Key Evolving RFID Systems: Forward/Backward Privacy and Ownership Transfer of RFID tags

Tassos Dimitriou[a,*]

[a] *Computer Engineering Dept., Kuwait University*
Research Academic Computer Technology Institute, Athens, Greece

Abstract

RFID, which stands for Radio Frequency Identification, is a relatively new technology often envisioned as an enabler of the Internet of Things. The widespread use of this technology, however, introduces many security and privacy risks since tags contain information that can be easily obtained by anyone with a reader. Eventually this can lead to tracking of users, profiling and violation of their basic right to privacy.

In this work we make an important step in providing for RFID privacy by letting users control all the tags they possess. The moment a person buys a tagged object and becomes the owner of it, no one should be able to find any information about the object or have access to the tag. We do this by developing and formalizing the notion of a *key-evolving* RFID system. In particular, (i) we explain how such a system can be made forward secure using pseudo-random generators and functions, (ii) we derive concrete results based on the security of the underlying primitives, and (iii), we explain how this can be realized in practice using a protocol that achieves secure ownership transfer and controlled delegation without relying on Trusted Third Parties.

Keywords: RFID Security, Ownership Transfer, Controlled Delegation, Authorization Recovery, Forward and Backward Privacy, Key-evolving Systems.

1. Introduction

Radio Frequency Identification uses small devices called RFID tags to remotely access and retrieve data stored in objects. Such embedded tags can have remarkable applications. At a smaller scale they can improve efficiency in inventory control, logistics and supply chain management. At a larger scale, they can enable the creation of an entirely new network, an Internet of Things, which can be used to promote access and connectivity for *anything*.

RFID provides a simple, unobtrusive and cost-effective system of item identification, connecting everyday objects and devices to large databases and networks. This technology will allow the interaction with physical objects and everyday items, turning the static objects of today into dynamic and intelligent ones. This can lead to the creation of innovative products and services, thus enabling new forms of communication between people and their environments as well as between objects themselves [1].

Applications of RFID technology are numerous; they include toll collection, supply-chain management, counterfeiting control, patient and child monitoring, and so on. E-government applications are well under development that include RFID in driver licences, passports or cash. RFID readers are already embedded in mobile phones. Advances in smart homes, personal robotics and wearable computing are also areas that will benefit the most from RFID deployment, eventually contributing to the vision of a fully interactive environment [2].

However, the use of this technology introduces serious privacy risks. As RFID tags respond to any reader request, even without knowledge of the owner, data stored in them can easily be retrieved by readers placed in strategic locations (entrance of buildings, points of interest in the city, and so on). This, in turn, may lead to tracking and profiling of individuals by the tags contained in the objects they carry [3, 4, 5]. Unfortunately, the scale and capacity of these new technologies can intensify this problem since invisible data exchange between things and people, or between things themselves, may occur without the consent of the owners and originators of such data. The question then becomes: *Who will control the data collected by all these devices embedded in the surrounding environment*?

This problem is further magnified by the ever-increasing use of RFID objects in personal environments, where mobile RFID readers (eg. embedded in mobile phones [6]) can interact with home appliances, thus enabling a more natural interaction between users and their home devices. It should be clear that the use of a centralized model where tag responses must be conveyed to some company database in order to be interpreted by the reader creates serious privacy and trust issues. In this work, we develop protocols

* *Email address:* tassos.dimitriou@ieee.org (Tassos Dimitriou)

specifically tailored for such personal environments where users, by means of their personal readers, directly own tags, thus removing the need of Trusted Third Parties required in centralized models.

Our Contribution: As a first step towards RFID privacy, we feel that anyone in possession of RFID-tagged items should be in complete control of the tags they carry. Additionally, we believe the vision of Internet of Things will be severely limited if it does not allow for the possibility of transferring objects to new owners without violating the privacy of past ones. However, the existing model makes the implicit assumption that the old and new owners must trust the same central database which by default controls all tag secrets. Hence tracking is still possible.

This paper extends and improves our prior work in [7] with significant new material. Our contributions can be summarized by the following: (i) We formalize the notion of a *key-evolving* RFID system that can be used to guarantee the concept of forward privacy. (ii) We show how such systems can be built using basic RFID systems and forward secure pseuro-random number generators or pseudo-random functions by showing how their security reduces to the security of the underlying primitives. (iii) We extend our solution and propose a realistic implementation of a system that guarantees owner privacy. This system additionally provides for controlled delegation and authorization recovery which are properties particularly useful for after-sales and maintenance services. (iv) Finally, we thoroughly analyze the security of this proposal and we show how it ensures the privacy of both new and past owners based on a set of realistic assumptions.

Paper Organization: The rest of the paper is organized as follows. In Section 2, we discuss related work and summarize previous efforts in providing for forward privacy and ownership transfer in RFID systems. In Section 3, we present a typical architecture of an RFID system, we motivate the need for forward privacy and discuss the threats that can be applied when tags can change owners. The definition and the security proofs for key-evolving, forward-secure RFID systems appear in Sections 4 and 5. Based on this, a realistic implementation is shown in Section 6. Finally, Section 7 concludes this work.

2. Related Work

The problem of RFID privacy[1] has been considered extensively in the literature but only a few works have examined ownership transfer of RFID tags.

In [8], an ownership transfer scheme has been proposed based on a tree of secret keys where each tag is preloaded with all the keys corresponding to the path from the root to the tag. When queried by a reader, tags generate a response from each such key in the path, so a reader that knows these secrets in advance can easily disambiguate the tags. Unfortunately, compromise of one tag may reveal information about others as it has been shown in [9, 10]. Furthermore, delegation is limited and the back-end system still knows the secrets of the tags. Thus owners (new and past) *must still trust the same central system* which controls all secret tag information.

Another ownership transfer protocol has been proposed in [11]. There a new key is planted to the tag after the interaction of the new owner with the back-end database. However, this approach suffers from a number of shortcomings; the database cannot be sure that the tag updated its secret which may lead to DoS and desynchronization problems. Also, if a tag is compromised, an adversary might be able to learn its previous identifier from past transactions, thus forward privacy is not guaranteed. Finally, the back-end database must always be *online* for tag access and ownership transfer, something that is not necessary in our proposal.

In [12], a protocol that enables ownership transfer is also presented. Each tag stores a counter, a secret and a server validator, and uses three pseudorandom functions to update the tag secrets in order to achieve both forward and backward privacy. However, implementing this protocol is not easy as a rather large number of computations is required by the tag and the maintenance of a number of hash chains by the central system to ensure authenticity of transactions. Thus the protocol is both computation and storage intensive. Additionally, in [13] a study of forward and backward untraceable schemes is proposed which is based on the analysis of pseudorandom bit generators. The authors present a privacy analysis of the scheme in [12] and show how to trace a tag even without requiring tag state corruption.

Berbain *et al.* [14] build upon the protocol of [15] trying to address the problem of securely updating a tag's state without relying on the use of hash functions to create the chain for one-wayness. However, this work does not deal with the problem of secure ownership transfer. An extension of this model appears in [16], where the internal states of the tags are continuously updated. The main difference with the work in [14] lies in the use of a secure protocol to address mutual authentication and the definition of a slightly relaxed notion of almost forward privacy.

In [17], the privacy of two more RFID tag ownership transfer protocols is investigated against the tag owners as adversaries. These schemes [18, 19] have been found vulnerable to attacks that target forward and backward privacy as well as the privacy of previous and new owners; these attacks succeed when the adversary is one of the owners in the system. A more thorough investigation of attacks that can be applied to secure ownership transfer protocols can be found in the work of Peris-Lopez *et al.* [20].

To address scalability issues in tag identification, Song

[1] See `http://www.avoine.net/rfid/index.html` for an online repository of RFID security-related papers.

and Mitchell [22] used a pre-computed table which contains hash chains for each tag. This allows for constant identification but only when the chain is not exhausted. The authors also introduced the concept of *authorization recovery* which allows a previous owner to identify a tag again. This is basically achieved by having the tag remember the old key and transfer control to the previous owner. A similar idea has been followed in [21] which allows a tag to be released but only to the original manufacturer. These ideas have been further considered in [23] and [24].

Fernandez-Mir *et al.* in [23], proposed a protocol supporting ownership transfer which again relies on the use of hash chains to identify a tag. The protocol further provides for *controlled delegation* in which the tag can be temporarily identified by a third party while the owner still reserves the right to control the tag. The authors provide for a number of schemes to defend against desynchronization issues, however if the number of tries exceeds a predefined threshold, a legitimate reader will not be able to identify the tag. Hence this threshold value is important to restrict denial of service attacks.

Ng *et al.* in [24] simplify the delegation process which does not require any more the current owner to hand a large number of IDs to the new reader in order to identify the tag. The proposed scheme satisfies most security properties of ownership transfer while only some hash calculations are required on the tag. However, the protocol does not consider replay attacks and as a result an old key may be planted to a tag and a delegate can extend the period of tag delegation. Additionally, the authors assume that there is always a target tag, which has been authenticated already. This simplifies protocol design but requires this step to be executed prior to all ownership operations. Later, in Section 6, we will show how to realize authorization recovery and controlled delegation by integrating authentication to them while still making ownership transfer possible.

