# A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags (Extended Abstract)

David Molnar*, Andrea Soppera**, and David Wagner***

UC Berkeley, British Telecom, and UC Berkeley

**Abstract.** The ability to link two different sightings of the same Radio Frequency Identification (RFID) tag enables invasions of privacy. The problem is aggravated when an item, and the tag attached to it, changes hands during the course of its lifetime. After such an *ownership transfer*, the new owner should be able to read the tag but the old owner should not.

We address these issues through an RFID *pseudonym protocol*. Each time it is queried, the RFID tag emits a different pseudonym using a pseudo-random function. Without consent of a special Trusted Center that shares secrets with the tag, it is infeasible to map the pseudonym to the tag's real identity. We present a scheme for RFID pseudonyms that works with legacy, untrusted readers, requires only one message from tag to reader, and is scalable: decoding tag pseudonyms takes work logarithmic in the number of tags. Our scheme further allows for *time-limited delegation*, so that we can give an RFID reader the power to disambiguate a limited number of pseudonyms without further help from the Trusted Center. We show how RFID pseudonyms facilitate the transfer of ownership of RFID tags between mutually distrustful parties.

Our scheme requires only limited cryptographic functionality from the tag: we need a pseudo-random function (PRF) and the ability to update tag state or to generate random numbers. Tag storage and communication requirements are modest: we give example parameters for a deployment of one million tags in which each tag stores only 128 bits, makes 6 PRF evaluations, and sends 158 bits each time it is read.

**Keywords:** RFID, privacy, pseudonym protocol, cryptography.

## 1 Introduction

Radio Frequency Identification (RFID) technology holds great promise, but it also raises significant privacy concerns. The term RFID represents a family of emerging technologies that enable object identification without physical or visual contact. The main idea is to give a unique identity to every object by attaching a tag. A tag is a small chip, with an antenna, that stores a unique ID and other

* dmolnar@eecs.berkeley.edu
** andrea.2.soppera@bt.com
*** daw@eecs.berkeley.edu

information which can be sent to a reading device. The reading device uses a database to link the tag ID with information about the object it is attached to.

Today's RFID systems do not authenticate the tag, so it is easy for an attacker to impersonate other tags. This allows an attacker to mislabel goods for illicit gain, e.g. causing an expensive item to be reported as a cheap one at check-out time. Future systems will need to provide a way for readers to authenticate tags and prevent such impersonation attacks.

The other main concern in RFID systems is the privacy of the user. Today, tags can be read remotely and invisibly by any reader. This leads to unwanted consequences, such as the surreptitious tracking of objects and people through time and space. For instance, any party could use the RFID tags to track people's movements without authorization, since the ability to recognize an RFID tag allows for tracking items, and by extension, the people associated with them. A future protocol should prevent unauthorized readers from violating the privacy of users.

Some of the early work in this area has proposed protocols for mutual authentication between the tag and the reader [9, 6]. Mutual authentication protects privacy, because the tag can insist that the reader authenticate itself and prove it is authorized before releasing the tag identity. However, mutual authentication is overkill for many RFID applications, because in most cases we simply want to know the tag's identity, and mutual authentication incurs an unnecessarily high performance overhead. Moreover, these mutual authentication schemes cannot be used with existing standard reader protocols. It would be better to have solutions that are compatible with legacy readers, which can only read and pass along a tag identifier.

We propose a cryptographic scheme that protects privacy while retaining many of the legitimate benefits of current RFID technology. The main idea is to introduce an RFID *pseudonym scheme* and to use a *Trusted Center* (TC) to enforce the desired privacy policy and limit which readers may read each tag. Each time the tag is read, it generates a new pseudonym and sends this pseudonym to the reader. The Trusted Center is able to decode this pseudonym and obtain the tag's identity. Online readers can simply contact the Trusted Center and request that the pseudonym be decoded (if allowed by the privacy policy). In addition, we provide mechanisms so that the decoding can be performed anywhere in the network, enabling us to support legacy readers and offline operations.

Our scheme provides two new features not seen in prior RFID protocols, namely *time-limited delegation* and *ownership transfer*. Delegation enables a reader to decode a particular tag's pseudonyms without any further assistance from the Trusted Center, by transferring the secrets associated with that tag to the reader. Time-limited delegation allows to provide a controlled form of delegation, where the reader receives only the ability to recognize the next $q$ pseudonyms for this tag (where $q$ can be chosen arbitrarily). We can use time-limited delegation to reduce the exposure if an adversary breaks into the reader: instead of losing the secrets for all tags for all time, we lose only what was delegated to that particular reader. Delegation also gives us a way to tolerate poor quality network connections between the reader and Trusted Center, since the reader does not need network access once it has received its delegated secrets. Finally, we show how to use delegation to help Alice and Bob, who both trust the same Trusted Center but do not trust each other, securely transfer an RFID-tagged item from one to the other. After the transfer, Bob has assurance that Alice can no longer read the RFID tag on the item, even though she could before. Our methods for ownership transfer require minimal or no online interaction by the Trusted Center itself.

