency. The scheme uses a hash function and a symmetric encryption scheme E. Each tag stores $E_k(ID)$ as its identifier, where k is a secret key and ID is a long term identifier (known only to the server). The server database contains ID, k, and $E_k(ID)$ for each tag. A reader queries a tag by sending a random number, and the tag responds with a hash of $E_k(ID)$ concatenated with the random number.

To transfer ownership of a tag, the present owner generates a new key k', updates its identifier to $E_{k'}(ID)$ in order to protect its privacy from the new owner, and then sends k' to the new owner via a secure channel. The new owner then creates a new key k'', computes a new tag identifier $E_{k''}(ID)$ in order to protect its privacy from the previous owner, and sends $E_{k'}(ID) \oplus E_{k''}(ID)$ to the tag so that the tag can obtain the new tag identifier. The scheme has the advantage that it requires a relatively small amount of tag computation; however, it has a number of security vulnerabilities. The value of $E_{k'}(ID) \oplus E_{k''}(ID)$ is not protected against message manipulation attacks, and the scheme does not prevent DoS attacks. In addition, if an attacker queries a tag twice using the same random number, then it receives the same response from the tag, enabling it to track the tag. Also, if a tag is compromised, an adversary might be able to learn its previous identifier from past transactions.

Fouladgar and Afifi [3, 4] present two methods for tag ownership transfer that are designed to guarantee the privacy of the new tag owner. In both schemes, a tag stores two secret keys k_p and k_u and a counter. Key k_p is used to compute pseudonyms and k_u is used to update both keys. The server database stores the values of ID, k_p and k_u for each tag. The first method [4] (the FA-1 scheme) uses a cryptographic hash function, and works under the assumption that both the old and new owners trust the same on-line server. The new owner of a tag

receives a hash-based pseudonym and a random number from the tag, and sends them to the present owner of the tag to request ownership transfer. The present owner then instructs the tag to set its counter to the maximum possible value. As a result, in the next session, the tag and the server of the new owner will update the copies of tag secrets k_p and k_u using a hash function and a random number generated by the tag. However, the scheme does not resist replay attacks, because a tag response is essentially the hash of a combination of the secret k_p and a random number generated by the tag, and thus an adversary can reuse the tag response to impersonate a tag. In addition, the tag keys k_p and k_u are fixed in every session after tag ownership transfer as long as the server does not deliberately change them. This might give rise to traceability threats.

The second scheme [3, 4] (the FA-2 scheme) uses symmetric cryptography, and achieves complete ownership transfer. That is, the database of the old owner does not maintain control of a tag and its secrets after ownership transfer. This was the first published scheme to take into account the possible need for after-sales service [3, 4]. For tag ownership transfer, the new owner of a tag receives a symmetric cryptography based pseudonym and a random number from the tag, and transmits them to the present owner asking for transfer of tag ownership. Before transferring ownership, the server of the present owner generates a random number, and makes both its database and the tag update the tag secrets k_p and k_u using random numbers generated by both the server and the tag, to protect its privacy from the new owner. It then passes the updated secrets to the new owner, along with other necessary information. To protect its privacy from the old owner of the tag, the new owner of the tag executes another session with the tag, thereby updating its keys. As a result, the old owner can no longer identify the tag. The new owner can also keep the key values transferred from the old owner if the tagged item has a warranty for after-sales service. However, the authors do not describe in detail how the tag can recover the old key kept by the old owner for after-sales service. Also, as in the FA-1 scheme, a server and a tag use the same keys k_p and k_u indefinitely after tag ownership transfer. If an adversary compromises a tag, then it may be able to trace the tag's past communications.

3 RFID System Requirements

3.1 Privacy and Security

RFID protocols should resist the following privacy and security threats [3, 15]:

- Tag information leakage: if an unauthorised reader can obtain a tag identifier, then it may be able to access the private information related to the tag held in the server database.
- Tag location tracking: if a tag's responses are linkable to each other, or distinguishable from those of other tags, then the tag's location could be tracked by unauthorised readers.