In this work we also attempt to formalize the notion of forward privacy in RFID tags by extending the model of Juels and Weis [25] that follows an adversary-game approach based on notions of indistinguishability. However, this protocol attempts to characterize the privacy of systems in which tags share *correlated* secrets and does not capture the notions of forward and backward privacy as we do here. A similar model can be found in [26], where the goal is to capture a range of adversarial abilities on the various communication channels between tags and readers, and in [27], where the goal is to study tradeoffs between security and efficiency in tree-based systems where tags *share* secrets. However, in both of these models the focus is not on the properties we wish to investigate here. A preliminary model for addressing forward and backward privacy for ownership transfer has appeared in our prior work [21] but without the actual constructions that we will develop in the next sections.

Vaudenay, in [28], proposed another model in which the adversary can corrupt tags at any time, resulting in a taxonomy of privacy in various classes and setting the question whether forward privacy without requiring public key cryptography is possible. *One positive aspect of our work is that this is possible by dividing time in key evolving periods.* This is an important result that enhances the use of RFID technology in practice. The work of Van Le *et al.* in [29] is based on the universal composability framework and defines security in terms of indistinguishability between real and ideal protocol simulations. Similarly, the work in Deng *et al.* [30] is based on the notion of zero knowledge privacy. The reader is referred to Hermans *et al.* [31] for a more detailed discussion of the properties of the various models.

Figure 1: Architecture of a typical RFID-system

3. Threat model, assumptions and design goals

A typical architecture of an RFID system includes RFID tags, readers and a back-end database (Figure 1). An RFID device carries an antenna and a radio transceiver, which read the radio frequency and transfer some piece of information to the reader. This information is subsequently passed to the back-end database which helps identify the tag and return relevant information about the tagged object.

3.1. Nature of communications

Communication between tags and readers is wireless and is therefore subject to eavesdropping attacks. For example, a person carrying an item equipped with an RFID tag (e.g. a briefcase), can be tracked down by the tag he/she is carrying by means of rogue readers; if the tag returns the same static identifier, the reader will be able to correlate tag's responses and ultimately identify the person carrying the tag. Reader-to-database communications, on the other hand, are considered secure since both can handle the overhead introduced by encryption. One unusual aspect of tag-reader communications is the asymmetry in signal strength: because most tags are *passive* (they respond to a signal broadcasted by the reader), it is usually much simpler to eavesdrop on data emitted by a reader towards a tag than vice versa.

3.2. Secret information stored in tags

Any secure reader-tag communication protocol typically involves, in addition to identification, tag authentication or mutual authentication. This is achieved using a key K shared between the tag and the reader (or back-end database) and appropriate challenge-response protocols. This key is used to secure any information transmitted by the tag and enhance user privacy against clandestine reading and scanning.

In this work we do not treat tags as tamper-resistant devices. Hence all information, including secret keys, stored in the tag is available to determined adversaries. Addressing this threat is an important requirement since otherwise an adversary could use this knowledge to identify tag interactions that occurred in the past and hence be able to track the owner's past behavior. This is captured by the notion of an *inside attacker* below (it is called a *strong attacker* in [22]). Additionally, such an attacker can use information about the current key to track the behavior and whereabouts of future tag owners. Hence knowledge of the key must be properly dealt with (however, the issue of whether an adversary can plant this key into a fake tag and simulate tag responses without being detected is outside the scope of this work). Finally, we assume that adversaries lack the ability to write back and modify the internal state of the tag without actually destroying the tag. Essentially, we consider the tag inoperable and no longer responding to commands if this happens.

3.3. Nature of attackers

Protecting the tag-reader interactions provides security only against *outside* attackers, i.e. third parties trying to find information about a user's tags and transactions. This protection is basically achieved by refreshing tag identifiers and making tag responses indistinguishable from random data. But there is still potential for violation of privacy as in all existing schemes the back-end database has knowledge of *all* the secrets associated with a company's tagged products. While this company may not be able to track the tags issued by other companies, it can still track the movements of tags issued by it. Furthermore, a *coalition* of such companies may create a federation network in order to exchange information about a user's movements, habits and profile.

As a consequence, once an item is purchased or changed hands, the user should be able to hide information about the tag, even by the original owners (central database or others) of this item. This can be achieved by *updating* the tag's secret so that past actions cannot longer be connected to current activities on the tag (forward privacy). Additionally, past owners should not be able to trace the actions of future owners (backward privacy).

So, our threat model need to be extended *beyond* the back-end database which possesses all tag secrets. It also has to include (possibly malicious) owners who have acquired tags and now they transfer them to new users (we call these *inside* attackers). These malicious insiders can do all sort of bad things with the tags, including trying to track the activities of the new (or past) owners. Thus ownership transfer must be done in a way to guarantee protection against these types of attacks. The two privacy issues related to ownership transfer can thus be summarized as follows ([21]-[24]):

- *New owner privacy*: Once a tag is acquired by a new owner, only the new owner should be able to access the tag. Past owners should not be able to use or track the tag anymore, nor identify the transactions made by the new owner.
- *Old owner privacy*: When a tag has a new owner, past interactions of the tag with previous owners should not be traceable.

In addition to privacy related attacks, we identify a set of threats that affect correct protocol design and should be handled by any secure tag-reader communication protocol. These include:

- *Tag/reader/server impersonation*: It should not be possible to impersonate a tag to a reader (or vice versa) without knowledge of the secret shared between the two. Additionally, any attempt to impersonate a server in order to recover tag secrets or lead to DoS problems (see below) should not be possible. Any such attempt should be easily defeated using appropriate challenge-response protocols and should not have any impact in proper tag identification or ownership transfer.
- *Replay/Intergrity attacks*: The protocol should be resistant to attacks where an attacker replays or forwards messages to fool one of the participants accept a request from the other as authentic. Thus messages should be authenticated and include fresh identifiers that participants can identify. Additionally, attempts to modify messages on the fly that can lead to erroneous results should not have any impact on protocol operations.
- *Desynchronization*: Blocked (or lost) messages transmitted between a tag and a reader should not result in one updating the secret state while the other does not. This would result in desynchronization and Denial-of-Service (DoS) as it would no longer be possible to identify or transfer tags to new owners.

3.4. Design goals and functional requirements

In addition to the security goals outlined above, our solution should avoid reliance on Trusted Third Parties (TTPs). This is especially important when transferring ownership since otherwise transactions made by the new owners could still be monitored by the TTP and the notion of ownership will not longer be well defined. Furthermore, our solution should be computation- as well as

communication-efficient. Since protocols will be run on resource-constrained RFID devices, dependency on expensive cryptographic operations should be avoided.

Finally, our protocol will address the following functional requirements of secure ownership transfer:

- *Controlled delegation*: The current owner maintains ownership and control of the tag but can delegate tag-access to another entity called the *delegate*. The delegate can only query and identify the tag for a limited (specified by the owner) number of times which, however, can be canceled at any time by the current owner.
- *Authorization recovery*: Authorization recovery allows a previous owner to regain access to the tag. This may be useful in situations where the previous owner is a manufacturer and the tagged product must be returned for service. This property can be realized using the methods of transferring ownership, however in the best case scenario the current owner essentially *shares* the tag, while in fact it *loses* ownership of the tag. A more preferable method is to simply delegate access in which case the current owner still retains complete control of the tag. Thus, authorization recovery can be thought as a special case of controlled delegation.
- *Complete ownership transfer*: The existing owner transfers all tag secrets to the new owner which takes complete control of the tag. However, this must be done in a way to ensure the privacy of both owners as per the following requirements.
- *Forward privacy*: This property ensures that past transactions with the tag are still protected even if the tag has a new owner now. Hence this is similar to the notion of old owner privacy outlined previously.
- *Backward privacy*: Similarly, future owners should be protected from current or past ones. Hence this is similar to the notion of new owner privacy.

We start by formalizing the concept of forward privacy in RFID tags as it is provided by a *key-evolving* RFID system. Then we show how this can be realized using pseudorandom generators and pseudorandom functions.

4. Basic Privacy for RFID tags

In this section we formalize the notion of privacy in RFID tags. We follow the model proposed in [25] that is based on indistinguishability of tag responses (a closely related model is the one proposed in [26]). The idea is that an RFID protocol is considered private if an adversary cannot differentiate between two different tags within the limits of its computational power and number of queries. For completeness, we briefly review the basic privacy model of [25]. Then, in Section 5, we provide our construction for forward secure key-evolving RFID systems.

Basic Privacy Model

We regard an RFID system $\mathcal{S} = (\textsc{Keys}, \mathcal{R}, \{\mathcal{T}_i\})$ as consisting of a single reader $\mathcal{R}$, a set of n tags $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$ and a randomized tag key generation function Keys. The model allows for a set of tag and reader functionalities such as SetKey, TagInit and ReaderInit.

- SetKey models the ability of adversaries to compromise tags as well as manufacture/clone arbitrary tags. A tag responds to a SetKey message by revealing its current key and replacing it with a new one sent by the caller.
- TagInit and ReaderInit messages associate a tag and a reader with a particular session id *sid*. As part of a current session id, a tag and a reader may exchange challenge-response messages. Once a tag is initialized with a *sid*, it can respond to a challenge c_j with a response r_j, which can be a function of the key, the previous challenge-response pairs and locally generated random bits. Similarly, a reader upon receiving a response r_j associated with *sid*, it can "accept"/"reject" the tag or compute the next challenge c_{j+1} as part of an ongoing protocol interaction.