We present two versions of our scheme. The first version stores a counter on the tag and provides all of the features discussed so far. For tags that do not support any form of writable non-volatile state, we also design a second version that requires only a random number generator and read-only state. However, this second version does not support time-limited delegation or ownership transfer.

Our scheme seems to be practical. It can leverage the existing infrastructure of readers. The tag need only provide support for symmetric-key cryptography and either a counter or a random number generator. These requirements appear to be reasonable for a large class of RFID applications, including many deployments that have already raised significant privacy concerns.

## 2 Towards a Secure RFID Tag Protocol

We begin by outlining the features our protocol is designed to provide and some of the key challenges in achieving these goals.

*Pseudonyms.* Our main goal is to allow authorized readers to identify and authenticate the RFID tag, while preventing unauthorized readers from determining anything about the identity of tags they interact with. One possible approach would be to require readers to authenticate themselves to the tag before they are allowed to read its contents; however, this would require changing the communication protocol between tags and readers, and thus would mean that existing readers would have to be replaced. Therefore, our approach is to build a RFID pseudonym protocol [8]. In our scheme, the RFID tag replies with a unique pseudonym that changes each time it is queried. The pseudonym is generated based on some secret key that is stored on the tag and known to authorized readers, so that authorized readers can identify the tag. However, without that secret, the pseudonym provides no information about the tag's identity. In particular, pseudonyms are unlinkable, so that unauthorized readers will be unable to tell if two pseudonyms came from the same tag. In this way, possession of the secret key controls the ability to link sightings of the same tag.

The tag-reader protocol is very simple: the reader interrogates the tag, and the tag responds with its current pseudonym. Our use of pseudonyms allows the scheme to be compatible with legacy readers, because the reader does not need to know anything about the way that pseudonyms are generated or decoded. Instead, the reader can forward the pseudonym it received to some other entity, and that other entity can recover the tag's identity from the pseudonym.

*Privacy.* It is important to be able to specify a privacy policy for each tag, restricting which readers are authorized to read that tag. Our architecture includes a central trusted entity, which we call the Trusted Center (TC), whose role is to manage and enforce these privacy policies. When a tag is enrolled into the system, it is loaded with a secret key generated for it by the TC. The TC keeps a database listing, for each tag, the secret key provided to that tag, the information associated with that tag (such as its identity), and that tag's privacy policy. Given any pseudonym from an enrolled tag, the TC can decode the pseudonym and determine the identity of the tag using the secret keys stored in its database.

Note that we do not require the existence of a single global Trusted Center that is trusted by everyone in the world. Although it would be possible to set up a worldwide key infrastructure with a single globally trusted root (e.g., administered by a consortium of tag manufacturers), this is not necessary. For example, a library deploying RFID could act as its own Trusted Center, enrolling a tag and writing secrets to it when the tag is applied to a library book. If libraries do not need to read each other's tags, then no library need trust any other.

In our system, the Trusted Center acts as a trusted third party that manages the privacy policy associated to tags. We envision that the Trusted Center might provide a way for the owner of each tag to specify a privacy policy for that tag, listing which readers are authorized to decode this tag's pseudonyms. Manufacturers might also specify a default policy when the tag is created, allowing us to support both opt-in and opt-out policies. When the Trusted Center receives a request from some reader to decode a particular pseudonym, the Trusted Center can decode the pseudonym, consult the tag's privacy policy, and decide whether to reveal the tag's identity to this reader.

This provides a simple way for users to delegate access only to specific readers. In the future, a RFID infrastructure might consist of thousands or even millions of RFID readers deployed across the planet, and we need a way for legitimate readers to read the tag. In a naive implementation, the Trusted Center might give a copy of the tag's secret key to each reader that is authorized to read the tag. However, this form of delegation is too coarse-grained, because the reader then permanently receives the ability to identify this tag for all time. We may not wish to place this much trust in every RFID reader that ever encounters the tag, because then compromise of any one reader could endanger the privacy of many users. The challenge is to provide time-limited delegation, where a reader's ability to read a tag can be limited to a particular time period.

*Time-limited Delegation.* Controlling delegation is easy if all readers are online—the reader can simply act as a dumb relay, passing on the pseudonym from the tag to Trusted Center and letting the TC reply with the identity of the tag (if permitted by this tag's privacy policy). However, this approach requires a costly interaction between the reader and TC every time a tag is read. Because today's readers may repeatedly broadcast queries to all tags within range at a rate of 50 times per second or so, the burden on the TC and the database may be very high: if there are 10 tags within range, we require 500 round-trip interactions per

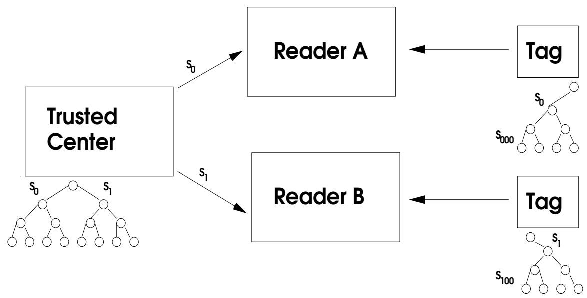


**Fig. 1.** The Trusted Center delegates access to two different readers.

second with the TC, multiplied times the number of readers. We instead focus on the problem of offline delegation.

