Defining Strong Privacy for RFID

Ari Juels[1] and Stephen A. Weis[2]

[1] RSA Laboratories, Bedford, MA, USA
ajuels@rsasecurity.com
[2] Massachusetts Institute of Technology, Cambridge, MA, USA
sweis@mit.edu

7 April 2006

Abstract. In this work, we consider privacy in Radio Frequency IDentification (RFID) systems. Our contribution is threefold: (1) We propose a simple, formal definition of strong privacy useful for basic analysis of RFID systems, as well as a different (weaker) definition applicable to multi-verifier systems; (2) We apply our definition to reveal vulnerabilities in several proposed privacy-enhancing RFID protocols; and (3) We formally analyze and suggest improvements to "Hash-Locks," one of the first privacy-enhancing RFID protocols in the literature.

Keywords: RFID, privacy, security, EPC, proximity cards

1 Introduction

Radio Frequency IDentification (RFID) systems offer improved efficiency in inventory control, logistics, and supply chain management. As such, they are of great interest to enterprises intensively reliant on supply chains, particularly large retailers and consumer product manufacturers. The long-term goal of these organizations is to integrate RFID on the retail level. Without proper protection, widespread adoption of retail RFID could raise privacy concerns for everyday consumers.

Briefly, RFID systems consist of two main components: *tags* and *readers*. Tags are radio transponders attached to physical objects. Radio transceivers, or readers, query these tags for some (potentially unique) identifying information about the objects to which tags are attached. Although readers are often regarded as a simple conduit to a back-end database, for simplicity we treat a reader and a back-end database as a single entity.

Historically, tags have been expensive, bulky devices used for tagging objects like shipping containers, munitions, automobile parts, or even live cattle. The high value of such objects justifies relatively expensive RFID devices that may perform complex computations, store ample data, and initiate their own communications. Since people do not typically carry cattle or shipping containers around with them, individual privacy is not an issue in these applications.

Concerns about privacy and security do arise, however, in new consumer applications of RFID. RFID technology is present in proximity cards or "prox cards" for physical access control, autombile keys, and in a variety of contactless payment systems. Many—if not the majority—of consumers in industrialized countries today make regular use of such systems. One pervasive future application will be "electronic product code" (EPC) tags. EPC tags will serve on consumer products as a successor to optical "UPC" bar codes.

In the case of EPC tags, private information might be read from tags attached to products like clothing, books, or prescription drugs. An insecure prox card payment system might leak information about the movements of its users. But even if the semantic meaning of information

on tags is well protected, tags may still be recognizable between appearances, and thus subject to tracking. This could violate individual "location privacy" for consumers, or could reveal strategic information about an enterprise or military supply chain.

Because of their intentionally simple design, EPC tags cannot support expensive, traditional cryptography and security functions—not even basic ciphers. Tight economic considerations suggest that this will remain the case for the foreseeable future. Next-generation EPC tags (such as the emerging, more expensive "Class 2" type) and other RFID tags, though, may be capable of performing symmetric-key cryptography. While some of the RFID literature, e.g., [10, 15] addresses privacy in tags that cannot perform cryptography, *we focus in this paper on privacy in RFID systems where symmetric-key cryptographic operations are possible.*

1.1 Privacy-preserving RFID protocols in the literature

Fortunately, much attention has been devoted to RFID security and privacy in recent years. Juels offers a survey of much of the related literature in [14] and Avoine maintains a current online bibliography at [3]. Rather than reviewing the literature here, we refer the reader to those sources. While many RFID security schemes have been proposed, there has been little formal analysis. Lacking formal security definitions, much existing work offers ad hoc notions of security.

1.2 Previous RFID-privacy definitions

Our proposed definition is similar in flavor to those of Avoine [2], who proposes very general and flexible definitions of RFID privacy. The aim of Avoine, however, is to capture a range of adversarial abilities, while we seek to characterize a very strong adversary with a relatively simple definition. Thus, we aim for specificity and simplicity over flexibility. For example, the Avoine definitions distinguish between adversarial eavesdropping on the reader-to-tag channel, i.e., reader transmissions to tags, and the tag-to-reader channel, i.e., tag transmissions to the reader. (In practice, the former is easier to eavesdrop on than the latter, as readers emit more power than tags.) In contrast, we simply assume the ability of an adversary to eavesdrop on both channels.

On the other hand, our model is more general than the Avoine models in one important respect: Ours characterizes the privacy of systems in which tags carry *correlated secrets*, as in the proposed system of Molnar, Soppera, and Wagner (MSW) [19, 18]. This feature is lacking in the Avoine models.

Consider the following scheme. Suppose that x is a secret value; let h represent a hash function (modeled as a random oracle) and E represent a symmetric-key cipher in the ideal-cipher model. At time step t, tag 1 outputs $(E_x[1,t], h(1,t,x))$; tag 2 similarly outputs $(E_x[2,t], h(2,t,x))$. Tag 3 outputs $(E_x[3,t], x \oplus h(1,t,x) \oplus h(2,t,x))$. Any pair of tags may be seen to offer strong privacy: An adversary cannot distinguish among different outputs of a given tag. But clearly an adversary capable of obtaining the outputs of the three tags at a given time t can completely break the privacy of this system by recovering x. The Avoine definitions, however, permit adversarial interaction with only two tags. They would classify this system as providing strong privacy ("Existential-UNT").

While this three-tag construction is slightly artificial, it actually reflects a feature in realistic RFID architectures. In the MSW system [18], for example, tags contain overlapping sets of secret keys derived from a tree structure.