An adversary $\mathcal{A}$ in this model can issue its own SetKey, TagInit, ReaderInit as well as challenge and response messages. When a tag receives a SetKey message it is considered compromised by the adversary. It is assumed that $\mathcal{A}$ can corrupt any of a number of tags except the two that are part of the indistinguishability game. The description of the basic indistinguishability game is shown in Figure 2.

The system is initially setup by running the key generation algorithm $\textsc{Keys}(1^k)$ which produces a set of keys to be assigned to the n tags. In the *training* phase the adversary is allowed to send any number of TagInit, ReaderInitand challenge-response commands within the appropriate time bounds, and corrupt via SetKey messages any set (up to $n-2$) of tags, where n denotes the number of tags in the system.

In the *challenge* phase, the adversary selects two uncorrupted tags as the challenge candidates. One of these tags is then randomly selected and presented to $\mathcal{A}$ via an appropriate oracle. Eventually, the adversary has to distinguish between the two tags in order to be successful. This is captured by the following definitions:

$$\mathbf{Adv}^{ind}_{\mathcal{A},\mathcal{S}} = 2 \cdot \Pr[\mathbf{Exp}^{ind}_{\mathcal{A},\mathcal{S}} = success] - 1$$

$$\mathbf{Adv}^{ind}_{\mathcal{S}}[r,s,t] = \max_{\mathcal{A}}\{\mathbf{Adv}^{ind}_{\mathcal{A},\mathcal{S}}\}.$$

The first term is the *indistinguishability* advantage of $\mathcal{A}$ in attacking $\mathcal{S}$. The second term is the indistinguishability advantage of $\mathcal{S}$, where the maximum is over all adversaries $\mathcal{A}$ that have an (r,s,t) complexity.

Experiment $\mathbf{Exp}^{ind}_{\mathcal{A},\mathcal{S}}[r,s,t]$:

- *Setup phase*

  $(k_1,\ldots,k_n) \leftarrow \text{KEYS}(1^k)$.
  Initialize $\mathcal{R}$ with $(k_1,\ldots,k_n)$ and use a SETKEY call to load tag $\mathcal{T}_i$ with key k_i.

- *Training phase*

  $\mathcal{A}$ may perform any number of READERINIT and TAGINIT calls without exceeding r and t calls respectively, communicate and compute without exceeding s steps overall, and compromise at most $n-2$ tags using SETKEY calls.

- *Challenge phase*

  - $\mathcal{A}$ selects two tags $\mathcal{T}_0$ and $\mathcal{T}_1$ that have not been compromised by SETKEY messages. These tags are removed from the current tag set. $\mathcal{A}$ is then given access to a tag oracle.
  - $b \overset{R}{\leftarrow} \{0,1\}$. The oracle provides $\mathcal{A}$ access to $\mathcal{T}_b$.
  - $\mathcal{A}$ may perform any number of READERINIT and TAGINIT calls without exceeding r and t calls respectively, communicate and compute without exceeding s steps overall, and compromise any tag in the current tag set using SETKEY calls, but *not* $\mathcal{T}_b$.
  - $\mathcal{A}$ outputs a guess bit b'.

  Exp is successful if $b = b'$.

Figure 2: Basic privacy experiment for the RFID system $\mathcal{S}$ = (KEYS, $\mathcal{R}$, $\{\mathcal{T}_i\}$).

5. Forward Privacy for RFID tags

A *key-evolving RFID system* $\mathcal{FS}$ = (KEYS, UPDATE, $\mathcal{R}$, $\{\mathcal{T}_i\}$) consists of a reader $\mathcal{R}$, a set of n tags $\{\mathcal{T}_i\}$, a key generation function KEYS and a *key update* function UPDATE which can be used to update the key of any tag $\mathcal{T}_i$. The scheme then evolves into stages, and in stage j the reader and $\mathcal{T}_i$ use a key denoted by k_i^j. Each stage key k_i^j is obtained from the previous key k_i^{j-1} with the application of the deterministic update algorithm as in $k_i^j \leftarrow \mathcal{FS}.\text{UPDATE}(\mathcal{T}_i, k_i^{j-1})$.

When such update operation is performed successfully, the old key is erased from the memory of the tag and the next stage key is implanted, provided the old key was also known to the reader. Otherwise, the update will be rejected and the tag will use the previous key. Also, an adversary using the UPDATE functionality is not given access to the new key. Access to the key can be obtained only through a SETKEY command which can be thought as compromising the tag (see also [25] for more on this functionality). Concrete instantiations of this update process will be given in subsequent sections.

Experiment $\mathbf{Exp}^{fsind}_{\mathcal{B},\mathcal{FS}}[r,s,t]$:

- *Setup phase*

  $(k_1^0,\ldots,k_n^0) \leftarrow \text{KEYS}(1^k)$.
  Initialize $\mathcal{R}$ with $(k_1^0,\ldots,k_n^0)$ and use a SETKEY call to load tag $\mathcal{T}_i$ with key k_i^0.

- *Training phase*

  $\mathcal{B}$ may perform any number of READERINIT and TAGINIT calls without exceeding r and t calls respectively, communicate, compute and UPDATE without exceeding s steps overall, and compromise at most $n-2$ tags using SETKEY calls.

- *Challenge phase*

  - $\mathcal{B}$ selects two tags $\mathcal{T}_0$ and $\mathcal{T}_1$ that have not been compromised by SETKEY messages and specifies two values i and j with $j < i$. $\mathcal{B}$ is then given access to a tag oracle.
  - $b \overset{R}{\leftarrow} \{0,1\}$. The oracle updates the keys of *both* tags using UPDATE operations until period i is reached. These tags are removed from the current tag set. $\mathcal{B}$ is then given access to $\mathcal{T}_b^j$ and the i-th period keys of tags $\mathcal{T}_0$, $\mathcal{T}_1$.
  - $\mathcal{B}$ may perform any number of READERINIT and TAGINIT calls without exceeding r and t calls respectively, communicate, compute and UPDATE without exceeding s steps overall, and compromise any tag in the current tag set using SETKEY calls, but *not* $\mathcal{T}_b^j$.
  - $\mathcal{B}$ outputs a guess bit b'.

  Exp is successful if $b = b'$.

Figure 3: Forward privacy experiment for the RFID system $\mathcal{FS}$ = (KEYS, UPDATE, $\mathcal{R}$, $\{\mathcal{T}_i\}$).

To define privacy under key knowledge, notice that privacy of tag $\mathcal{T}_i$'s transactions under k_i^j must be maintained even if the adversary is in possession of k_i^k, for $k > j$. In general, *forward privacy* means that even if a secret key is compromised at the present (or the tag acquires a new owner), all past transactions (with prior owners) still remain secure. This is why we divide time in *key evolving periods* trying to ensure that privacy is ensured across *different ownership* periods. To capture this, in Figure 3 we define an experiment associated with an adversary algorithm $\mathcal{B}$.

The goal of this experiment is to formalize the notion of forward privacy when a tag changes owner. The keys

of the two tags selected by adversary $\mathcal{B}$ in the challenge phase are updated successively until period i is reached. Then one of the tags is selected at random and is given to the adversary along with the j-th period tag key while the i-th period keys are kept secret. The idea is that if the system is secure then the adversary will not be able to distinguish among the i-th period tags. This will ensure that privacy of *past* owners is guaranteed even if a tag has been compromised or changed hands.

The following definitions (see also [21, 25]) attempt to capture this:

$$\mathbf{Adv}^{fsind}_{\mathcal{B},\mathcal{FS}} = 2 \cdot \Pr[\mathbf{Exp}^{fsind}_{\mathcal{B},\mathcal{S}} = success] - 1$$

$$\mathbf{Adv}^{fsind}_{\mathcal{FS}}[r,s,t] = \max_{\mathcal{B}}\{\mathbf{Adv}^{fsind}_{\mathcal{B},\mathcal{S}}\}.$$

The first term is the *forward security indistinguishability* advantage of $\mathcal{B}$ in attacking $\mathcal{FS}$. The second term is the indistinguishability advantage of $\mathcal{FS}$, where the maximum is over all adversaries $\mathcal{B}$ that have an (r,s,t) complexity.

5.1. A generic construction with a forward secure PRG

We now demonstrate how such a forward secure RFID system can be constructed when the UPDATE functionality is achieved using a forward secure pseudorandom bit generator (FS-PRG).

A stateful generator $\mathcal{GEN} = (\text{KEY}, \text{NEXT}, k, l)$ is determined by a pair of algorithms KEY, NEXT and two positive integers k, l [32]. Integer l is a limit on the maximum number of k-bit output blocks we want the generator to produce. The KEY generation algorithm provides the initial seed for the generator while the NEXT algorithm, upon input of the current state, returns an output block Out_i and an s-bit string St_i that represents the next state of the generator (Figure 4).

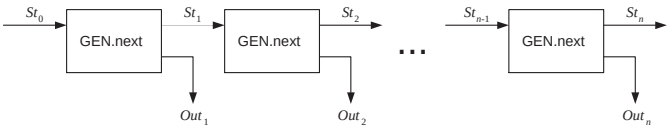

Figure 4: Operation of a forward secure pseudorandom bit generator

A standard pseudo-random generator is considered secure if its output is indistinguishable from a random string [33]. For a generator to be *forward secure*, we require that the past generated output bits remain secure even if an adversary had obtained its current state. This is captured by the experiment of Figure 5.