In our scheme, the TC can compute a time-limited secret that provides only the ability to disambiguate pseudonyms for a particular tag for a limited number of times. In particular, the TC computes a secret that allow to recognize the next $q$ pseudonyms from this tag, where $q$ is arbitrary and can be specified by the privacy policy. This secret can be communicated to the reader through any secure channel, and for the next $q$ tag-reads the reader does not need to interact with the TC in any way. After reading the tag $q$ times, the reader loses the ability to link tag readings and must contact the Trusted Center to ask for re-authorization. (See Figure 1.)

Delegation is helpful for cases where readers have intermittent or low-bandwidth connectivity. When a reader first sees a tag it is unable to recognize, the reader can send the pseudonym it received to the TC. If this reader is authorized for this tag, the TC can return not only the tag's identity but also a secret that allows reading the tag for a limited time—say, for 1000 queries—without requiring further interaction with the TC. Delegation provides benefits even for online readers, because the locality in tag sightings can be used to greatly improve performance and reduce communication with the TC.

Our scheme also supports recursive delegation: after we delegate to Alice limited-access to the tag, she can further re-delegate to Bob the power to query this tag, and Bob can further delegate to Carol, and so on. Moreover, the rights delegated can be limited arbitrarily at each step. For instance, if Alice receives a secret that lets her identify the tag for the next 100 queries, she can compute a secret for Bob that will let him read the tag for the next 40 queries, a secret for Bill that lets Bill read the tag for the 30 queries after that, and so on. To the best of our knowledge, no previous protocol for RFID privacy has addressed delegation, let alone provided support for recursive delegation.

*Ownership Transfer.* A related problem to delegation is that of *ownership transfer*, where Alice gives an RFID-tagged item to Bob. After the transfer of ownership, Bob should be able to read the item but Alice should not. Pseudonyms allow us to cleanly deal with ownership transfer from Alice to Bob. If Alice has

| Scheme | $T_{Reader}$ | $S_{Reader}$ | $T_{TC}$ | $S_{TC}$ | # Msg | Comm | Delegation? |
|---|---|---|---|---|---|---|---|
| OSK [8] | O($N$) | O($N$) | NA | NA | 1 | O(1) | No |
| AO [1] | O($N^{2/3}$) | O($N^{2/3}$) | NA | NA | 1 | O(1) | No |
| MW [6] | O($\log N$) | O(1) | NA | NA | O($\log N$) | O($\log N$) | No |
| **Basic** | O($D$) | O($D$) | O($\log N$) | O($2^{d_1}$) | 1 | O($\log N$) | Yes |
| **Optimized** | O($D$) | O($D$) | O($\log N$) | O(1) | 1 | O($\log N$) | Yes |

**Fig. 2.** Comparison to previous RFID privacy schemes. Here $T_{TC}$ and $S_{TC}$ stand for the time and storage requirements of the Trusted Center, with the Reader requirements marked similarly. $N$ is the total number of tags in the system, $d_1$ is the depth of the Trusted Center's tree, and $D$ is the number of tags delegated to a particular reader. In practice, we expect $D \ll N$. The Optimized Scheme uses a PRF to generate the TC's tree of secrets and truncates the tag outputs, as described in Section 6.

not been delegated the ability to disambiguate pseudonyms, no further work is needed: once Bob registers his ownership of this tag, the TC can simply deny any future requests from Alice to read this tag. If Alice has been delegated secrets that let her read this tag, we have two methods for ensuring Alice can no longer link a tag after it is passed to Bob. Both are described in more detail in Section 5.

*Scalable Lookup.* A major technical challenge in the design of such systems is how to make them scale to a large number of tags. Consider a TC with a database of $N$ tags. Naively, decoding a pseudonym might require a linear scan through all $N$ tag keys, which may not be practical for an RFID system with $N = 10^6$ tags. Instead, we design a scheme with logarithmic complexity: the TC does just O($\log N$) work to disambiguate a pseudonym. In Figure 2, we compare our system to previous RFID pseudonym schemes.

Delegation incurs some performance overhead at the readers. In our scheme, a reader that has received $D$ delegations will require O($D$) work per tag queried. In practice we expect $D$ will be small compared to the total number of tags; for example, $D$ might be the number of tags in a single shipment of goods. Therefore, we expect this performance level to be adequate in practice.

## 3 Notation

We use a pseudo-random function (PRF) $F : K \times \{0,1\}^n \to \{0,1\}^n$ and a pseudo-random generator (PRG) $G : K \to K \times K$. Also, we use $G_0(k)$ and $G_1(k)$ to denote the first and second part of $G(k)$, respectively, so that $G(k) = (G_0(k), G_1(k))$.