In other work, Juels [13] presents what he calls a "minimalist" model for tag privacy and security. This model is geared at low-cost tags that cannot perform symmetric-key cryptography. The accompanying formal privacy definition may be regarded as a specialized narrowing of our

definitions here. The Juels definition specifically assumes bounds on the number of queries that an adversary can make in mounting a man-in-the-middle attack. Thus, that definition does not aim to capture RFID privacy in a strong sense. Similarly, Golle et al. [11] define privacy is a sense specific to a cryptographic primitive they propose called "universal re-encryption" of which they mention RFID as a possible application. Their definition excludes certain forms of active adversarial attack.

Organization

In section 2, we present our formal definition for strong RFID privacy. In section 3, we examine several different privacy-preserving RFID systems proposed in the literature. We show how each of these systems falls short of our strong definition of privacy; we demonstrate practical attacks that undermine important privacy characteristics. In section 4, we briefly examine a privacy-preserving RFID idea called "Hash-Locks" that meets our definitional requirements. We prove that one variant meets our privacy definition and propose a simple improvement that achieves privacy in a broader sense. We conclude in Section 5.

2 Formal Privacy Definition

In this section we give a formal definition of privacy in RFID systems that can model a variety of security protocols and attacks. We regard a system as comprising a single reader $\mathcal{R}$ and a set of n tags $\mathcal{T}_1, \ldots, \mathcal{T}_n$. Each party is a probabilistic interactive Turing machine with an independent source of randomness and unlimited internal storage. Tags and readers are modeled as "ideal functionalities," similar to entities in Canetti's Universal Composability paradigm [8]. Functionalities have a well-defined interface, may receive messages, and may respond with messages of their own.

We do not use the term "oracle" to describe tags and readers, since it sometimes implies a deterministic function where inputs always have the same output. In our model, tags and readers may maintain internal state, are randomized, and may adaptively change their output. We also want to avoid confusion when we later use "oracle" in the context of the random oracle model.

2.1 Tag Functionalities

Each tag functionality $\mathcal{T}_i$ stores an internal secret *key* and a session identifier *sid*. A tag can be assigned a new *key* via a SetKey message. A tag responds to a SetKey message by disgorging its current key. The caller may then send an arbitrary new key to replace the prior key. A tag SID can be set to a new value *sid* via the message (TagInit, *sid*). TagInit messages delete information associated with an existing *sid*. In other words, a tag may be involved in only one protocol session at a time.

A tag may respond to a protocol message or challenge, denoted c_j, with a response r_j. The tag functionality interface is illustrated in Figure 1. We explain it further below.

2.2 Reader Functionality

The reader $\mathcal{R}$ is initialized with private key material. For the purposes of this model, this key material is immutable and internal to the reader. Tag data may be thought of as residing in a back-end database containing records of the tags "owned" by a particular reader. The reader functionality initializes a new session upon receipt of a message of the form ReaderInit. When receiving a ReaderInit message, $\mathcal{R}$ generates a fresh session identifier, *sid*, and the first challenge of an interactive challenge-response protocol, c_0.

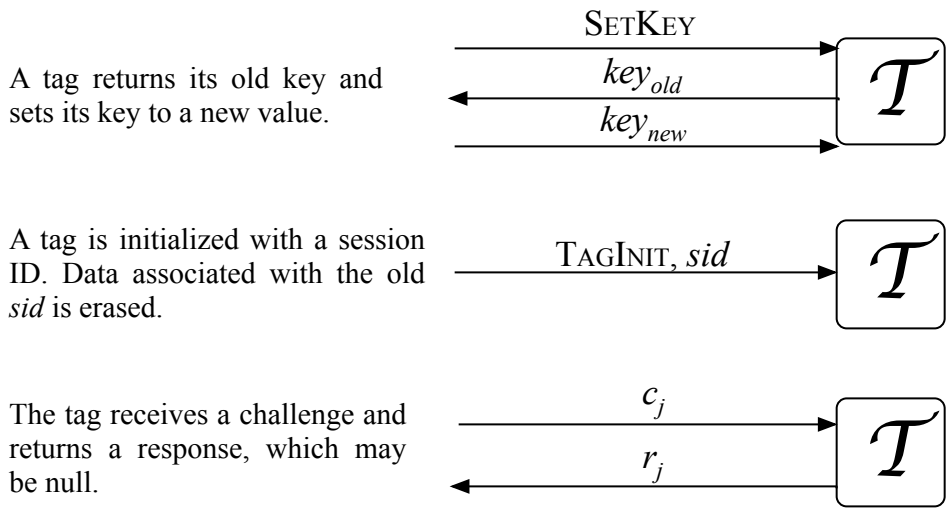

Fig. 1. Illustration of the Tag functionality interface.

For each ReaderInit received, the reader creates a new internal entry of the form $(sid, \text{“open”}, c_0)$. Any responses containing *sid* are appended to that entry, as well as subsequent challenges, or any other auxilliary data. This entry is marked as “closed” and becomes read-only when the reader ultimately accepts or rejects a session. The interface to a reader functionality $\mathcal{R}$ is illustrated in Figure 2; further explanation will follow.