Adversary $\mathcal{C}$ is allowed to choose when it wants to break in. Eventually it has to decide whether the blocks it has been given are randomly produced bits (case for $b = 0$) or the output of the generator (case for $b = 1$). In both cases, the state of the generator advances properly with respect to its operation. Such adversary $\mathcal{C}$ is considered successful if she can distinguish between the two cases. These cases are considered in the following definitions:

Experiment $\mathbf{Exp}^{fsprg-b}_{\mathcal{C},\mathcal{GEN}}$:

- *Setup phase*

  $St_0 \xleftarrow{R} \text{KEY}(1^s)$

- *Training and Challenge phase*

  – $b \xleftarrow{R} \{0,1\}$. $\mathcal{C}$ is fed with output blocks from the generator until it decides to "break in", in stage $i < l$. The oracle updates the state of the generator until stage i is reached as follows:

  Repeat

  $(Out_i, St_i) \leftarrow \text{NEXT}(St_{i-1})$

  if $(b == 0)$ $Out_i \xleftarrow{R} \{0,1\}^k$

  Until Stage i

  – $\mathcal{C}$ is given access to state St_i.

  – $\mathcal{C}$ outputs a guess bit b'.

Exp is successful if $b = b'$.

Figure 5: Forward privacy experiment for PRG

$$\mathbf{Adv}^{fsprg}_{\mathcal{C},\mathcal{GEN}} = \Pr[\mathbf{Exp}^{fsprg-1}_{\mathcal{C},\mathcal{GEN}} = 1] - \Pr[\mathbf{Exp}^{fsprg-0}_{\mathcal{C},\mathcal{GEN}} = 1]$$

$$\mathbf{Adv}^{fsprg}_{\mathcal{GEN}}[t] = \max_{\mathcal{C}}\{\mathbf{Adv}^{fsprg}_{\mathcal{C},\mathcal{GEN}}\}.$$

The first term is the *fsprg advantage* of $\mathcal{C}$ attacking $\mathcal{GEN}$. The second term is the fsprg advantage of $\mathcal{GEN}$, where the maximum is over all adversaries $\mathcal{C}$ that have time-complexity at most t.

The construction

We now show how a key evolving RFID system can be designed given any secure RFID system and a forward secure PRG. The initial key k^0_i for any tag $\mathcal{T}_i$ corresponds to the initial state of the generator. The "key" corresponding to stage j is a triple $K^j_i = (j, k^j_i, St_j)$ indicating the stage j, the actual key k^j_i used to access the tag, and state information St_j which will be used to generate the next-stage key. The UPDATE function of the key-evolving RFID system $\mathcal{FS} = (\text{KEYS}, \text{UPDATE}, \mathcal{R}, \{\mathcal{T}_i\})$ is given below:

$\mathcal{FS}.\text{UPDATE}(\mathcal{T}_i, j, k^j_i, St_j)$

$(k^{j+1}_i, St_{j+1}) \leftarrow \mathcal{GEN}.\text{NEXT}(St_j)$

Return $(j+1, k^{j+1}_i, St_{j+1})$

At this point we are assuming that both the reader and each tag share the FS-PRG's state but we don't go into details how this may happen in real life. Here, we want to argue that if the reader and each tag can securely update their state based on a FS-PRG then the system remains

secure and no usable information leaks. Later, in Section 6, we will show how to realize this in practice through an update protocol that involves both a reader and a tag.

In the theorem that follows we show that that the key-evolving RFID system $\mathcal{FS}$ remains secure as long as the underlying RFID system $\mathcal{S}$ is secure and the generator is forward secure.

Theorem 1. *Let $\mathcal{S} = (\text{KEYS}, \mathcal{R}, \{\mathcal{T}_i\})$ be a secure RFID system where keys have length k, and $\mathcal{GEN} = (\text{KEY}, \text{NEXT}, k, l)$ a forward secure PRG. Let $\mathcal{FS}$ be the key-evolving RFID system associated with $\mathcal{S}$ and $\mathcal{GEN}$. Then*

$$\boldsymbol{Adv}^{fsind}_{\mathcal{FS}} \leq \boldsymbol{Adv}^{fsprg}_{\mathcal{GEN}} + l \cdot \boldsymbol{Adv}^{ind}_{\mathcal{S}}$$

Proof: The proof follows the techniques that establish the notion of forward security in private key cryptosystems [32]. So, let $\mathcal{B}$ be an attacker that tries to break the forward security of the key-evolving RFID scheme $\mathcal{FS}$. Our goal is to upper bound the advantage $\mathbf{Adv}^{fsind}_{\mathcal{B},\mathcal{FS}}$. For that we will design an adversary $\mathcal{C}$ attacking the forward security of the generator, and an adversary $\mathcal{A}$ attacking the basic RFID secheme $\mathcal{S}$. Then we will bound the fsind-advantage of $\mathcal{B}$ in terms of ind-advantage of $\mathcal{A}$ and the fsprg-advantage of $\mathcal{C}$.

The adversary $\mathcal{C}$

The goal of $\mathcal{C}$ is to distinguish between a truly random sequence of blocks and one that is output from the generator. For this, it will simulate the experiment $\mathbf{Exp}^{fsind}_{\mathcal{B},\mathcal{FS}}$ by letting the output blocks it receives used as the keys of the challenge RFID tags in the $\mathcal{FS}$ scheme. $\mathcal{C}$ will test whether or not $\mathcal{B}$ succeeds on the given sequence of blocks. If so, it will bet that the block sequence is pseudorandom, otherwise it is a random one.

Notice that the experiments $\mathbf{Exp}^{fsprg-1}_{\mathcal{C},\mathcal{GEN}}$ and $\mathbf{Exp}^{fsind}_{\mathcal{B},\mathcal{FS}}$ are identical since they both use the output bits of the generator. Hence

$$\Pr[\mathbf{Exp}^{fsprg-1}_{\mathcal{C},\mathcal{GEN}} = 1] = \mathbf{Adv}^{fsind}_{\mathcal{B},\mathcal{FS}} \qquad (1)$$

The adversary $\mathcal{A}$

We design an adversary $\mathcal{A}$ attacking the basic RFID scheme $\mathcal{S}$ of [25]. The goal of $\mathcal{A}$ is to distinguish between two tags $\mathcal{T}_0$ and $\mathcal{T}_1$. For this, it uses the adversary $\mathcal{B}$, but on a sequence of random strings as keys, rather than the keys obtained via the pseudo-random generator as follows:

$St_0 \leftarrow \mathcal{GEN}.\text{KEY}$
$i \leftarrow 0$
Repeat
  $i \leftarrow i + 1$
  $(Out_i, St) \leftarrow \mathcal{GEN}.\text{NEXT}(St_{i-1})$
  $k_i \xleftarrow{R} \{0,1\}^k$
  $K_i \leftarrow (i, k_i, St_i)$
Until (guessed stage)

For this, it has to make a guess on the stage at which $\mathcal{B}$ decides to attack. Then it uses the results of $\mathcal{B}$ on the stage j tags, as a as the answer to its own challenge. Since the total number of stages implied by the generator is l, we have $\mathbf{Adv}^{ind}_{\mathcal{A},\mathcal{S}} = \Pr[\mathbf{Exp}^{fsprg-0}_{\mathcal{C},\mathcal{GEN}} = 1]/l$ or that

$$\Pr[\mathbf{Exp}^{fsprg-0}_{\mathcal{C},\mathcal{GEN}} = 1] = l \cdot \mathbf{Adv}^{ind}_{\mathcal{A},\mathcal{S}} \qquad (2)$$

Combining Equations (1) and (2), we get

$$\begin{aligned} \mathbf{Adv}^{fsind}_{\mathcal{B},\mathcal{FS}} &= \Pr[\mathbf{Exp}^{fsprg-1}_{\mathcal{C},\mathcal{GEN}} = 1] \\ &= \mathbf{Adv}^{fsprg}_{\mathcal{C},\mathcal{GEN}} + \Pr[\mathbf{Exp}^{fsprg-0}_{\mathcal{C},\mathcal{GEN}} = 1] \\ &= \mathbf{Adv}^{fsprg}_{\mathcal{C},\mathcal{GEN}} + l \cdot \mathbf{Adv}^{ind}_{\mathcal{A},\mathcal{S}} \end{aligned}$$

□

5.2. The case for PRFs

A forward secure pseudorandom generator (FS-PRG) can be based on any pseudorandom bit generator that stretches a short seed into a longer sequence of bits. Similarly, an FS-PRG can be derived using a pseudorandom function (PRF). This will provide the necessary connection with Section 6 that gives a realistic protocol for forward privacy in RFID tags.

A pseudorandom function F is a mapping $\{0,1\}^k \times \{0,1\}^l \rightarrow \{0,1\}^L$ of k-bit keys and l-bit input strings to L-bit output strings. A PRF F is considered secure [34] if its indistinguishable from a random function with domain $\{0,1\}^l$ and range $\{0,1\}^L$. Such a PRF can be easily converted into a standard pseudorandom bit generator $G : \{0,1\}^s \rightarrow \{0,1\}^{k+s}$ and then a stateful generator $\mathcal{GEN}$ as follows:

Let S be an s-bit string and r a random l-bit string. We let $G(S)$ denote the first $k + s$ bits of the sequence $F(S, r) \parallel F(S, r + 1) \parallel F(S, r + 2) \parallel \ldots$, where '$\parallel$' denotes concatenation and the increments of r are taken modulo 2^l. This defines a standard PRG $G : \{0,1\}^s \rightarrow \{0,1\}^{k+s}$.