In practice we might use AES as the PRF. Recent results on low-gate-count implementations of AES suggest that AES may be within reach for all but the lowest-end RFID tags [2]. We might also define the PRG in terms of the PRF, for instance defining $G$ by $G_b(k) = F_k(0^{n-1}\,b)$, so that the tag needs only a single cryptographic primitive. One should be careful to ensure that the inputs

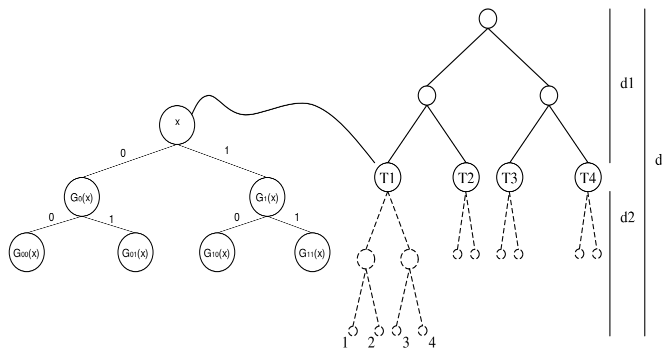


**Fig. 3.** An example tree of secrets for four tags in our RFID pseudonym scheme. The nodes drawn with solid lines correspond to secrets shared only between the tags T1,...,T4 and the Trusted Center. Each of these secrets is drawn uniformly at random and independently of each other. The dashed line nodes are secrets in *delegation trees*, where keys at child nodes are derived by the GGM construction from the key at their parent. On each read, a tag updates its state to use the next leaf in the delegation tree for its next pseudonym. To delegate limited-time access to a tag, the Trusted Center can give out subtrees of the delegation tree; for example, the immediate parent of 1 and 2 allows learning T1's identity in time periods 1 and 2, but not in time periods 3 and 4.

to the PRF when used for PRG-emulation are disjoint from the inputs to the PRF elsewhere in the protocol, for instance by having the first bit of the PRF input indicate which mode it is being used in.

If $s \in \{0,1\}^*$ is a bitstring, we use $s_{1..i}$ to denote the first $i$ bits of $s$, and $\mathrm{len}(s)$ to denote the length of $s$ (in bits). Also, we place the nodes of a complete depth-$d$ binary tree in one-to-one correspondence with $\{0,1\}^{\leq d}$, the set of bitstrings of length at most $d$. The empty string represents the root of the tree. If $s$ is any internal node, $s\,0$ and $s\,1$ are used to represent its left and right children, respectively. Thus each bitstring of length $\leq d$ identifies a node in the binary tree by specifying the path from the root that reaches it. We sometimes also use $s$ to refer to the integer $s_1 2^{n-1} + \cdots + s_{n-1} 2 + s_n$ obtained by viewing the string $s$ as a number written in big-endian binary notation.

If $f : S' \to T$ is a function and $S \subseteq S'$, let $f|_S : S \to T$ denote the restriction of $f$ to $S$. When given a function $H : \{0,1\}^{\leq d_1} \to K$ defined on $\{0,1\}^{\leq d_1}$, we will extend it to a function defined on all of $\{0,1\}^*$ using the recurrence $H(s\,b) = G_b(H(s))$.

**Tag State:** (initialized by TC.EnrollTag)
$c$, a counter in $\{0,1\}^d$.
$S$, a set with $S \subseteq \{0,1\}^{\leq d_1}$.
$h$, where $h : S \to K$.

Algorithm Tag.Respond():
1. Pick $r \in_R \{0,1\}^k$ uniformly at random.
2. Set $p := (F_{h(c_{1..1})}(r), F_{h(c_{1..2})}(r), \ldots, F_{h(c_{1..d})}(r))$.
3. Set $c := c + 1$.
4. Return $(r, p)$.

**Fig. 4.** Algorithms and state for the RFID tag.

## 4 Our Protocol

*Tree of Secrets.* Our protocol is based around a tree of secrets of depth $d = d_1 + d_2$ as shown in Figure 3. Each node in the tree has its own $k$-bit secret key. For simplicity, we describe our scheme in terms of a complete binary tree $\{0,1\}^{\leq d}$, though we will later generalize this to larger branching factors.

The first $d_1$ levels of the tree contain node secrets that are chosen uniformly and independently at random by the Trusted Center during system initialization (see algorithm TC.GenTC in Figure 5). Each node at depth $d_1$ corresponds to a unique tag. When the tag is enrolled into the system, it receives all keys on the path from its node to the root. Therefore, each tag only needs capacity to store $d_1$ secrets.

The next $d_2$ levels of the tree contain secrets that are derived using a GGM tree construction [3]: each node is labelled with a secret, and the secrets for its children are derived by applying a PRG. Knowing a secret at level $\geq d_1$ allows computation of the secrets for every descendent in the subtree rooted at that node, but nothing else.