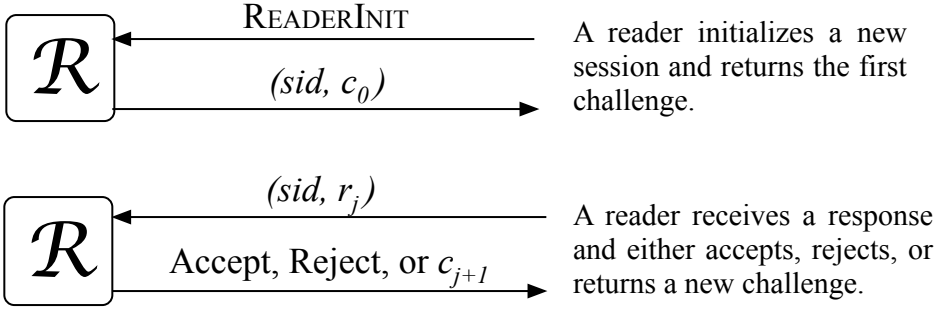

Fig. 2. Illustration of the Reader functionality interface.

2.3 Functionality Interaction

The *sid* values generated by passing ReaderInit messages to $\mathcal{R}$ may be passed as a TagInit message to some $\mathcal{T}$. Once a tag is initialized with a *sid*, it can respond to challenge c_j with response r_j. This response can be a function of the *key*, *sid*, the $j-1$ previous challenge-response pairs, and locally generated randomness. The tag may internally log both the latest received c_j and the

response r_j. This protocol log and any other data except *key* are deleted by a tag upon receipt of a TagInit message. Thus, a tag's responses can only depend on that particular session.

Similarly, reader functionality $\mathcal{R}$ responds to messages of the form (sid, r_j) by first searching for a $(sid, \text{"open"}, c_0, r_0, \ldots, c_j)$ session entry, if any exists. At this point, $\mathcal{R}$ performs a verification step by computing a function over its entire internal state, including all open and closed sessions and any internal key material. $\mathcal{R}$ can output an "accept" or "reject" and mark the session entry as $(sid, \text{"closed"}, c_0, r_0, \ldots, c_j, r_j)$. Alternatively, $\mathcal{R}$ can compute the next challenge c_{j+1} and append it to the corresponding session entry.

2.4 Parameterized Adversaries

An adversary $\mathcal{A}$ can issue its own SetKey, TagInit, ReaderInit, challenge, and response messages. We parameterize the number of ReaderInit messages sent by $\mathcal{A}$ with r, the number of computational steps it performs by s, and the number of TagInit messages sent by t.

We do not parameterize the number of explicit challenge and response messages an adversary may issue because it will be determined by the number of TagInit and ReaderInit messages. That is, communication costs are entirely determined by r and t.

Nor do we parameterize the number of SetKey messages sent by $\mathcal{A}$. An adversary able to issue SetKey messages essentially models both the ability to corrupt a tag, since adversaries obtain its internal tag keys, and to manufacture arbitrary tags, since adversaries can then set tag's *key* to an arbitrary value. For this reason, we will consider any tag that receives a SetKey message from the adversary as "corrupted." We assume an adversary is able to send as many SetKey messages as it likes (within its computational step bound s), at any time, to any set of at most $n-2$ tags. The rationale for this $n-2$ bound will be explained in the following section.

2.5 Privacy Experiment

We now present a "privacy experiment" reminiscent of the classic indistinguishability under chosen-plaintext attack (IND-CPA) and under chosen-ciphertext attack (IND-CCA) cryptosystem security experiments. The idea is that an RFID protocol may be considered private for some parameter values if no adversary has a significant advantage in this experiment.

The goal of the adversary in this experiment is to distinguish between two different tags within the limits of its computational power and functionality-call bounds.

That is why at least two tags need to be uncorrupted. In the first phase of this experiment, an adversary $\mathcal{A}$ will be able to issue any messages and perform any computation within its parameter bounds. The adversary $\mathcal{A}$ controls the communication channel between $\mathcal{R}$ and each $\mathcal{T}_i$, and may obtain or corrupt any *sid*, c_j, or r_j values at will.

The adversary selects two uncorrupted tags as challenge candidates. These challenge candidates are removed from the pool of tags at large. One of these challenge candidates is then randomly selected and presented to $\mathcal{A}$ (effectively as a tag oracle). $\mathcal{A}$ may then issue any messages and perform any computation it likes within its parameter bounds, *except* issuing a SetKey command to the tag oracle. (For most protocols, that would make the experiment trivial, although later we briefly explore variants allowing corruption of challenge candidates.)

All key material is generated using a randomized function $\text{Gen}(1^k) \to (key_0, \ldots, key_n)$. This function is considered part of the overall RFID system and thus privacy depends on its characteristics. This function is computable by an adversary. Obviously, its output in the experiment is kept secret from $\mathcal{A}$.

We denote our privacy experiment for an RFID system by $\mathbf{Exp}^{priv}_{\mathcal{A},\mathcal{S}}[k, n, r, s, t]$. Here, $\mathcal{S} = (\text{Gen}, \mathcal{R}, \{\mathcal{T}_i\})$, where $\{\mathcal{T}_i\}$ contains n tags. We let k be a security parameter. Adversary $\mathcal{A}$ with

parameters r, s, and t is denoted by $\mathcal{A}[r, s, t]$, where r, s and t are respective parameters for $r$eader initializations, computation $s$teps, and $t$ag initializations. We detail our experiment in Figure 3.

RFID Privacy Experiment

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

Setup:

(1) $\textsc{Gen}(1^k) \rightarrow (key_0, \ldots, key_n)$.

(2) Initialize $\mathcal{R}$ with $(key_0, \ldots, key_n)$.

(3) Set each $\mathcal{T}_i$'s key to key_i with a SetKey call.

Phase 1 (Learning):

(4) $\mathcal{A}$ may do the following in any interleaved order:

- (a) Make ReaderInit calls, without exceeding r overall calls.
- (b) Make TagInit calls, without exceeding t overall calls.
- (c) Make arbitrary SetKey calls to any $(n-2)$ tags.
- (d) Communicate and compute, without exceeding s overall steps.