The stateful generator $\mathcal{GEN} = (\text{KEY}, \text{NEXT}, k, l)$ associated to F is given by the following construction:

$\mathcal{GEN}.\text{KEY}$
  $St_0 \xleftarrow{R} \{0,1\}^s$
  Return St_0

and

$\mathcal{GEN}.\text{NEXT}(St)$
  $S \leftarrow G(St)$
  Return $(S[1,k], S[k+1,s])$

where the notation $[i, j]$ denotes the substring of S consisting of the bits in positions i through j. The following theorem from [32] states the security of the stateful generator $\mathcal{GEN}$ defined above to the security of the PRF F.

Theorem 2. *Let $\mathcal{GEN} = (\text{KEY}, \text{NEXT}, k, l)$ be the stateful generator associated with pseudorandom function F. Then*

$$\boldsymbol{Adv}^{fsprg}_{\mathcal{GEN}} \leq l \cdot \boldsymbol{Adv}^{ind}_{\mathcal{F}}$$

This theorem provides the connection with the notion of a key-evolving RFID system that we developed in the previous sections. In light of the results of Theorem 1, this also says that a key evolving RFID system can be made secure as long as the underlying RFID system $\mathcal{S}$ and the pseudorandom function F are secure.

6. Ownership Transfer made Easy

So far we have showed that if the underlying basic system is secure (Section 4) and the reader and a tag share some state S which can both update using either a PRG or a PRF (Section 5), then the resulting system is forward secure. Thus, even if some adversary has compromised a tag now, past transactions with this tag remain secure.

However, this discussion implicitly assumed that UPDATE creates the next key *concurrently* at both reader and tag without going into the details as to how these entities co-ordinate on this update process. In this section we will show how this can happen in real life without the involvement of any third party. In particular, we will explain how the current owner can update the tag secret – and eventually take complete control of a tag – so that prior owners cannot trace the tag anymore.

The remainder of this section is organized as follows. In Sections 6.1-6.2, we discuss implementation details and ways to obtain tag secrets. In Section 6.3, we present a basic protocol to identify and authenticate a tag that is secure in the sense of [25]. Then, in Section 6.4, we show how to make it forward secure using a simple update functionality; in the same section we also discuss the mechanics of ownership transfer. Section 6.5 focuses on controlled delegation and authorization recovery. Finally, Section 6.6 analyzes the security properties achieved.

6.1. Implementation details

From an implementation point of view, we have already showed how a key-evolving RFID system can be realized using pseudo-random number generators and pseudo-random functions. In what follows, for presentation simplicity, we will rely on the existence of a secure hash function F that is both one-way and collision resistant. Hash functions, particularly keyed ones, have been used as pseudo-random functions in public key cryptosystems. However, most of the approaches to ownership transfer that rely on hash functions and other more expensive cryptographic operations do not comply with the EPC Global Class-1 Gen-2 (C1G2) standard for passive RFID tags because they require a number of gates that typically are not available for passive tags. Passive tags can accommodate roughly 3K gates while hash functions typically require 8-10K gates (see [36] and the references therein).

However, our protocol can also be instantiated using simple XOR and PRG operations, as per the work in Section 5, which are easily implemented on low-cost RFID tags. Our protocols can use a 128-bit PRG that is provably secure and requires less than 2K gates ([36, 37, 38]). This can be done by replacing each hash function call $F(K, R_1, \ldots)$ with a call to the PRG seeded by the key K maintained in the tag and all fresh numbers $R_1, \ldots$ exchanged between the tag and the reader. Thus, from a security point of view the use of a 128-bit PRG is well supported as long as no form of synchronization is needed between the reader and the tag, other than the key shared between the two. Although the EPC C1G2 standard offers support for 16-bit pseudo random number generation, it is not considered secure because of its vulnerability to brute-force attacks. Hence the use of 128-bit PRNGs with fast response times and a small gate footprint (for example, [39] and [40] have developed hardware implementations which require 1453 and 1566 logic gates, respectively) are compliant with low-cost passive tags, they are more secure and a better option for large-scale deployments of these tags. However, here, a hash function F will be our main building block as in many past works [3, 20, 22, 21, 23, 24].

We denote by K_O the key shared between the reader of the current owner and the tag. K_O allows the tag owner to perform all tag related operations including identification, key-update and controlled delegation. In the case when tag access has been *delegated*, a different key is used to query and identify the tag. This is denoted by K_D and its sole purpose is to allow the delegate to query the tag – all other operations remain the privilege of the current owner. This functionality is enabled by updating the `DELEGATED` bit of the tag during controlled delegation. When `DELEGATED=TRUE`, tag identification is performed using key K_D. In this case, a `COUNTER` (see [24] for a similar approach) also controls the number of accesses permitted by the current owner to the delegate.

Table 1 summarizes the notation that will be used in the remainder of the paper.

6.2. Acquiring a tag

Here we consider the situation where a user has acquired a set of tagged products and wants to interact with them either at home or at a business-like environment through the use of personal RFID readers. As tag responses will be made indistinguishable from random data in order to defend against tracking attacks, readers should be capable of interpreting and making sense of these responses. However, for this to happen, knowledge of the secret key stored in each tag is required.

When buying a large number of tagged items (eg. transferring stock from one company to another) the easiest way to acquire their keys is to assume the existence of a secure channel between the buyer and the current owner (say the manufacturer). In this case, keys can be transmitted in a secure way and made available to the buyer. When considering checkout of just a few items at a point-of-sale, users can use their smart phones to securely connect to a check-out server and receive the key on the fly (a similar method can also be applied between owners during ownership transfer). Alternatively, the secret key can be printed

Notation	Meaning
Owner_i	i-th owner of the tag.
K_O	A secret key possessed by tag and shared with the reader of current owner. This key enables the execution of all tag-control operations.
K_D	This key allows the delegate to query and identify the tag. It is set to a special value NIL when the tag is not delegated.
`DELEGATED`	Defines the state of the tag. It is set to 1 if the tag is delegated, otherwise it is set to 0. Notice that the same functionality can be achieved by testing whether K_D is NIL.
`COUNTER`	Decreased every time a tag is accessed using K_D. When `COUNTER` reaches 0, delegated access is not longer possible.
M_1, M_2	Concatenation of messages M_1 and M_2.
$F_K(M)$	Application of a keyed hash function F on message M and key K.
N_R, N_T	Nonce identifiers (random numbers used once) sent by reader and tag, respectively.

Table 1: Notation and Terminology

in a receipt using a 2D barcode and have the phone scan the receipt.

All the above (this is not necessarily an exhausted list) suggest a number of ways to acquire the secret key in a tag. However, none of these methods solves the ownership problem since the key is basically *shared* between the buyer and the manufacturer (previous owner). As the previous owner can still read and scan all its tagged products, it is not hard to see that the privacy of the buyer is still at risk. Hence what is needed is a method to *update* the secret key in the tag when a product changes hands. This is the main issue in ownership transfer and the topic of Section 6.4.

6.3. Querying a tag

Once the key of a tag is known to the user, a user's reader can interact with the tag through the protocol shown in Figure 6.

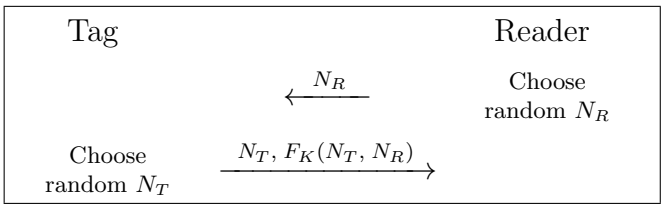

Figure 6: Basic identification protocol for interacting with a tag. The tag is accessed using a key K which is either the owner's key K_O or a delegated key K_D.

In this protocol the reader first sends a nonce N_R and the tag responds back with $\langle N_T,\ F_K(N_T,\ N_R)\rangle$, where N_T is another nonce chosen by the tag. This is not a novel protocol but we will extend it to achieve ownership transfer and forward privacy. Similar protocols can be found in a number of previous works ([8, 9, 35]) and can be shown to be secure in the sense captured by the experiment of Section 4. This is because tag responses look indistinguishable from random data since the tag always includes a fresh identifier N_T in the evaluation of F and no information leaks because of the one-wayness of F) (for a formal proof the reader is referred to [25]).

Notice that upon receiving $F_K(N_T, N_R)$, the reader must do a linear search in the set of stored keys in order to disambiguate the tag. This is something that can be fixed by enhancing the protocol using the techniques described in [8] and [9] (basically creating a "tree of secrets" and arranging the tags in a tree hierarchy) in order to achieve logarithmic search complexity. Notice, however, that increase in efficiency may result in some loss of privacy as was demonstrated in [9] and [10].

In more recent work, [22] and [23] achieve constant time identification by maintaining hash tables of tag IDs. However, the use of hash chains introduces *desynchronization* issues which may either lead to longer, linear time responses when synchronization is lost or tracking by timing the server's responses when trying to disambiguate the tag. The authors in [24] avoid this issue altogether by introducing the *fixed target* assumption: during ownership transfer, there is always a target tag, which has already been authenticated, such that the reader knows exactly the secret key associated with the tag. While this assumption makes sense when trading or interacting with a small number of items, the issue of efficient tag identification still remains. Here we note down these alternatives and stress that in a personal environment where RFID tags are embedded in everyday items (clothes, shoes, drugs, books, food containers, etc.) scalability is not really a concern.