Formally, the TC chooses a function $H : \{0,1\}^{\leq d_1} \to K$ uniformly at random, and $H(s)$ denotes the key associated with node $s$ in the tree. We extend the function $H : \{0,1\}^{\leq d_1} \to K$ to a function $H : \{0,1\}^{\leq d} \to K$ by the rule $H(s\,b) = G_b(H(s))$ for all $s \in \{0,1\}^{\geq d_1}$, $b \in \{0,1\}$. For the rest of this paper, we assume this extension is implicitly performed whereever necessary, and we do not distinguish between $H$ and its extended version.

Each tag receives $H|_S$ for some prefix-closed set $S = \{t_{1..1}, \ldots, t_{1..d_1}\}$ corresponding to the path to the root. This means that the tag effectively learns the function $H|_{S'}$, where $S' = \{t_{1..1}, \ldots, t_{1..d}\}$, though it only needs to store the first $d_1$ secrets in this list.

*Tag Responses.* Each tag $T$ keeps a counter $T.c$. The counter identifies a leaf at level $d$ of the tree; thus, each counter value corresponds to a new pseudonym for this tag. The tag responds to a query from a reader by generating a random number $r$ and sending a pseudonym

$$(r, p) = (r, (F_{h(c_{1..1})}(r), F_{h(c_{1..2})}(r), ..., F_{h(c_{1..d})}(r)))$$

where the $h(c_{1..i})$ values represent secrets along the path in the tree of secrets from the root to the tag's current leaf $T.c$. The tag then increments the counter $c$. See Figure 4.

Notice that because the counter $c$ is incremented on each query, the tag will use a different path of secrets, and therefore a different pseudonym, for every query. This is what enables delegation, because we can give the reader a subtree of secrets that will expire after a certain number of tag reads. The tag's workload is quite modest: only $d + d_2$ invocations of the PRF are needed per query. By varying the branching factor and depth of the tree, we can trade off between the complexity of Tag.Respond and the complexity for the reader. See Section 6.

*Decoding Pseudonyms.* Given a pseudonym $(r, p)$, it is possible to use the tree structure to efficiently decode this pseudonym and discover the identity of the tag that generated this pseudonym. The main idea is to use a depth-first search to find a path in the tree that matches the response $p$. We start at the root of the tree of secrets. At each node $s$, we can check whether the left child $s\,0$ or the right child $s\,1$ matches entry $p_i$ in the response by checking whether $F_{s0}(r) = p_i$ or $F_{s1}(r) = p_i$, respectively. In this way, wrong paths can be quickly pruned. See TC.IdentifyTag in Figure 5.

Given a pseudonym, this procedure lets the TC identify the tag's real identity $ID$. Based on the identity of the tag, the identity of the reader, and the privacy policy for this tag, the TC can then decide whether to reveal $ID$ to the reader. This provides a mechanism for enforcing a privacy policy regarding which readers are allowed to learn which tag IDs.

*Delegation.* Our protocol also allows the TC to delegate access to a certain interval of pseudonyms to an offline reader. This can be thought of as allowing the reader to perform the mapping itself from a pseudonym $(r, p)$ to the tag's identity $ID$, but only if the tag's counter value is in a prescribed interval $[L, R]$ (for some $1 \le L \le R \le 2^d$).

Recall that each leaf of the tree corresponds to a different pseudonym for a tag. To delegate access to leaves in an interval $[L, R]$, the Trusted Center first determines the smallest set $S \subseteq \{0,1\}^{\ge d_1}$ of tree nodes that cover the interval $[L, R]$. We say that $S$ covers $[L, R]$ if for all $x \in [L, R]$, there exists $s \in S$ so that $s$ is a prefix of $x$. The Trusted Center then sends $H|_S$ to the reader along with the tag's identity. Now, when the reader sees the pseudonym $(r, p)$, the reader no longer needs to communicate with the Trusted Center. Instead, the reader can perform a depth-first search starting at each node in $S$, since $H|_S$ contains everything the reader needs to know to perform this search. See Figures 5 and 6.

After the tag updates itself past the leaf $R$, however, the reader can no longer recognize any subsequent pseudonyms from this tag. This is because the counter

**TC State**:
$H : \{0,1\}^{\leq d_1} \to K$, a function.

Algorithm TC.GenTC():
1. Let $H : \{0,1\}^{\leq d_1} \to K$ be a random function, i.e., pick $H(s) \in_R K$ uniformly at random for each bitstring $s$ of length at most $d_1$.

Algorithm TC.EnrollTag($ID$):
1. Find the smallest integer $t \in \{0,1\}^{d_1}$ that hasn't been assigned to any other tag. Assign $t$ to this tag.
2. Set $S := \{t_{1..j} : 1 \leq j \leq d_1\}$.
3. Return $(t\,0^{d_2}, S, H|_S)$ as the state for this tag.

Algorithm TC.Delegate($L, R$):
1. Let $S$ denote the minimal subset of $\{0,1\}^{\geq d_1}$ such that for all $x$ with $L \leq x \leq R$, there exists $s \in S$ so that $s$ is a prefix of $x$.
2. Return $H|_S$.