Phase 2 (Challenge):

(5) $\mathcal{A}$ selects two tags $\mathcal{T}_i$ and $\mathcal{T}_j$ to which it did *not* send SetKey messages.

(6) Let $\mathcal{T}_0^* = \mathcal{T}_i$ and $\mathcal{T}_1^* = \mathcal{T}_j$ and remove both of these from the current tag set.

(7) Let $b \overset{R}{\leftarrow} \{0, 1\}$ and provide $\mathcal{A}$ access to $\mathcal{T}_b^*$.

(8) $\mathcal{A}$ may do the following in any interleaved order:

- (a) Make ReaderInit calls, without exceeding r overall calls.
- (b) Make TagInit calls, without exceeding t overall calls.
- (c) Make arbitrary SetKey calls to any tag in the current tag set *except* $\mathcal{T}_b^*$.
- (d) Communicate and compute, without exceeding s overall steps.

(9) $\mathcal{A}$ outputs a guess bit b'.

Exp succeeds if $b = b'$.

Fig. 3. The parameterized privacy experiment for the RFID system (Gen, $\mathcal{R}, \{\mathcal{T}_i\}$).

2.6 Privacy Definition

A protocol run within an RFID system $\mathcal{S}$ =(Gen,$\mathcal{R}, \{\mathcal{T}_i\}$) is defined to be private if no adversary $\mathcal{A}[r, s, t]$ has a non-negligible advantage in successfully guessing b in the experiment in Figure 3.

We now define the notion of (r, s, t)-privacy for RFID systems. The variables r, s, and t can be functions of system parameters like k, if desired, as we shall see in our analyses. We use the notation $poly(k)$ to represent any polynomial function of k.

Definition 1 (RFID (r, s, t)-Privacy).
A protocol initiated by $\mathcal{R}$ in an RFID system $\mathcal{S}$ = (Gen,$\mathcal{R}, \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$) with security parameter k is (r,s,t)-private if:

$$\forall \mathcal{A}[r, s, t] \ \Pr\left[\mathbf{Exp}^{priv}_{\mathcal{A},\mathcal{S}}[k, n, r, s, t] \textit{ succeeds in guessing } b.\right] \leq \frac{1}{2} + 1/poly(k).$$

2.7 Implications of (r, s, t)-Privacy

A few observations on our privacy definition are in order. Definition 1 requires that $\mathcal{R}$ not differ significantly in its acceptance rate across tags. Otherwise, $\mathcal{A}$ could just pick, for example, one tag

that $\mathcal{R}$ accepts frequently and one that it accepts less frequently as its challenge candidates. $\mathcal{R}$ must essentially "own" all the tags in $\{\mathcal{T}_i\}$, or at least treat them all equally.

$\mathcal{R}$ also cannot be trivially history-dependent nor can its acceptance be influenced significantly by $\mathcal{A}$. For example, an $\mathcal{R}$ which accepts each tag only a fixed number m times cannot be private in the face of an $\mathcal{A}$ able to make more than $m+1$ ReaderInit calls. Such an adversary could simply complete m legitimate queries to a tag Phase 1, and see if the reader accepts $\mathcal{T}_b^*$ in Phase 2. We discuss exactly this attack in Section 3.

Our definition treats protocol-level privacy issues only. In the real world, there are effectively many possible side channels. As Avoine and Oechslin observe [5], for example, if tags emit a distinctive "radio fingerprint," then no protocol-level countermeasures can prevent privacy infringement. And of course, bearers of privacy-preserving RFID tags are vulnerable to tracking if they additionally carry wireless devices that lack privacy protections.

An important implication of Definition 1 is that the Gen function cannot output low-entropy or strongly correlated keys. Otherwise, $\mathcal{A}$ might have a significant advantage in guessing keys given the output from SetKey calls. This illustrates how privacy by our definition *depends on the entire RFID system* (Gen, $\mathcal{R}$, $\{\mathcal{T}_i\}$). There are classes of $\mathcal{R}$ and Gen that simply cannot be private under this definition.

Remarks: (1) A superficial difference between the Avoine model and our own is that the target tags in the Avoine model are determined not by the adversary, but by a "Challenger." This Challenger has unspecified functionality, but presumably selects target tags uniformly at random. In an asymptotic sense, adversarial and random selection of target tags are equivalent: Random selection will yield any desired pair of target tags with non-negligible probability.
(2) One may consider a variant of $\mathbf{Exp}^{priv}_{\mathcal{A},\mathcal{S}}$ experiment with a stronger adversary $\mathcal{A}$ that is allowed to corrupt $\mathcal{T}_b^*$ in Phase 2 of the experiment, i.e. removing the words "except $\mathcal{T}_b^*$" from experiment step (8c). In most schemes, this will trivially violate privacy. It is worthwhile, however, to consider this definition of *forward (r,s,t)-privacy*. In a scheme with forward privacy of this kind, corrupting a tag does not link it to its past output. (Forward privacy is a goal of the OSK/AO scheme that we analyze below.) Forward privacy is strictly stronger than (r, s, t)-privacy as defined in Definition 1.

2.8 Cross-reader (r, s, t)-privacy

A limitation of our definition is that it models a world in which a reader $\mathcal{R}$ "owns" every tag, i.e., a monolithic and closed system. While this model provides important insights into the privacy properties of an RFID system, different (weaker) definitions can be useful for analyzing real-world scenarios.