6.3.1. The case for delegation

One last thing to mention regarding the identification protocol of Figure 6 (not shown in order to avoid overloading the figure) is that the `DELEGATED` bit maintained in the tag controls which of the two keys K_O or K_D will be used in tag identification. If the bit is set during controlled delegation (Section 6.5), the temporary key K_D is selected so that the delegate also has access to the tag. In this case, an associated `COUNTER` is decreased with every identification until it reaches 0. When this happens, `DELEGATED` is set to `FALSE`, K_D is erased (set to `NIL`) and tag identification is again only possible using the owner's key K_O (see also Section 6.5).

6.4. Updating the key of a tag

As explained in Section 6.2, acquiring the secret key of a tag suffers from two major shortcomings. First, it implicitly assumes that all owners (both current and old ones) must trust the same central database which by default controls all tag secrets. Second, it doesn't really allow a user to take complete control of the tag by "planting"

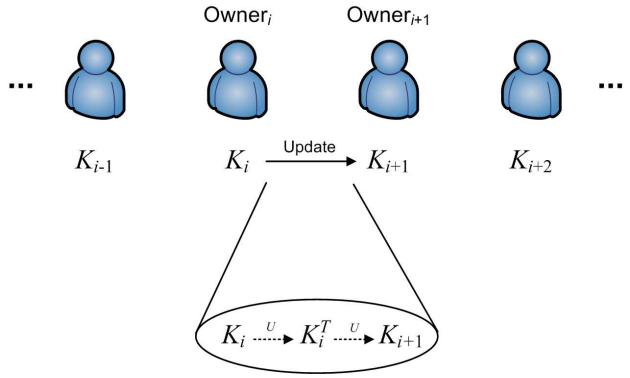

Figure 7: Sequence of owners and tag secrets. K_i^T is a temporary key used to hide the value of K_i or K_{i+1} to the adjacent owner (refer to Section 6.4.1). The dotted arrows indicate an arbitrary number of updates.

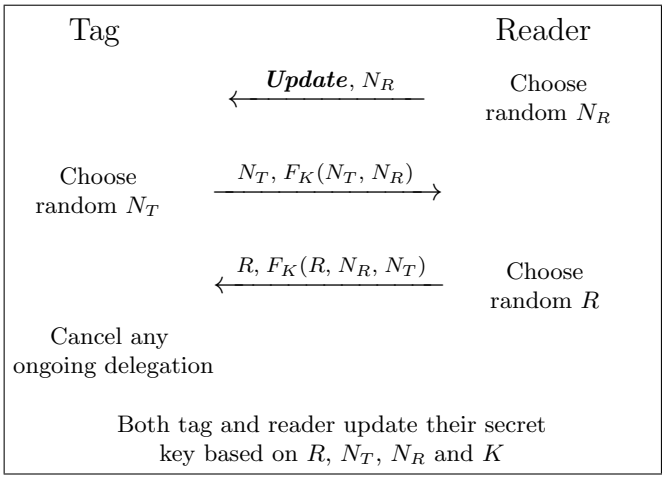

Figure 8: Updating the tag's key to a new one.

a new secret key to it. Thus the vision of the Internet of Things is severely limited as it does not allow for the possibility of transferring objects among owners without violating the privacy of anyone involved in this process. The scheme we envision here is shown in Figure 7.

This model is an extension of the basic, traditional model which consists of just one owner: Owner_0. This really corresponds to a central database containing the secrets of associated tags. Thus all other users do not really possess a tag but only *share* it with the central database. In the model of Figure 7, tag T owned by Owner_i (say Alice) carries a key K_i, known only to her. When this tag is to be transferred to Owner_{i+1} (say Bob), the new owner has to *update* its key to a new one (K_{i+1}) known only to him.

Currently, two families of key updating protocols can be found in the literature.

- *Centralized approaches*: Here, a Trusted Third Party (TTP) mediates in the transfer of the secret keys between the previous and the new owner (see for example [36] for a recent survey). This solution does not scale well as all requests must go through the TTP but there is a more fundamental concern that makes this approach less desirable: the *complete* trust placed on this central repository of keys. As the TTP possesses all tag secrets, nothing prevents the TTP from tracking and profiling the users carrying tagged objects (or giving away this information). Hence this solution makes sense in closed environments or when the objects belong to the same or collaborating organizations.

- *Private environment*: An isolated environment exists where the adversary cannot monitor all the data exchanged between the tag and the reader. Thus if the adversary ever misses a single message, she will not be able to track the tag anymore as she will fail to access the newly created key. Notice that there is no restriction on the number of times the key-update protocol can be run, which makes it even more difficult for the adversary to follow all tag-reader interactions. Hence we move from an know-it-all adversary in the case of centralized approaches to an omnipresent one, which is much more difficult to realize in practice. This is the approach followed in many recent works [21]-[24] and the one we follow here as well.

We emphasize again that during the update process, Owner_{i+1} interacts only with the tag and *no one* else, thus eliminating the need for trusted third parties. The protocol for updating a tag's secret is shown in Figure 8. This is the same as the basic one (Figure 6) but includes a third step to authenticate the reader. In this last step, the reader computes another nonce R and transmits it along with $F_K(R, N_R, N_T)$. This last part acts as a Message Authentication Code (MAC). If the MAC verification succeeds, the tag accepts the reader as authentic and proceeds to update its key with the help of F as in

$$K_{new} = F_K(N_R, R, N_T, \text{“Update”}).$$

We note again that updating the secret key of the tag must take place in a secure environment. This is necessitated by the fact that the previous owner already has knowledge of the secret key. However, if the previous owner (or an adversary) misses any of the nonces N_R, N_T transmitted in the protocol, she will not be able to get hold of the new key and track the tag anymore. We will come back to this matter when we analyze the security properties offered by this protocol in Section 6.6.

Finally, one last operation performed by the tag is to *cancel* any ongoing delegated access. As the key-update may occur during ownership transfer (see next section) and the new owner may not be willing to allow access to the tagged object, any such pending delegation must be terminated. This is enforced by setting the `DELEGATED` bit to `FALSE` and destroying the delegation key (K_D set to `NIL`).

6.4.1. Ownership Transfer

We now consider the actual mechanics of ownership transfer (OT). This process takes place when the previous owner (Alice) transfers all tag secrets to the new owner (Bob). However, this transfer should take place only *after* Alice has *updated* the key used to access the tag. If this not the case, Bob could use the key to profile and track Alice after possibly recording all the transactions made in the past. Additionally, Bob should also update the secret key after receiving it from Alice since otherwise Alice could also use the key to profile his future interactions with the tag.

If we look back at Figure 7, the above process can be summarized as follows:

1. Owner_i updates its key K_i to a new temporary key K_i^T which hands over to Owner_{i+1}. This prevents Owner_{i+1} from tracing the transactions made with key K_i.
2. Owner_{i+1} also updates K_i^T to K_{i+1} in order to prevent Owner_i from violating its privacy. As mentioned already, this update should take place in an environment where there is no possibility of eavesdropping by the previous owner. As updates incur no overhead, an arbitrary number of them can be executed to ensure that the resulting key K_{i+1} is known only to the new owner. If the previous owner ever misses an update, security of K_{i+1} is guaranteed.

6.5. Controlled Delegation and Authorization Recovery

Controlled delegation occurs when the current owner wishes to delegate (temporary) identification rights to another entity, called the *delegate*. Such a procedure could be used to reduce the computational and communication overhead on the back-end server. For example, if this server is in charge of a very large number of items then assigning/delegating some of these items to a new reader or mobile device may help improve identification time and reduce the communication latency between the reader and the back-end database. Delegation not only provides a solution to these performance issues but can also improve *availability* since even if the central database goes offline due to network failures, offline identification is still possible. Eventually, when the tag is identified for a number of times specified by the current owner, the delegate loses its right to identify the tag; in that case a new request must be made to the current owner.

This protocol (Figure 9) has the same structure as the key-update protocol of Figure 8. The choice of messages is necessitated by the requirement that the tag must be sure the request is fresh, otherwise a delegate could just replay an old request to extend the delegation period.

The reader first issues a '`Delegate`' command along with the maximum number of allowable delegations MAX. In the second message, the tag replies with a fresh number R_T and a MAC of the exchanged parameters. The key K used to validate the MAC is the owner's key K_O.

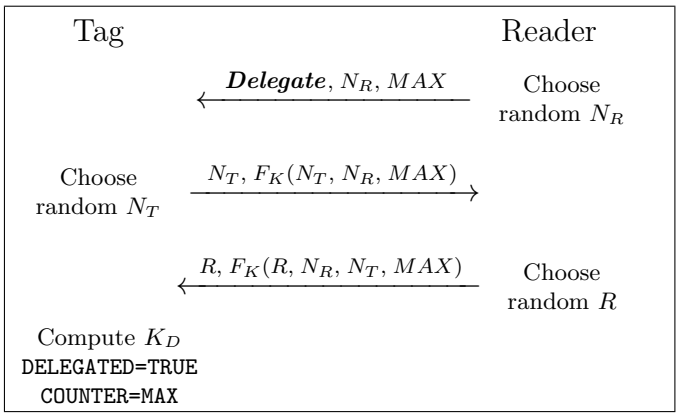

Figure 9: Protocol for controlled delegation. The delegated key K_D is derived from the exchanged values and the owner's secret key K_O.