– Eavesdropping: an adversary could listen to communications between a reader and a tag.
– Tag impersonation: an adversary could impersonate a tag without knowing the tag's internal secrets. It could communicate with a reader instead of the tag and be authenticated as the tag.
– Replay attacks: an adversary could intercept messages exchanged between a reader and a tag, and replay them.
– Man-in-the-middle attacks: an adversary could insert or modify messages sent between a reader and a tag without being detected.
– Denial-of-service attacks: an adversary could interrupt or impede messages sent between a reader and a tag. Such an attack could cause a server and a tag to lose synchronisation [1, 8, 15].
– Backward traceability: if an adversary compromises a tag, then it might be able to trace previous transactions between a reader and the tag using knowledge of the tag internal state.
– Forward traceability: if an adversary compromises a tag, then it might be able to trace future transactions between a reader and the tag using knowledge of the tag internal state.
– Server impersonation: if an adversary compromises a tag, then it might be able to impersonate a legitimate server to the tag using knowledge of the tag internal state.

3.2 Performance

For cost reasons, an RFID tag typically has constrained memory and processing capabilities. Therefore, the volume of data stored in a tag, the complexity of tag computations, and the number and size of exchanged messages should be minimised.

3.3 Tag Ownership Transfer

In addition to the requirements presented in sections 3.1 and 3.2, some RFID systems are designed to support secure tag ownership transfer. In such RFID systems, changes of tag owner could occur frequently, and thus a secure and privacy-preserving means of tag ownership transfer is needed.

Tag ownership means having authorisation to identity the tag and control all the information related to the tag. Tag ownership transfer implies a shift of such capabilities to a new owner.

The following requirements for secure tag ownership transfer have been identified [3, 8, 11]:

– **New owner privacy**: Once ownership of a tag has been transferred to a new owner, only the new owner should be able to identify and control the tag. The previous owner of the tag should no longer be able to identify or trace the tag.

– **Old owner privacy**: When ownership of a tag has been transferred to a new owner, the new owner of a tag should not be able to trace past interactions between the tag and its previous owner.
– **Authorisation recovery**: In some special cases, such as after-sales service for an RFID tagged object, the previous owner of a tag might need to temporarily recover the means to interact with it. In such a case the present owner of the tag should be able to transfer its authorisation rights over the tag to the previous owner.

The possible need for authorisation recovery in an RFID system was first raised in [3, 4]. However, it seems that no concrete protocol to address this possible requirement has been proposed. In the next section, we introduce RFID protocols for tag ownership transfer which support this requirement.

4 RFID Protocols for Tag Ownership Transfer

We propose RFID authentication protocols fulfilling the requirements described in section 3.3. The protocols operate in conjunction with the scheme introduced in [15] (referred to here as the SM protocol), which satisfies most of the requirements identified in section 3.1.

4.1 Preliminaries

We use the following notation.

h	A hash function, $h : \{0,1\}^l \rightarrow \{0,1\}^l$
f	A keyed hash function, $f : \{0,1\}^l \times \{0,1\}^l \rightarrow \{0,1\}^l$ (a MAC algorithm)
T	A tag
S_j	The server of the j-th owner of T ($j \geq 1$)
l	The bit-length of a tag identifier t, or a random string r (where l should be large enough so that an exhaustive search to find the l-bit value is computationally infeasible.)
s	A string of l bits assigned to T
t	Tag T's identifier of l bits, which equals $h(s)$
$\hat{x}$	The most recent value of x
r	A random string of l bits
$\oplus$	XOR operator
$\|$	Concatenation operator
$\leftarrow$	Substitution operator
$x \gg a$	Right circular shift operator, which rotates all bits of x to the right by a bits, as if the right and left ends of x were joined.
$x \ll a$	Left circular shift operator, which rotates all bits of x to the left by a bits, as if the left and right ends of x were joined.
$\in_R$	The random choice operator, which randomly selects an element from a finite set using a uniform probability distribution.

Our protocol works under the following assumptions. A server communicates with tags via its reader, using an insecure RF interface. A server maintains a secure database of information for the tags that it owns, and has significantly greater processing ability than a tag. Each tag has a rewritable memory that may not be tamper-resistant, can generate pseudorandom numbers, and can compute hash functions h and f.