Consider an RFID system that contains two types of tags, type A and type B. The reader $\mathcal{R}$ recognizes, i.e., accepts valid tags of type A. But it rejects all tags of type B. Our definition above suggests that such a system can *never* be privacy preserving. An attacker can break the privacy of the system by distinguishing between type A and type B tags.

In the real world, though, such scenarios frequently arise. There are many, independent RFID systems; people carry heterogeneous constellations of tags, such as SpeedPass tags, contactless credit-cards, proximity cards, automobile immobilizer chips, and so forth. The very diversity of tag types leaks information. An unusual collection of tags could even be uniquely identifying. Thus, while our privacy definition is useful for characterizing privacy *within* a single RFID system, an extended or variant definition is needed to analyze multi-verifier privacy—and capture its inherent limitations.

For this purpose, a different (weaker) privacy definition is useful. An attacker capable of mounting unrestricted man-in-the-middle attacks can determine whether or not a given tag is recognized by a given reader $\mathcal{R}$. Such an attacker can, of course, easily violate the privacy of an RFID-tag bearer by determining the presence of type A and type B tags on her person. For this reason—and also because man-in-the-middle attacks can be tricky to mount in RFID environments—we propose a variant of our privacy definition. This variant excludes man-in-the-middle attacks by means of a very simple restriction on the adversary in $\mathbf{Exp}^{priv}_{\mathcal{A},\mathcal{S}}[k,n,r,s,t]$: When $\mathcal{A}$ emits a ReaderInit call, the system first emits TagInit calls with random *sid*s to all tags. We call the modified experiment $\mathbf{Exp}^{cross-read-priv}_{\mathcal{A},\mathcal{S}}[k,n,r,s,t]$, and characterize a system in the obvious way as *cross-reader* (r,s,t)-private.

Cross-reader (r,s,t)-privacy is useful in analyzing an environment with multiple, independent verifiers (RFID readers). We can model such a system as an environment with just one reader $\mathcal{R}$; we simply assume type A and type B tags as above. Again, type A tags are those that lie within the closed system for reader $\mathcal{R}$, and type B tags are those in other, independent systems.

What is especially interesting about cross-reader (r,s,t)-privacy is its intersection with tag *authentication*. As we show below, in a system where $\mathcal{R}$ recognizes all tags, i.e., all tags are of type A, the original Weis et al. "hash-lock" system [26] is (r,s,t)-private and cross-reader (r,s,t)-private for polynomial r,s,t. In a system with tags of types A and B, however, the Weis et al. system is *not* cross-reader private. This is true *because the system does not provide strong authentication for tags*. Without mounting a man-in-the-middle attack, an adversary can simulate a target RFID tag and determine whether it is of type A or B, thereby undermining privacy. As we show, however, by adding strong authentication to the Weis et al. system, we can confer cross-reader (r,s,t)-privacy for polynomial r,s,t in a multi-verifier environment.

Remark: Even cross-reader privacy definition is unachievable when type A and type B tags may be distinguished outside the protocol layer. For example, if the two types operate at different frequencies, an attacker can distinguish between them without reference to any valid reader. In such cases, the best privacy that can be hoped for is indistinguishability, i.e., cross-reader privacy among tags of a given physical type.

3 Analysis of Several Proposed RFID Schemes

We now examine several RFID privacy schemes proposed in the literature: (1) That of Ohkubo, Suzuki, and Kinoshita (OSK) [23] and a variant proposed by Avoine and Oechslin (AO) [6] (with a follow-up by Avoine, Dysli, and Oechslin [4]); (2) That of Nohara, Inoue, Baba, and Yasuura (NIBY) [20]; (3) The YA-TRAP scheme of Tsudik [25]; and (4) A recent "zero-knowledge" scheme of Engberg, Harning, and Jensen (EHJ) [9] (EHJ).

In our analyses of OSK/AO and NIBY, we demonstrate an adversarial algorithm that shows that the algorithms do not achieve strong privacy according to our definition. We believe that this adversarial algorithm is practical and that it reveals a privacy vulnerability that has gone unnoticed in the literature. The vulnerability is similar in both OSK/AO and NIBY. It involves an adversary interacting with a target tag and modifying its state. The aim of the modification is to "mark" the target tag so that it is recognizable by the adversary at a later time. The "mark" takes the form of state that is invalid in the view of a legitimate RFID reader. The adversary effectively plants a timestamp in a tag that is too far in the future to be treated as valid by the reader.

We believe that this vulnerability in tags that do not accept reader input has been overlooked in the literature because it bypasses the natural intuitive view of privacy modeling. For example, in the OSK/AO system, tags do not accept input from a querying reader (apart from a trivial read

command). The only state change in tags is the incrementing of an internal counter. Thus, these systems appear to reduce an adversary to strictly passive attacks. Our formal definition of privacy, however, highlights the fact that an adversary can in fact mount an *active* attack by aggressively changing the internal state of tags.

Tags in OSK/AO and NIBY systems are counter based and output-only. We also examine two systems in which readers modify tag state by means of timestamps.

The YA-TRAP system is a new proposal by Tsudik that has not yet received formal security analysis. Tsudik already notes that the system does not withstand the full range of possible active attacks. With these limitations in mind, Tsudik has designed YA-TRAP for settings in which RFID-tag information is batched, as in warehouses, rather than for consumer applications. Our analysis gives firmer shape to this design aim. We show a privacy limitation in which, as in OSK/AO, a tag may be "marked" by being invalidated. More interestingly, an extension of this attack can "mark" a tag with a unique timestamp that may be indirectly read by an adversary. Our analysis confirms Tsudik's view of the circumscribed settings for which YA-TRAP is appropriate.