The role of the third message is to authenticate the reader and the delegation command. In this last step, the reader computes another nonce R and transmits it along with $F_K(R, N_R, N_T, MAX)$. If the verification succeeds, the tag accepts the reader's request as authentic (and fresh) and creates the delegation key K_D by evaluating

$$K_D = F_K(N_R, R, N_T, MAX, \text{“Delegate”}).$$

Additionally, the `DELEGATED` bit is set and the `COUNTER` maintained in the tag is initialized to MAX. The owner also computes K_D and hands it to the delegate using a secure channel and any of the methods outlined in Section 6.2. The delegate now has access to the tag but *only* for querying the tag. As mentioned in Section 6.3, each time the tag is queried using K_D, the `COUNTER` is decreased by 1 until it reaches zero. Once this happens, delegation expires (`DELEGATED` is set to `FALSE`) and K_D is erased. Querying the tag now is again only possible using the owner's key K_O.

6.5.1. Canceling Delegation

To regain complete control of the tag, the current owner may apply any of the following alternatives:

1. Wait until the `COUNTER` reaches zero in which case delegation expires automatically.
2. Update the owner's key using the key-update protocol of Figure 8. As we have already mentioned in Section 6.4, a byproduct of key-update is cancelation of any ongoing delegations.
3. Issue a direct '`Cancel Delegation`' command. The exchange of messages is similar to the one in Figure 9 and is omitted here. When the tag authenticates the MAC received by the reader in the third message, it proceeds to cancel delegation by setting the `DELEGATED` bit to `FALSE` and erasing the key K_D.

6.5.2. Authorization Recovery

Authorization recovery is useful when the current owner must temporarily authorize a previous owner to have access to the tag. This may be the case when the item must

be returned for service and the previous owner is the manufacturer or the shop where the item was purchased.

Authorization recovery can be performed in three ways. The first one is to use the temporary key given by the previous owner when ownership transfer was carried out, and *re-instate* it during authorization recovery. This is the method followed in [22] which, however, entails the danger of the current owner losing control of the tag. This can happen if the previous owner is allowed access to *all* operations within the tag, in which case she can simply update the tag key, thus shutting out the legitimate owner and gaining complete control of the tag.

The second method is to perform complete ownership transfer. This approach is also problematic as it is not clear who is the owner of the tag. If we define ownership of the tag as "possession of the key which allows the execution of major operations on the tag", then both approaches at best allow a tag to be *shared* between owners. But that is the wrong approach to tag ownership.

The method we follow here is similar to the one in [23] and [24]; we treat authorization recovery as a *special case* of controlled delegation. By delegating the authority to identify the tag, the current owner maintains control of the tag while it can still provide access to the previous owner. If more than simple identification is required, then authorization recovery reduces to ownership transfer.

6.6. Security Analysis

In this section we describe how our solution satisfies the security and privacy requirements posed in Section 3.

Impersonation and Replay attacks. The use of the shared secret defends against attacks where an attacker masquerades as a reader or a tag (attack on reader/tag authentication for the purpose of tracking the tag or fooling the reader into accepting a tag as authentic). In the first case, an attacker without knowledge of the secret key has no better strategy than sending random data in the third protocol message of both the key-update and delegation protocols as it has no control over the tag's nonce N_T. Hence the request will be rejected by the tag. In the second case, the only strategy to impersonate a tag and fool a reader is to replay old information. However, in this case a reader will not accept the tag's response as authentic since the attacker can have no control over the reader's nonce N_R. Thus both impersonation and replay attacks are easily dismissed. In the following, we concentrate on the privacy characteristics of ownership transfer.

Confidentiality and Privacy enforcement. All the protocols ensure basic protection against tracking and violation of privacy since no static identifiers are emitted by the tag (recall the use of random, unpredictable numbers and the one-wayness of F). Combining this with the use of a MAC to authenticate responses, we also obtain security against modification attempts. The only way for a previous owner, Owner_i to track the tag (and hence also Owner_{i+1}) is if she knows all important parameters to the key-update process: K, N_R, N_T and R. As long as the previous owner ever misses any of those (and this is a realistic assumption as the update protocol can be run any number of times), she will not be able to compute the new key. This is why we insisted that update requests must take place in a private environment as opposed to the complete trust placed in the case of a TTP.

Forward privacy. The protocol inherits the properties of a forward secure RFID system. Theorems 1 and 2 suggest that if the underlying RFID system and the pseudorandom function are secure, then so is the key-evolving system. The underlying RFID system is secure (in the sense of Section 4) since no fixed identifiers are emitted (see also [25] for a formal proof of this). Similarly, the update procedure in the protocol of Figure 8 and the stateful generator (construction in Section 5.2) are based on the same principle: updating the state using a secure function and a set of nonces.

Forward privacy is about making sure that tag transactions that took place in the past are still protected even if the tag secret is known (or compromised) at the current moment. Thus if the tag has been transferred to a new owner, then transactions of previous owners still remain secure (old owner privacy). To see why forward privacy is preserved, consider a new owner Owner_{i+1} who wants to have access to a tag previously owned by Owner_i (recall Figure 7). If Owner_{i+1} is simply given access to the key K_i used by Owner_i, then it can clearly track the past actions of Owner_i. To prevent this "privacy breach", we insisted that the old owner must *update* its key at least once before it hands it over to the new owner. As a result the key given to the new owner is different from the key used to access the tag by Owner_i. Thus the mechanics of ownership transfer (Section 6.4.1) guarantee that a future owner cannot track the movements of past ones.

Backward privacy. Backward privacy is about making sure that tag transactions of future owners are protected by current owners who possess the tag's key. Thus this notion is equivalent to the new owner privacy requirement. This notion cannot be formally proved as we did for the case of forward privacy since the one-wayness of the pseudorandom function only guarantees that current keys do not leak any information about past keys and not vice versa.

Thus, if an attacker compromises the current tag key, she can track future interactions in which the compromised key is used. However, the mechanics of ownership transfer (Section 6.4.1) also protect future owners from past ones (or compromised keys) as long as the old owner (or adversary) cannot intercept all exchanged messages. When the new owner Owner_{i+1} is handed a key from the previous one, it has to update its key at least once, assuming of course that an adversary cannot eavesdrop on *all* future interactions of the tag (recall the existence of a private environment for key-updates). Thus, if Owner_{i+1} securely updates the key, then future interactions are secured.

In summary, if key privacy is broken at some point during the lifetime of the tag, it can be restored by just one secure key update.

Desynchronization issues. Some early proposals of RFID authentication protocols suffer from *desynchronization* attacks. In such attacks, the adversary sends a large volume of authentication queries to a tag in order to desynchronize it with respect to information stored in the reader. This is typical of protocols that share information (such as hash chains [22], [23]) between tag and reader in order to make key search and tag recognition faster. This additional information can help the reader associate a secret value with a particular tag and thus interpret more easily tag responses. However, when a tag does not authenticate reader responses, it may update its state erroneously thus preventing future interactions with the reader (Denial of Service). Desynchronization may also lead to tracking and loss of privacy, if the shared information acts as a side channel in which an adversary can utilize to track the tag.

"Desynchronization" in the protocol of Figure 8 may occur only if the last protocol message is lost or jammed by some adversary. This may result in the reader updating the key to K_{new} and the tag retaining the old one. However, this can hardly be called desynchronization as it doesn't have any repercussions to the functionality or privacy of the tag: the owner can test this by having its reader query the tag with the new key. In case the tag is not recognized, the tag is still in possession of the old key. To resolve this, the reader can issue another update operation using the old key until the update succeeds. Thus the only requirement is that the reader retains the previous key until the new key has been successfully installed in the tag. Then the old key can be safely deleted from the reader. Similarly, loss of the third message in the protocol of Figure 9 is again not an issue as basically the tag will not generate the delegation key K_D. But that can be fixed by simply issuing again the delegation command.

A summary of the properties achieved by our protocol and a comparison with the most relevant ones is shown in Table 2.

Table 2: Comparison with existing protocols

Properties	[22]	[23]	[24]	This work
Mutual Authentication	✓	✓	$\diamond^{6}$	✓
Anonymity/Privacy	$\diamond^{1}$	$\diamond^{1}$	✓	✓
Forward Security	✓	✓	✓	✓
Backward Security	✓	✓	✓	✓
Replay Resistance	✓	✓	$-^{6}$	✓
De-sync. Resistance	$\diamond^{1,2}$	$-^{5}$	✓	✓
Controlled Delegation	✓	✓	$\diamond^{6}$	✓
Authorization Recovery	$\diamond^{3}$	$\diamond^{3}$	$\diamond^{6}$	✓
Easiness of Delegation	$\diamond^{4}$	$\diamond^{4}$	✓	✓
Server Complexity	$O(1)^{2}$	$O(1)^{5}$	$O(n)^{7}$	$O(n)^{8}$
Tag Complexity (OPs)	3-4	2-3	2-5	1-2
Server Storage (per tag)	$O(m)$	$O(m)$	$O(1)$	$O(1)$
Tag Storage (bits)	$2l + l_m$	$3l + l_c$	$2l + l_c$	$2l + l_c$

✓: Provided $\diamond$: Partially provided $-$: Not provided

m: Length of hash chains
n: Number of tags in system
l_c: Number of bits in counter
l_m: Number of bits in m
l: Security level
OPs: Hash function or PRG operations

1 In a desynchronized state, a side channel is created which can lead to violation of privacy.
2 When the tag is queried more than m times, the server needs to perform a linear search to authenticate tag.
3 Server must restore secret of previous owner. Who is the real owner of the tag?
4 The secret key along with m identifiers must be transferred to new entity.
5 Denial of Service (DoS) is possible if an attacker attempts to query a tag beyond a system specific threshold value.
6 An old key may be planted to a tag and a delegate can extend the period of tag delegation.
7 Authors assume that there is always a target tag, which has been authenticated already.
8 Tag identification can become $O(\log n)$ using tree-based protocols.