Algorithm TC.IdentifyTag($r, p$):
1. Return $\mathrm{DFS}(r, p, 1, \epsilon)$, where $\epsilon$ denotes the empty bitstring.

Algorithm $\mathrm{DFS}(r, p = (p_1, .., p_d), i, s)$:
1. If $i = d + 1$, return $\{s_{1..d_1}\}$.
2. Set $ids := \emptyset$.
3. If $F_{H(s\,0)}(r) = p_i$ then set $ids := ids \cup \mathrm{DFS}(r, p, i+1, s\,0)$.
4. If $F_{H(s\,1)}(r) = p_i$ then set $ids := ids \cup \mathrm{DFS}(r, p, i+1, s\,1)$.
5. Return $ids$.

**Fig. 5.** Algorithms and state for the Trusted Center.

Tag.$c$ will have updated past the subtree of secrets known to the reader. The reader's access to the tag has effectively expired, and at this point the reader must re-apply to the TC if it wants continued access.

Note that decoding a pseudonym takes the reader $O(D)$ invocations of the PRF (for $D = |S|$), since the reader must check every value in its delegated subset $S$ for a match with the tag's response.

*Second Version: Eliminating the Counter.* In low- and middle-end RFID technologies, writing permanent state such as a counter on each tag read may be difficult, making our first protocol inapplicable. For example, the EPC Gen II specification requires a random number generator, but EPC tags are read at a distance of several meters and may not have enough power available for writes.

We now design a second version of the protocol that eliminates the need for updateable non-volatile state, assuming the tag can generate random numbers on demand. We replace the counter Tag.$c$ with a $d$-bit value whose first $d_1$ bits are fixed at the unique value $t$ (as before) and whose last $d_2$ bits are chosen uniformly at random for each query. Thus, the tag uses the same subtree to

**Reader State:** (updated by TC.Delegate)
$h : S \to K$, for some $S \subseteq \{0,1\}^{\geq d_1}$, with $S$ initialized to $\emptyset$.

Algorithm Reader.IdentifyTag$(r, p)$:
1. Set $ids := \emptyset$.
2. For each $s \in S$ such that no prefix of $s$ is in $S$, do:
3. Set $ids := ids \cup \mathrm{DFS}(r, p, \mathrm{len}(s) + 1, s)$.
4. Return $ids$.

**Fig. 6.** Algorithms and state for the reader.

generate pseudonyms, but instead of walking along the leaves from left to right one at a time, it instead picks a random leaf each time it is read. The Trusted Center's algorithms remain unchanged in either case. Unfortunately, the second version of our protocol does not support time-limited delegation or ownership transfer.

*Security and Privacy.* Our protocol provides replay-only security against impersonation attack and privacy against a radio-only adversary. Informally, this is because each pseudonym emitted by a tag is indistinguishable from other pseudonyms unless the secret keys are known; we give formal definitions and proofs in the full version of the paper [5].

Our protocol provides replay-only security against impersonation attack even if an adversary can compromise tags. This is because each tag has at least one secret not shared with any other tag; to perform a successful non-replayed impersonation, the adversary would need to predict the value of a PRF keyed with such a secret.

Privacy, on the other hand, degrades under tag compromise. This is because tags may share secrets in the tree of secrets. The amount of degradation depends on the branching factor of the tree. At one extreme, a single-level tree with a branching factor of $N$ loses no privacy under tag compromise. At the other extreme, two randomly chosen tags in a binary tree have a chance of $1/2^k$ of sharing $k$ secrets. Each deployment can pick the branching factor that makes the best tradeoff between privacy loss under tag compromise and reader complexity. Even at high branching factors, however, our scheme provides benefits via delegation.

## 5 Ownership Transfer

Ownership transfer in RFID is the following problem: Alice gives an RFID tag to Bob. How do we prevent Alice from later reading the RFID tag? This problem is crucial for limiting the trust required in readers which may need to read tags at some point in the tag's lifetime.

In the case that Alice has not been delegated access to the RFID tag, ownership transfer in our model is simple. The Trusted Center is notified of the transfer and updates a privacy policy associated with the tag. Afterwards, Alice requests access to the tag's ID. The Trusted Center then checks the privacy policy, sees Alice no longer owns the item, and denies access. In case Alice has been already been delegated access to the tag, we introduce two methods for ownership transfer.

*Soft Killing.* In the first method, *soft killing*, Bob queries the Trusted Center and learns how many leaves were delegated to Alice. Suppose this number is $k$. Bob then reads the tag $k+1$ times. The tag will then have updated past Alice's access, so she will no longer be able to disambiguate the tag's pseudonyms. Notice that even if Bob knows how many leaves were delegated to Alice, he still cannot distinguish a tag delegated to Alice from any other tag without Alice's help; this is because the tag will emit a new, pseudorandom, pseudonym on each read. Therefore knowing the number of leaves delegated to Alice does not hurt the privacy of our protocol.