The EHJ system is particularly interesting because it is under commercial development, with privacy being emphasized as its distinguishing feature. The vulnerability we highlight in the EHJ system is of a different nature than the others. In that system, an adversary can intercept a flow in the authentication protocol between a tag $\mathcal{T}_i$ and the reader, and interrupt the associated session. By resuming the session at a later time using the intercepted flow, the adversary can uniquely identify $\mathcal{T}_i$.

There is one common thread across all of the vulnerabilities we explore. As our formal definition shows, even a single bit of information—namely whether a tag functions or malfunctions—has an important bearing on RFID-system privacy.

3.1 OSK/AO schemes

Ohkubo, Suzuki and, Kinoshita (OSK) [23] propose a simple scheme for privacy-preserving tag identification in the face of active attacks; the OSK scheme also provides forward security. Suppose that tag i is initialized with secret x_i; let h_1 and h_2 be independent one-way functions (most conveniently modeled as random oracles). When queried, the tag updates its secret key by applying h_1. The tag outputs $ID_{i,t} = h_2(h_1^{(t)}(x_i))$.

Readers in the OSK scheme share secrets with tags. On input $ID_{i,t}$, a reader determines i by brute-force search. OSK propose a fixed upper bound m on the number of time steps over which tags operate. (After m time steps, presumably a tag yields no output or random output while it awaits re-initialization or retirement.) OSK propose that readers pre-compute a giant table $T = \{[ID_{i,t}, (i,t)]\}$ where $1 \leq i \leq n$ and $1 \leq t \leq m$. When structured appropriately, T can support rapid look-up of a tag identifier.

Kinoshita et al. have implemented the OSK protocol in an active RFID tag [17].

Avoine, Dysli, and Oechslin (AO) [6, 4] propose use of *Hellman tables* as a special form of pre-computation on T. Hellman [12] originally studied the problem of *breaking* symmetric keys. He considered the resource requirements of an attacker seeking to recover secret key k from a ciphertext $C = e_k(M)$ on a pre-determined message M. Of course, the attacker might simply perform a brute-force key search, investing computational effort $O(n)$, where n is the size of the full key space. Or she might pre-compute and store a table of size $O(n)$ consisting of $C = \{C_i = e_{k_i}(M)\}_{k_i \in K}$, and simply look up C in the table; this is what OSK originally proposed.

Hellman, however, identified a useful time-space trade-off. An attacker can pre-compute a table of size $O(N^{2/3})$, now known as a Hellman table, that permits successful key search with computational effort only $O(N^{2/3})$. Loosely speaking, the idea is to group sequences of ciphertexts

in C into deterministically traversable chains of size $O(N^{2/3})$. Within the Hellman table are stored only the first and last elements of these chains. Starting with a ciphertext C, the attacker can traverse the induced chain until she locates the last element. By consulting the Hellman table, she can then determine the first element in the chain; on traversing the chain from this point, she can determine k_i.

AO cleverly observed that the problem of symmetric-key search for readers in privacy-preserving RFID systems is nearly identical with that of breaking keys. For a reader to determine which tag it is communicating with, the reader can "crack" the ciphertext C_i to determine k_i and thus T_i. AO apply a variant of the Hellman technique to the OSK system, yielding a scheme with considerably more practical look-up costs. (They in fact propose use of a variant called *rainbow chains* put forth by Oechslin [22].) Rather than constructing a simple look-up table T, a reader can make use of a Hellman table $\tilde{T}$.

The AO approach can thus render the original OSK more practical.[3]

Privacy vulnerability in OSK/AO: Avoine analyzes the OSK (and, by extension, the AO) scheme for tag identification, and concludes that it affords the strongest form of privacy [2]. As we now explain, however, OSK is in fact vulnerable to an attack that undermines strong privacy in the system.[4]

Very simply, an adversary can exploit the upper bound m on the number of tag time steps accommodated in the look-up table T or $\tilde{T}$. The adversary queries a tag until its counter value exceeds m. In essence it mounts a denial-of-service attack against the tag: The adversary exhausts the supply of valid output values stored in the tag.

This strategy depends upon the ability of the adversary to obtain a single bit of information from a reader, namely whether or not it recognizes the output of a given tag as valid. For most natural RFID tag applications, such information will be easily obtainable. For example, a proximity card either opens or fails to open a door, a payment card is either accepted or rejected by a point-of-sale device, and so forth.

The adversarial algorithm in Figure 4 demonstrates that OSK/AO does no achieve (r, s, t)-privacy for $t > m$ and $r \geq 1$ (and $s > m$). Indeed, for these parameters, $\mathbf{Exp}^{priv}_{\mathcal{A},\mathcal{S}}[n, k, r, s, t] = 1$.

OSK/AO Adversarial Algorithm

1. In Phase 1, $\mathcal{A}$ selects a pair of distinct tags $\mathcal{T}_i$ and $\mathcal{T}_j$ uniformly at random.
2. $\mathcal{A}$ sends m queries to $\mathcal{T}_i$ (sends m TagInits).
3. $\mathcal{A}$ submits $\mathcal{T}_i$ and $\mathcal{T}_j$ as its challenge candidates.
4. In Phase 2, $\mathcal{A}$ initializes a protocol between $\mathcal{T}_b^*$ and $\mathcal{R}$.
5. If $\mathcal{R}$ accepts $\mathcal{T}_b^*$, $\mathcal{A}$ guesses $b = 1$, i.e. $\mathcal{T}^* = \mathcal{T}_j$. Else $\mathcal{A}$ guesses $b = 0$, i.e. $\mathcal{T}^* = \mathcal{T}_i$.