We thus implicitly assume that there are functions h and f which are suitable for a low-cost tag, sufficiently secure, and collision-resistant [4, 15]. Standard cryptographic hash functions such as SHA-1 or members of the MD family are rather unsuited to today's low-cost tags [2]. Instead, the following methods could be adopted to serve as hash functions. Weis [17] proposes the use of non-linear feedback shift registers to construct low-cost RFID hash functions [4, 7]. Yüksel et al. [18] propose several universal hash functions designed specifically for efficient hardware implementations and ultra-low power devices [4]. Pramstaller et al. [12] present a compact hardware implementation of Whirlpool, a hash function standardised by ISO/IEC and evaluated by the New European Schemes for Signatures, Integrity and Encryption (NESSIE) project [13].

We also assume that there is a sufficiently secure pseudorandom number generator (PRNG) for a low-cost tag. In practice, a block cipher or an iterated keyed hash which takes a cheap and weak pseudorandom source (for instance circuitry noise) and an internal key as inputs can be used as a PRNG [9, 16].

4.2 Protocol Description

We introduce a novel authentication protocol for tag ownership transfer, based on the SM protocol. The protocol consists of two sub-protocols: an ownership transfer protocol **P1**, and a secret update protocol **P2**. We also propose an RFID authentication protocol (**P3**) to provide authorisation recovery in line with the requirement identified in section 3.3.

P1: Ownership Transfer Protocol

In this protocol (referred to as **P1**), the new owner of a tag first requests the current owner for ownership of the tag. If the request is valid, the current owner then transfers all the information related to the tag to the new owner via a secure channel. **P1** is the same as the SM protocol, except for the addition of the ownership request and transfer of tag information; it is summarised in figure 1(a).

In order to use this protocol, the database of server S_j must contain the following entries for a tag T that it manages: the current secrets (t, s), the most recent secrets $(\hat{t}, \hat{s})$, and any other necessary information for T. Tag T stores a secret t as its identifier.

In order to take ownership of tag T, server S_{j+1} communicates with both T and S_j, as follows.

1. S_{j+1} generates a random string r_1 of l bits, and sends it to T.

2. In response, T generates a random string r_2 of l bits, computes $M_1 = t \oplus r_2$ and $M_2 = f_t(r_1 \oplus r_2)$, and then sends M_1 and M_2 to S_{j+1}.
3. S_{j+1} forwards r_1, M_1 and M_2 to S_j, with a request for ownership of T (R_T).
4. On receipt of this message, S_j performs the following steps:
   (a) If the received request R_T is valid, then S_j searches through its database for a pair (t, s) for which the value of t satisfies $M_2 = f_t(r_1 \oplus M_1 \oplus t)$.
   (b) If such a pair (t, s) is found, S_j sets $r_2 = M_1 \oplus t$ and computes $M_3 = s \oplus (r_2 \gg l/2)$. Otherwise, the session stops.
   (c) S_j updates the copies of the most recent secrets as $\hat{s} \leftarrow s$ and $\hat{t} \leftarrow t$, and the copies of the current secrets as $s \leftarrow (s \ll l/4) \oplus (t \gg l/4) \oplus r_1 \oplus r_2$ and $t \leftarrow h(s)$.
   (d) S_j sends M_3 to S_{j+1}, and transfers the updated secrets (t, s) and the other necessary information for T (Info) to S_{j+1} via a secure channel.
5. When S_{j+1} receives (t, s), Info, and M_3 from S_j, it stores (t, s) and Info in its database, and forwards M_3 to T.
6. T then performs the following steps:
   (a) T computes $s = M_3 \oplus (r_2 \gg l/2)$, and checks that $h(s) = t$.
   (b) If the verification succeeds, T has authenticated S_{j+1} as an authorised server, and updates its secret as $t \leftarrow h((s \ll l/4) \oplus (t \gg l/4) \oplus r_1 \oplus r_2)$. Otherwise, the session stops.

P2: Secret Update Protocol

After carrying out **P1**, server S_{j+1} has ownership of tag T. However, if S_j is malicious, it could still identify or trace T using the information it passed to S_{j+1}. Thus S_{j+1} needs to execute another protocol (**P2**) to establish new secrets with T, in order to protect the new owner's privacy. Protocol **P2** should be performed at a distance from any readers connected to S_j, in order to prevent S_j eavesdropping on the messages.