The benefit of soft killing is that it does not require shared secrets between the tag and reader. The downside is that soft killing requires many tag reads. Soft killing also opens up the possibility for a denial of service attack if an adversary reads the tag many times; Alice can recover from this by simply asking the Trusted Center to delegate more access.

*Increasing The Tag Counter.* In the second method, we allow Bob to increase the counter on a tag from $c$ to $c'$. Bob does so by sending the tag a random seed $r$, after which Bob and the tag can perform mutual authentication and establish a secure channel with the shared secret $F_{h(c)}(r)$. Bob then sends $c'$, plus a proof that Bob knows the secret for the leaf $c'$, to the tag over the secure channel. The tag checks that $c' > c$, so Bob can only increase the tag's counter, not decrease it. By doing so, Bob can "leapfrog" the tag over Alice's delegated leaves and be sure that Alice can no longer read the tag. Increasing the counter requires only one read, but also requires the tag to implement a substantially more complex protocol.

## 6 Optimizations and Weakening Assumptions

*Reducing TC Storage.* In our protocol as described, the Trusted Center must generate and store $2^{d_1+1}$ independent random values. We can reduce this storage to a single key by instead having the Trusted Center use a PRF with a master key $mk$ that is never revealed to any other party. The PRF evaluated at a nodeID yields the secret for the node: $H(s) = F_{mk}(s)$ for $s \in \{0,1\}^{\leq d_1}$.

*Random Number Generation.* In some RFID technologies, it may be difficult to generate random numbers. If the tag can support writable non-volatile state, we can replace the random number generator with a PRF run in counter mode. See Figure 7. We stress that the key $rk$ used for random-number generation is not shared with any reader at any time.

PRNG.Initialize()
1. Initialize `ctr` to 0.
2. Pick secret key $rk \in_R K$.

PRNG.GetNextNonce()
1. Return $F_{rk}(\texttt{ctr++})$.

**Fig. 7.** Generating nonces with a PRF and a counter.

| Number of Tags | Tag Storage | Communication | Tag Computation | Reader Computation |
|---|---|---|---|---|
| $2^{20}$ | 128 bits | 158 bits | 6 | $6 \cdot 2^{10}$ |
| $2^{30}$ | 192 bits | 168 bits | 7 | $7 \cdot 2^{10}$ |
| $2^{40}$ | 256 bits | 178 bits | 8 | $8 \cdot 2^{10}$ |

**Fig. 8.** Concrete resource use of our scheme for some example parameters. We use a branching factor of $2^{10}$ in all cases, use a 64-bit $r$ value with truncation, and we assume tags will be read at most $2^{20}$ times. Tag and reader computation are both measured in expected number of PRF evaluations.

*Truncating PRF Values.* Instead of sending full PRF values in a tag response, it is more efficient to send truncated versions. This reduces communication overhead at the cost of following false paths during the depth-first search. To avoid misidentification of tags, we recommend truncating only at the internal nodes and sending the full-length PRF output at the leaves. If internal nodes are truncated to $a$ bits, the tag's response becomes $(r, p)$ where $p := (F_{h(c_{1..1})}(r) \bmod 2^a, ..., F_{h(c_{1..d-1})}(r) \bmod 2^a, F_{h(c_{1..d})}(r))$. With full-length values at the leaves, the probability of misidentification is negligible.

When PRF responses are truncated, identifying a tag requires searching through the tree, and this search might follow false paths that do not correspond to the true tag identity. If the branching factor is exactly $2^a$, it is possible to show that the search process is a birth-death process and that the expected complexity of the search is $\mathrm{O}(2^a \times \lg N) = \mathrm{O}(2^a \times d)$.

*Branching Factor and Concrete Examples.* Truncation greatly reduces communication overhead while only slightly impacting the complexity of tag identification. For instance, with a binary tree of depth $d = 40$, we might truncate PRF values to 1 bit at internal nodes and use a 64-bit PRF output at the leaves. With these parameters, the response $p$ will be 103 bits long, while the search complexity remains minimal.

In practice, we would use trees with branching factors much larger than 2. A larger branching factor reduces the depth of the tree, thus reducing tag storage and computation, at the cost of more computation for the Trusted Center and reader. For example, consider an RFID system with $N = 2^{20}$ tags, each of which will be read at most $2^{20}$ times. We construct a four-layer tree of secrets with branching factor $1024 = 2^{10}$ at all levels. Each tag stores two 64-bit secrets $s_1, s_2$, with the second secret being the root of a GGM tree that covers the final two tree levels. Each pseudonym requires two PRF invocations to compute $s_3, s_4$ and four PRF invocations to compute the response. Total tag storage is $2 \cdot 64 = 128$ bits and total tag computation is 6 applications of the PRF. If we truncate the tag's responses to 10 bits at internal nodes and 64 bits at the leaf, and use a 64-bit $r$, the tag's total communication is $64 + 30 + 64 = 158$ bits. The work for the reader, on the other hand, is only $6 \cdot 2^{10}$ applications of the PRF. We show concrete parameters for this and some other examples in Figure 8.