Fig. 4. Adversarial algorithm for OSK/AO

A simple modification of the adversarial algorithm in Figure 4 allows an adversary to distinguish among multiple tags by desynchronizing them in different degrees. The adversary queries $\mathcal{T}_i$ a total

[3] Note, however, that because a Hellman table $\tilde{T}$ requires pre-computation over a fixed number of entries, it is unclear how to modify the table to accommodate new entries. Insertion of new entries into a simple table T is more straightforward.

[4] The vulnerability we demonstrate appears to exist even within the Avoine privacy model. The analysis in [2] may simply represent an oversight.

of q_i times to "mark" it; the adversary lets q_i be unique for each tag. On later sighting a target tag $\mathcal{T}$, the adversary queries $\mathcal{T}$ a total of m times. Let $r_1, r_2, \ldots, r_m$ be the resulting outputs. The adversary feeds these outputs to the reader in reverse order $r_m, r_{m-1}, \ldots$ until the reader accepts some r_k. The adversary concludes that the counter value for the tag $q = m - k$. Assuming that no normal reading has taken place during the interval of time between the "marking" and the sighting of the tag, the adversary concludes that if $q = q_i$, the sighted tag is $\mathcal{T}_i$. Note that it the adversary staggers q_i values sufficiently, this strategy works even when some limited amount of normal reading has effected changes in tag counter values.

The adversarial algorithm we describe here would appear to be feasible in a practical setting. In [6], Avoine and Oechslin propose $m = 1024$ as an example parameter for a practically realizable AO system. As EPC tags (Class 1 Gen 2) have a theoretical read rate of up to 1000 per second [1], the adversarial algorithm we describe could well be feasible in many systems. In the original OSK system, which is less efficient than the AO variant, m would have to be considerably smaller to permit practical deployment, and the vulnerability we describe more easily exploitable.

Remarks: (1) The adversarial strategy we describe for OSK/AO here induces desynchronization between tags and the reader. It is tempting to try to evade this problem with a protocol modification in which the reader rejects input from a $\mathcal{T}_i$ if the current tag counter q_i differs by more than d from the previous counter perceived by the reader. This does not solve the privacy problem, however: It is equivalent to setting $m = d$. Moreover, if d is small, then the system becomes vulnerable to denial-of-service attacks. Another possible countermeasure is simply to cause tags to stop yielding output after the m^{th} query. Again, this creates a vulnerability to denial-of-service attacks.

In a natural setting, the adversary might not always have access to fresh tags, i.e., tags whose counters are equal to 0. A simple modification of the adversarial algorithm in Figure 4, however, can still distinguish between tags. By querying a target tag a sufficient number of times, the adversary can "mark" it. For example, suppose that $m = 1024$, and suppose that prior to the testing phase of our definition experiment, $\mathcal{T}_i$ and $\mathcal{T}_j$ have each been queried q_i and q_j times, where $q_i, q_j \in_U [0, 1, 2, \ldots, 50]$. Then the adversary merely has to submit 51 queries to $\mathcal{T}_i$ to "mark" it such that it may be distinguished from $\mathcal{T}_j$ in the guessing phase 2.

One possible countermeasure to such attacks is "throttling," i.e., deliberate slowing of the rate of tag response. A rather different privacy model for tags would be needed to accommodate the effect of this countermeasure, however [13].
(2) MSW carries a similar vulnerability in principle, as tags can support only a pre-determined number of queries, as observed by [18]. Under realistic parameterization, however, a privacy-infringing attack would require an infeasibly large number of queries. In a practical sense, therefore, MSW is not vulnerable.

3.2 Unlinkable ID-Matching Scheme

At WPES '05, Nohara, Inoue, Baba, and Yasuura (NIBY) [20] proposed a scheme similar in flavor to MSW. Their basic protocol permits leaves of potentially non-uniform depth within a tree T of maximum depth d. A tag is presumed to output random values when depth-first-search queries exceed the depth of its corresponding leaf.

An adversary can undermine strong privacy in this scheme in much the same manner as OSK/AO; non-uniformity in the NIBY scheme, however, renders it more vulnerable. In particular, for a given tag $\mathcal{T}_i$, an adversary can learn the depth d_i of the corresponding leaf in the identifier tree with overwhelming probability using the following algorithm:

Algorithm $\mathsf{Probe}(d, \mathcal{T}_i)$

1. Send d queries to $\mathcal{T}_i$ (send d TagInits).; let $Z_e = \{z_e^{(i)}\}_{e=1}^{d}$ denote the sequence of d outputs yielded by the e^{th} query, for $1 \leq e \leq d$.
2. For every e, $1 \leq e \leq d$, replace $z_e^{(d)}, z_e^{(d-1)}, \ldots, z_e^{(e)}$ with random values.
3. Feed $Z_e, Z_{e-1}, \ldots$ to $\mathcal{R}$ until $\mathcal{R}$ accepts some $Z_{e'}$.
4. Output e'.

Suppose that the probability that two leaves in the tree T differ in depth is non-negligible. With use of the Probe algorithm as a subroutine, we can construct an adversarial algorithm that breaks (r, s, t)-privacy in NIBY for $r, t > 3d$ (and $s > 2d$). This algorithm is depicted in Figure 5.