Prior to running **P2**, we assume that server S_{j+1} has a pair of secrets (t, s) for tag T, obtained as a result of executing **P1**; we also suppose that T has identifier t. **P2** involves the following steps.

1. S_{j+1} generates random strings r_1 and s' of l bits, and computes $t' = h(s')$. S_{j+1} then computes $M_1 = f_t(r_1) \oplus t'$ and $M_2 = s \oplus (t' \gg l/2)$, and sends r_1, M_1 and M_2 to T.
2. When T receives r_1, M_1 and M_2 from S_{j+1}, it performs the following steps:
   (a) T computes $t' = M_1 \oplus f_t(r_1)$ and $s = M_2 \oplus (t' \gg l/2)$.
   (b) If $h(s) = t$, T has authenticated S_{j+1} as an authorised server. Otherwise, the session stops.
   (c) T updates its secret as $t \leftarrow t'$, and generates a random string r_2 of l bits.
   (d) T computes $M_3 = f_t(r_1 \oplus r_2)$ using the new secret t, and sends r_2 and M_3 to S_{j+1}.
3. S_{j+1} checks that M_3 is equal to $f_{t'}(r_1 \oplus r_2)$. If the validation succeeds, S_{j+1} now knows that T has had the new secret t', and updates secrets s and t for T to s' and t', respectively. Otherwise, S_{j+1} goes to step 1, and starts a new session.

If **P2** completes successfully, S_{j+1} and T share new secrets known only to them, and S_j is no longer able to identify or trace T. Both S_j and S_{j+1} can also keep the pair (t, s) provided by S_j for use in the event that S_j needs to identify T again. **P2** is summarised in Figure 1(b).

P3: Authorisation Recovery Protocol

As discussed above, the previous owner of a tag may need to temporarily interact with the tag again. The following authentication protocol (referred to as **P3**) enables this to occur.

P3 enables server S_{j+1} to make T change its secret back to the value it had when S_{j+1} took ownership of T from S_j. Prior to running **P3**, we assume that S_{j+1} stores the following information for tag T: the current secrets (t, s), the most recent secrets $(\hat{t}, \hat{s})$, and the old secrets also known to S_j, $(\bar{t}, \bar{s})$ say; we also suppose that tag T has identifier t. **P3** is the same as **P2** except that t' is set equal to $\bar{t}$, i.e. the old value of t. After successful execution of **P3**, T stores $\bar{t}$ as its identifier. As a result, S_j can identify T again.

If S_{j+1} executes **P1** and **P2** again, it can recover authorisation on T from S_j.

5 Analysis

We evaluate **P2** and **P3** with respect to privacy, security, and performance, using the requirements presented in section 3. We do not analyse **P1** here, since an analysis of an essentially identical protocol is given in [15].