## 7 Related Work

Weis et al. provide "hash lock" protocols for private mutual authentication [9]. As we have discussed, mutual authentication is not needed in scenarios when only tag identification is required, and it incurs significant performance costs. Their schemes also require readers to perform work linear in the number of total tags and do not support time-limited delegation to offline readers. Because they choose independent secrets for each tag, however, they do not suffer from privacy loss under tag compromise.

Molnar et al. show how to use a tree of secrets to achieve mutual authentication protocol with complexity logarithmic in the number of tags [6]. Their scheme requires at least 3 rounds of communication between tag and reader, while we use only one message from tag to reader. More importantly, their work does not support delegation, nor does it work with legacy readers. Our work uses a similar tree construction to achieve logarithmic work, but applies the idea to RFID pseudonyms. Our recursive tree-walking scheme has some similarities with the traitor tracing scheme of Naor et al. [7].

Ohkubo et al. introduce a scheme for RFID pseudonyms [8]. In their protocol, recovering the tag identity requires work linear in the number of possible tags, while we achieve logarithmic work. They propose storing the expected next output of each RFID tag as an optimization, but this cannot be kept up to date unless without online reader-TC interaction on every tag read. Avoine and Oechslin propose a time-space tradeoff technique that improves the complexity of the Ohkubo et al. protocol to $O(N^{2/3})$ time with a table of size $O(N^{2/3})$, but their protocol does not support delegation as ours does [1]. Both protocols could be extended to support a form of delegation by giving out the individual secrets for each time period, but this requires much more state on the reader. On the other hand, both schemes avoid the problem of privacy loss under tag compromise, because all tags have independently chosen secrets.

The time-memory tradeoff of Avoine and Oeschslin also requires picking a hash chain length in advance. Once a tag exceeds the number of readings set by this length, the entire table must be recomputed before that tag can be read. This is a problem because readers may read a tag 50 times per second. Further, there are active attacks that violate our privacy goals. An adversary that has the ability to trigger a legitimate reader to query a tag and learn whether the reader accepts or rejects can determine if the tag has been read more times than provided for by the hash chain length; this would enable the adversary to distinguish tags. While these concerns are alleviated by longer chain lengths, these lengths also increase the amount of space and time required for the time-memory tradeoff.

Juels gives a scheme for one-use RFID pseudonyms [4]. Unlike our protocol, Juels's scheme does not require a PRF; a simple XOR is enough. Juels also discusses methods for rotating and changing pseudonyms to support ownership transfer. His protocol, however, only allows a tag to emit a limited number of pseudonyms before it must be refreshed through interaction with a trusted reader. Juels outlines an extension to the protocol which removes this restriction using a PRG, but this method requires tight synchronization between reader and tag. Further, his protocol does not work with legacy readers, and it does not support delegation as ours does. Again, in Juels's system, compromising one tag does not aid the adversary in identifying another.

## 8 Conclusions

We have described a new cryptographic protocol for RFID privacy. In our scheme, tags generate pseudonyms that can only be decoded with knowledge of the appropriate secrets, and privacy is protected by controlling which parties are given access to these secrets. The key ingredient of our protocol is a set of secrets organized in a tree format. This tree structure enables many powerful features, including support for legacy readers, disconnected operation, flexible privacy policies, delegation to authorized readers, time-limited delegation, recursive delegation, and ownership transfer between users. At the same time, our scheme is practical and scalable: it requires only a few PRF invocations and a modest amount of communication between the tag and reader, even for very large deployments. We believe our protocol could enhance the privacy protection for a wide range of current and future deployments of RFID technology.

## 9 Acknowledgements

We thank Gildas Avoine, Michael Backes, Trevor Burbridge, Etienne Dysli, Arnaud Jacquet, Pascal Junod, Vivekanand Korgaonkar, Philippe Oechslin, and the anonymous reviewers for helpful comments. The authors also gratefully acknowledge support from NSF grant CCR-0093337, the Sloan Research Fellowship, and British Telecom. David Molnar was supported by an Intel OCR Fellowship and an NSF Graduate Fellowship.

## References

1. Gildas Avoine and Philippe Oechslin. A scalable and provably secure hash-based RFID protocol. In *IEEE PerSec*, 2005.
2. Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In *CHES*, 2004.
3. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
4. Ari Juels. Minimalist cryptography for RFID tags, 2003. `http://www.rsasecurity.com/rsalabs/staff/bios/ajuels/publications/minimalist/index.html`.

5. David Molnar, Andrea Soppera, and David Wagner. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags (full version). To appear in Cryptology ePrint Archive, 2005. `eprint.iacr.org/2005/`.
6. David Molnar and David Wagner. Security and privacy in library RFID: Issues, practices, and architectures. In *ACM CCS*, 2004.
7. D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO*, 2001.
8. Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic approach to a privacy friendly tag. In *RFID Privacy Workshop, MIT*, 2003.
9. Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *Security in Pervasive Computing*, 2004.