7. Conclusions

In this work we have formalized the notion of *key-evolving* RFID systems using PRGs and PRFs that can be proven to be forward secure. We have derived concrete results based on the security of the underlying primitives and presented a simple, yet realistic protocol that allows users to securely own and transfer RFID tags. The protocol ensures forward and backward privacy and protection against a number of attacks such as tracking, impersonation, desynchronization and so on. Additionally, the protocol is very simple to use as it requires only one tag-reader interaction and no reliance on trusted third parties. Furthermore it can be run as many times as necessary. Using the protocol, ownership transfer, controlled delegation and authorization recovery are possible without violating the privacy of past and future owners.

8. Acknowledgments

The author would like to thank the anonymous reviewers for their useful comments that helped improve the paper considerably.

References

[1] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, Imrich Chlamtac, "Internet of things: Vision, applications and research challenges," In Ad Hoc Networks, Volume 10, Issue 7, Pages 1497-1516, September 2012.
[2] Luigi Atzori, Antonio Iera, Giacomo Morabito, "The Internet of Things: A survey." In *Computer Networks*, 54 (15), p. 27872805, 2010.

[3] Ari Juels, "RFID security and privacy: A research survey," IEEE Journal on Selected Areas in Communication, 2006.

[4] Tassos Dimitriou, "RFID Security: Attacks and countermeasures," Book Chapter in RFID Security: Techniques, Protocols and System-On-Chip Design, Paris Kitsos, Yan Zhang (eds), Springer Verlag, 2008.

[5] Quan Z. Sheng, Sherali Zeadally, Aikaterini Mitrokotsa, Zakaria Maamar. "RFID technology, systems, and applications," In Journal of Network and Computer Applications, Volume 34, Issue 3, May 2011, Pages 797-798.

[6] http://www.nfc-forum.org

[7] Tassos Dimitriou,"Key Evolving RFID Systems: Forward Privacy and Ownership Transfer of RFID Tags," In 2014 IEEE RFID Technology and Applications (RFID-TA) Conference, 2014.

[8] David Molnar, Andrea Soppera and David Wagner, "A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags," in *SAC*, 2005.

[9] Tassos Dimitriou, "A Secure and Efficient RFID Protocol that could make Big Brother (partially) Obsolete," in *4th IEEE Intern. Conference on Pervasive Computer and Communications (PerCom)*, 2006.

[10] Karsten Nohl, David Evans (2006) "Quantifying Information Leakage in Tree-Based Hash Protocols," in *8th International Conference on Information and Communications Security (ICICS)*, USA.

[11] K. Osaka, T. Takagi, K. Yamazaki, and O. Takahashi. "An Efficient and Secure RFID Security Method with Ownership Transfer," in *IEEE Intern. Conf. on Computational Intelligence and Security (CIS'06)*, 2006.

[12] C. H. Lim and T. Kwon. "Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer," In *8th Intern. Conference on Information and Communications Security (ICICS '06)*, December 2006.

[13] Raphael C.-W. Phany, Jiang Wu, Khaled Oua, Douglas R. Stinson. "Privacy Analysis of Forward and Backward Untraceable RFID Authentication Schemes", In *Wireless Personal Communications*, Volume 61, Issue 1, November 2011.

[14] C. Berbain, O. Billet, J. Etrog, H. Gilbert. "An efficient forward private RFID protocol," in *Proc. of the 16th ACM conference on Computer and Communications Security*, 2009.

[15] M. Ohkubo, K. Suzuki and S. Kinoshita "Cryptographic Approach to Privacy-friendly Tags," in *RFID Privacy Workshop*, MIT, Massachusetts, USA, 2003.

[16] O. Billet, J. Etrog, and H. Gilbert, "Lightweight Privacy Preserving Authentication for RFID Using a Stream Cipher," in *Proceedings of Fast Software Encryption (FSE 2010)*, LNCS 6147, pp. 55–74, 2010.

[17] Mohammad Reza Sohizadeh Abyaneh. "On the privacy of two tag ownership transfer protocols for RFIDs," in *International Conference for Internet Technology and Secured Transactions*, 2011.

[18] K. Elkhiyaoui, E. Blass, R. Molva. "ROTIV: RFID Ownership Transfer with Issuer Verification," in the *RFIDSec2011*, Amherst, USA, June 2011.

[19] C. Chen, Y. Lai, C. Cheng Chen, Y. Deng, Y. Hwang. "RFID Ownership Transfer Authorization Systems Conforming EPCglobal Class-1 Generation-2 Standards," in *International Journal of Network Security*, Vol.13, No.1, July 2011.

[20] P. Peris-Lopez, J.C. Hernandez-Castro, J.M.Estevez-Tapiador, Tieyan Li and Yingjiu Li "Vulnerability analysis of RFID protocols for tag ownership transfer," in *Computer Networks*, 2010.

[21] T. Dimitriou. "RFID-DOT: RFID Delegation and Ownership Transfer made simple." In *4th International Conference on Security and Privacy for Communication Networks*, SECURECOMM 2008.

[22] B. Song and C.J. Mitchell. "Scalable RFID security protocols supporting tag ownership transfer." In *Computer Communications*, Vol. 34, No. 4, 2011, p. 556–566.

[23] Albert Fernandez-Mir, Rolando Trujillo-Rasua, Jordi Castella-Roca, and Josep Domingo-Ferrer. "A Scalable RFID Authentication Protocol Supporting Ownership Transfer and Controlled Delegation." In Juels, A., Paar, C. (eds.) RFIDSec. LNCS, Volume 7055, Springer (2011).

[24] C. Yu Ng, W. Susilo, Y. Mu,R. Safavi-Naini. "Practical RFID ownership transfer scheme. *Journal of Computer Security*, 19 (2), 319–341, 2011.

[25] A. Juels and S. Weis. "Defining Strong Privacy for RFID," in *Proceedings of the 5th IEEE International Conference on Pervasive Computing and Communications Workshops*, 2007. Also in Cryptology ePrint archive, www.iacr.org

[26] Gildas Avoine. "Adversary Model for Radio Frequency Identification," in Cryptology ePrint archive, www.iacr.org

[27] Ivan Damgård and Michael Østergaard Pedersen. "RFID Security: Tradeoffs between Security and Efficiency," in Cryptology ePrint archive, www.iacr.org, July 2006.

[28] S. Vaudenay, "On privacy models for RFID." In *the 13th International Conference on Theory and Application of Cryptology and Information Security*, ASIACRYPT '07.

[29] T. Van Le, M. Burmester, and B. de Medeiros, "Universally Composable and Forward-Secure RFID Authentication and Authenticated Key Exchange," in *Proceedings of ASIACCS'07*, ACM Press, pp. 242-252, 2007.

[30] R. Deng, Y. Li, M. Yung and Y. Zhao, "A New Framework for RFID Privacy," in *Proceedings of Esorics 2010*, LNCS 6345, pp. 1-18, 2010.

[31] J. Hermans, A. Pashalidis, F. Vercauteren, and B. Preneel, "A New RFID Privacy Model," In *European Symposium on Research in Computer Security (ESORICS 2011)*.

[32] Mihir Bellare and Bennet Yee. "Forward Security in Private-Key Cryptography", in *Topics in Cryptology - CT-RSA 03*, Lecture Notes in Computer Science Vol. 2612, M. Joye ed., Springer-Verlag, 2003.

[33] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits," SIAM Journal on Computing, Vol.‘ 13, No.‘ 4, 850-864, November 1984.

[34] O. Goldreich, S. Goldwasser, S. Micali, "How to construct random functions," Journal of the ACM, Volume 33, Issue 4, October 1986.

[35] David Molnar and David Wagner, "Privacy and Security in Library RFID Issues, Practices, and Architectures," *Conference on Computer and Communication Security*, 2004.

[36] S. Sundaresan, R. Doss, W. Zhou, S. Piramuthu. "Secure ownership transfer for multi-tag multi-owner passive RFID environment with individual-owner-privacy," In *Computer Communications* 55 (2015) 112-124.

[37] M. Burmester, B.D. Medeiros, "RFID Security: Attacks, Countermeasures and Challenges." In *5th RFID Academic Convocation, The RFID Journal Conference*, 2007.

[38] M. Burmester, J. Munilla, "A Flyweight RFID Authentication Protocol." In *Workshop on RFID Security (RFIDSec)*, 2009.

[39] H. Lee and D. Hong, "The tag authentication scheme using self-shrinking generator on RFID system." In *Transactions on Engineering, Computing and Technology* . Vol. 18, 5257, 2006.

[40] P. Peris-Lopez,J.C. Hernandez-Castro,J.M. Estevez-Tapiador, and A. Ribagorda, "LAMED - A PRNG for EPC Class-1 Generation-2 RFID specification." In *Comput. Stand. Interfaces* 31, 1, 8897.