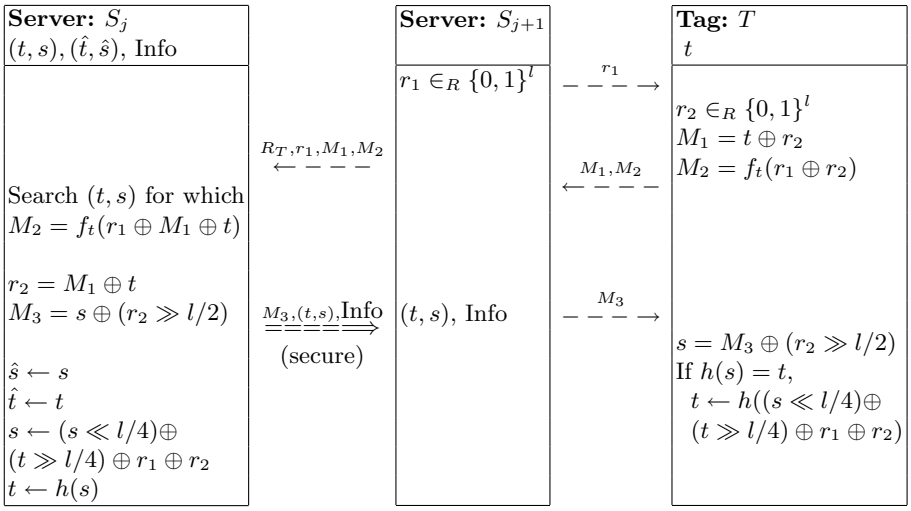
5.1 Privacy and Security

P2 and **P3** are mutual authentication protocols. In these schemes, when tag T receives r_1, M_1, and M_2 from server S_{j+1}, T authenticates S_{j+1} by obtaining s from the messages and checking that $h(s) = t$. This works because s is a secret for T known only by S_{j+1}. S_{j+1} authenticates T by checking that the received M_3 is correct, since it is computed using t, which is known only to T and S_{j+1}. S_{j+1} can also confirm that T has the same secret t as S_{j+1}.

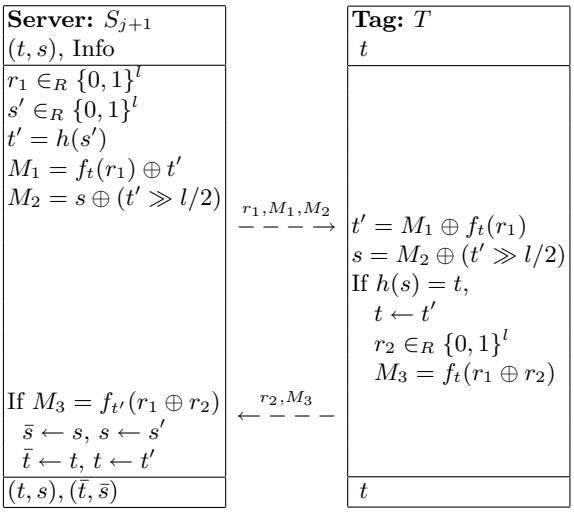
The schemes protect against tag information leakage because T's responses are a function of its secret t, and thus only the server that knows the secret is able to identify T and access the tag information. The schemes protect against tag location tracking because T's responses are anonymous, since they are a function of random strings r_1 and r_2, and are independent of one another.

The messages exchanged between the server and the tag are computed using r_1 and r_2, secrets t and s, and a keyed hash function f. The protocols resist eavesdropping, replay attacks, and man-in-the-middle attacks.

To impersonate a tag, an attacker must be able to compute a valid response M_3. However, it is difficult to compute such a message without knowledge of a secret t which is used as a key for a keyed hash function f.



(a) Ownership Transfer Protocol



(b) Secret Update Protocol

Fig. 1. Tag Ownership Transfer Protocol

P2 and **P3** resist denial-of-service attacks. Suppose that an adversary prevents message M_3 reaching S_{j+1} in **P2** (or **P3**). Then T will update its identifier, but S_{j+1} will not. However, S_{j+1} knows the updated value of t, and can use the value to recover synchronisation with T.

Even if T is compromised, the protocols do not enable backward traceability, because T's secret t is updated using a non-invertible hash function h, and hence the previous secret identifiers of T are not computable. The schemes also resist forward traceability if an adversary does not obtain the values of either r_1 or r_2 exchanged between the server and the tag after T has been compromised, because, like in the protocol described in [15], t is updated using r_1 and r_2 after every successful session. Even if an adversary has compromised T, it cannot impersonate a legitimate server in **P2** and **P3** without additional information. This is because the server's message M_2 is a function of the secret s known only to the server, and the adversary cannot obtain the value even if it compromises T. To succeed in such an attack, the adversary must first obtain s via some other method.

P2, **P3** and the previous art introduced in section 2 are compared in Table 1. The table shows that the proposed schemes protect against the identified privacy and security threats.

Table 1. Privacy and Security Properties

	LK	OTYT	FA-1	FA-2	**P1(SM)**	**P2/3**
Tag information leakage	√	√	√	√	√	√
Tag location tracking	√	·	√	√	√	√
Eavesdropping	√	√	√	√	√	√
Tag impersonation	√	√	√	√	√	√
Replay attack	√	√	·	√	√	√
Man-in-the-middle attack	√	·	√	√	√	√
Denial-of-service attack	√	·	√	√	√	√
Backward traceability	√	·	√	·	√	√
Forward traceability	∗	·	·	·	∗	∗
Server impersonation	∗	·	·	·	∗	∗

√ : resists such an attack
∗ : resists attack under an assumption
· : does not protect against such an attack

5.2 Performance

P2 and **P3** are efficient in terms of non-volatile memory and communication requirements, because a tag needs only l bits of non-volatile memory to store its secret t, and only two messages need to be exchanged to provide mutual authentication between the server and the tag.

P2 and **P3** have modest computational requirements; the only cryptographic functions (which are significantly more computationally complex computations than arithmetic and logical operations) required by **P2** or **P3** are at most three hash function computations in the tag and the server. In **P2**, both T and S_{j+1} need to compute h once and f twice. In **P3**, T needs to compute h once and f twice, and S_{j+1} needs to compute f twice.

Table 2 compares the performance characteristics of **P2** and **P3** with the secret update process for other proposed schemes. The table indicates the type and number of cryptographic functions required of a tag, the type and number of entries stored in tag non-volatile memory, and the number of exchanged messages in each protocol. In Table 2, PF is a pseudorandom function, HF is a hash function, SE is a symmetric encryption, k_s is a tag identifier, k_w is a server validator, k_e is a tag identifier $E_k(ID)$, k_p is a key used to compute pseudonyms, k_u is a key used to update keys, and c is a counter.

Table 2. Performance of Secret Update Protocol

	LK	OTYT	FA-1	FA-2	**P2/3**
Tag computations	4 PF	1 HF	5 HF	2 SE	3 HF
Tag storage	k_s, k_w, c	k_e	k_u, k_p, c	k_u, k_p, c	t
Message flows	3	3	3	3	2

5.3 Tag Ownership Transfer

P1, **P2**, and **P3** meet the three requirements for tag ownership transfer identified in section 3.3.

First, **P2** is designed to protect the privacy of the new owner from the old owner of a tag T. That is, future interactions between the new owner and T are secure against tracing by the old owner. Ownership of T is transferred to the new owner in **P1**, and the new owner and T establish new secrets using **P2**. As a result, the old owner is no longer able to read T.

The protocols also protect the privacy of the old owner from the new owner of T. In **P1**, after the old owner updates secrets s and t for T using a hash function h, it transfers the updated secrets to the new owner. Thus, the new owner cannot trace previous transactions between the old owner and the tag, using knowledge of the updated secrets.

Finally, **P3** provides authorisation recovery, the third requirement described in section 3.3. **P3** causes T to change its secret t to $\bar{t}$, which the new owner received from the old owner when ownership of T was transferred. As a result, the old owner recovers authorisation to identify T, and thus can interact with T again.

Table 3 compares **P1**, **P2** and **P3** to the schemes described in section 2 with respect to the security of tag ownership transfer, where $\circ/\times$ respectively denote

a property provided/not provided. The new protocols provide all the identified properties.

Table 3. Properties for Tag Ownership Transfer

	MSW	SIS-1	SIS-2	LK	OTYT	FA-1	FA-2	**P1,P2,P3**
New owner privacy	×	○	○	○	○	×	○	○
Old owner privacy	×	×	×	×	○	×	○	○
Authorisation recovery	×	×	×	×	×	×	×	○

6 Conclusion

In some RFID applications it is necessary to allow for transfer of tag ownership. We have identified three requirements for secure and privacy-preserving tag ownership transfer: new owner privacy, old owner privacy, and authorisation recovery. We have proposed novel RFID authentication protocols for tag ownership transfer that meet such requirements. The scheme consists of three protocols: **P1**, an ownership transfer protocol, based on the SM protocol presented in [15], **P2**, a secret update protocol, and **P3**, an authorisation recovery protocol. We believe that this latter protocol is the first proposed practical authentication scheme for authorisation recovery.

We have also analysed and compared **P2** and **P3** to the prior art. The schemes satisfy the identified privacy and security requirements. **P2** and **P3** have desirable performance characteristics; a tag needs less non-volatile memory than in previously proposed schemes, performs just three hash function computations, and the number of message flows between the tag and the server is only two, with mutual authentication. **P1**, **P2**, and **P3** also provide all the identified requirements for tag ownership transfer.

Acknowledgements

I am very grateful to Chris J. Mitchell for his helpful input and comments.

References

1. H. Chien and C. Chen. Mutual authentication protocol for RFID conforming to EPC class 1 generation 2 standards. *Computer Standards & Interfaces*, 29(2):254–259, February 2007.
2. M. Feldhofer and C. Rechberger. A case against currently used hash functions in RFID protocols. Printed handout of Workshop on RFID Security – RFIDSec 06, July 2006.

3. S. Fouladgar and H. Afifi. An efficient delegation and transfer of ownership protocol for RFID tags. In *First International EURASIP Workshop on RFID Technology*, Vienna, Austria, September 2007.
4. S. Fouladgar and H. Afifi. A simple privacy protecting scheme enabling delegation and ownership transfer for RFID tags. *Journal of Communications*, 2(6):6–13, November 2007.
5. T. Haver. Security and privacy in RFID applications. Masters thesis, Norewegian University of Science and Technology, Trondheim, Norway, June 2006.
6. A. Juels. RFID security and privacy: A research survey. *IEEE Journal on Selected Areas in Communications*, 24:381–394, February 2006.
7. M. Lehtonen, T. Staake, F. Michahelles, and E. Fleisch. From identification to authentication — a review of RFID product authentication techniques. In *Printed handout of Workshop on RFID Security — RFIDSec 2006*, 2006.
8. C. Lim and T. Kwon. Strong and robust RFID authentication enabling perfect ownership transfer. In P. Ning, S. Qing, and N. Li, editors, *Conference on Information and Communications Security — ICICS '06*, volume 4307 of *Lecture Notes in Computer Science*, pages 1–20, Raleigh, North Carolina, USA, December 2006. Springer-Verlag.
9. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*, volume 6 of *Discrete Mathematics and Its Applications*. CRC Press, 1996.
10. D. Molnar, A. Soppera, and D. Wagner. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography — SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 276–290, Kingston, Canada, August 2005. Springer-Verlag.
11. K. Osaka, T. Takagi, K. Yamazaki, and O. Takahashi. An efficient and secure RFID security method with ownership transfer. In Y. Wang, Y. Cheung, and H. Liu, editors, *Computational Intelligence and Security — CIS 2006*, volume 4456 of *Lecture Notes in Computer Science*, pages 778–787. Springer, Berlin, September 2006.
12. N. Pramstaller, C. Rechberger, and V. Rijmen. A compact FPGA implementation of the hash function whirlpool. In *ACM/SIGDA 14th international symposium on Field Programmable Gate Arrays — FPGA'06*, ACM Press, pages 159–166, New York, 2006.
13. B. Preneel et al. Final report of European project IST-1999-12324: New European schemes for signatures, integrity, and encryption. Available at: www.cosic.esat.kuleuven.be/nessie/, April 2004.
14. J. Saito, K. Imamoto, and K. Sakurai. Reassignment scheme of an RFID tags key for owner transfer. In T. Enokido, L. Yan, B. Xiao, D. Kim, Y. Dai, and L.T. Yang, editors, *Embedded and Ubiquitous Computing – EUC 2005 Workshops*, volume 3823 of *Lecture Notes in Computer Science*, pages 1303–1312. Springer, Berlin, November 2005.
15. B. Song and C. J. Mitchell. RFID authentication protocol for low-cost tags. In V. D. Gligor, J. Hubaux, and R. Poovendran, editors, *ACM Conference on Wireless Network Security — WiSec '08*, pages 140–147, Alexandria, Virginia, USA, April 2008. ACM press.
16. G. Tsudik. YA-TRAP: Yet another trivial RFID authentication protocol. In *Fourth IEEE Annual Conference on Pervasive Computing and Communications — PerCom 2006*, pages 640–643, Pisa, Italy, March 2006. IEEE Computer Society Press.

17. S. Weis. Security and privacy in radio-frequency identification devices. Master's thesis, Massachusetts Institute of Technology (MIT), Massachusetts, USA, May 2003.
18. K. Yüksel. Universal hashing for ultra-low-power cryptographic hardware applications. Master's thesis, Dept. of Electronical Engineering, Worcester Polytechnic Institute, Worcester, MA, USA, 2004.