NIBY Adversarial Algorithm

1. In Phase 1, $\mathcal{A}$ selects a pair of distinct tags $\mathcal{T}_i$ and $\mathcal{T}_j$ uniformly at random.
2. $\mathcal{A}$ computes $d_i = \mathsf{Probe}(d, \mathcal{T}_i)$ and $d_j = \mathsf{Probe}(d, \mathcal{T}_j)$.
3. $\mathcal{A}$ submits $\mathcal{T}_i$ and $\mathcal{T}_j$ as its challenge candidates.
4. In Phase 2, $\mathcal{A}$ computes $d' = \mathsf{Probe}(d, \mathcal{T}_b^*)$.
5. If $d' = d_j$, $\mathcal{A}$ guesses $b = 1$, i.e. $\mathcal{T}^* = \mathcal{T}_j$. Else, $\mathcal{A}$ guesses $b = 0$, i.e. $\mathcal{T}^* = \mathcal{T}_i$.

Fig. 5. Adversarial algorithm for NIBY

The authors analyze the security of a restricted case of their scheme in which all leaves are presumed to have equal depth. The adversarial algorithm we describe here is not applicable in that case. Our analysis therefore suggests that use of a uniform tree for NIBY is essential to privacy.

3.3 YA-TRAP: Yet Another Trivial RFID Authentication Protocol

Tsudik describes an RFID authentication protocol that he calls YA-TRAP (Yet Another Trivial RFID Authentication Protocol) [25]. As discussed above, Tsudik aims this protocol at environments in which tag information is processed in batches, rather than for more fine-grained applications like access control and tagging of individual consumer items. Tsudik has not yet formally analyzed the security properties of YA-TRAP; such analysis is forthcoming [24]. In the meantime, our investigation here confirms the importance of confining YA-TRAP to the limited settings for which it is designed.

In YA-TRAP, the reader shares a unique key x_i with tag $\mathcal{T}_i$. A tag $\mathcal{T}_i$ also stores an internal timestamp t_i, namely the time at which it was last interrogated by a reader.

To interrogate a tag, a reader transmits the current time $t_{\mathcal{R}}$. The tag compares $t_{\mathcal{R}}$ with t_i. If $t_{\mathcal{R}}$ is stale with respect to t_i, i.e., $t_{\mathcal{R}} \leq t$, then the tag outputs a random response. Otherwise, the tag outputs $R = H_{x_i}[t_{\mathcal{R}}]$, where H_{x_i} represents an HMAC computed with secret key x_i; the tag also sets $t_i \leftarrow t_{\mathcal{R}}$, i.e., updates its timestamp.[5]

To validate the response of a tag, a reader checks whether $R = H_{x_j}[t_{\mathcal{R}}]$ for any secret key x_j in its database. If so, the reader accepts the tag, otherwise it rejects it.

We consider two adversarial strategies against the YA-TRAP protocol. The first is based on denial of service. An adversary queries a tag with t_{max} to "mark" it, where t_{max} is some time in the far future. The tag sets its internal time value $t_i \leftarrow t_{max}$, and outputs random values in

[5] The premise is that HMAC output is indistinguishable from a random distribution, so successful or failed tag responses are indistinguishable to an adversary. While this premise is not true of HMAC in general, it may be achieved with an appropriate underlying hash function.

response to all future queries. Thus, a reader rejects the tag in all future sessions, permitting the adversary to distinguish the tag from one that is "unmarked." We outline this adversarial strategy in Figure 6.

YA-TRAP Adversarial Algorithm

1. In Phase 1, $\mathcal{A}$ selects an arbitrary pair of distinct tags $\mathcal{T}_i$ and $\mathcal{T}_j$.
2. $\mathcal{A}$ sends a TagInit to $\mathcal{T}_i$ and sends $c_0 = t_{max}$.
3. $\mathcal{A}$ submits $\mathcal{T}_i$ and $\mathcal{T}_j$ as its challenge candidates.
4. In Phase 2, $\mathcal{A}$ initializes a query by sending a ReaderInit to $\mathcal{R}$ and a TagInit to $\mathcal{T}_b^*$.
5. $\mathcal{A}$ relays $\mathcal{R}$'s challenge $c_0 = t_{\mathcal{R}}$ to $\mathcal{T}_b^*$ and the response r_0 back to $\mathcal{R}$.
6. If $\mathcal{R}$ accepts r_0, $\mathcal{A}$ guesses $b = 0$, i.e. $\mathcal{T}^* = \mathcal{T}_i$, and otherwise guesses $b = 1$.

Fig. 6. Adversarial DoS algorithm for YA-TRAP

As the adversarial algorithm in Figure 6 demonstrates, the YA-TRAP protocol does not achieve $(r, s, t) = (1, O(k), 2)$-privacy. It is worth noting that the phase 1 portion of the attack does not involve $\mathcal{R}$ and can be mounted by an adversary with access to $\mathcal{T}_i$ alone. The phase 2 portion does not require a man-in-the-middle setting. A real-life adversary could wait until $\mathcal{R}$ tries to legitimately query $\mathcal{T}_b^*$, and then eavesdrop, i.e., play a purely passive role.

Building on this first strategy, we arrive at a second and more potent adversarial strategy that allows an adversary to "mark" a tag uniquely—not simply to invalidate it. To mark a tag $\mathcal{T}_i$ in this way, the adversary selects a future time u_i. The adversary queries tag $\mathcal{T}_i$ with it, causing $t_i \leftarrow u_i$.

The time u_i serves as a uniquely individuating "mark" for tag $\mathcal{T}_i$. Starting at any time prior to u_i, an attacker can test whether a given, unidentified tag $\mathcal{T}$ is in fact $\mathcal{T}_i$. The process involves two stages:

1. **Probing T:** The adversary selects two slightly spaced times u_i^{before} and u_i^{after} such that $u_i^{before} \leq u_i < u_i^{after}$. The adversary then queries the tag $\mathcal{T}$ with u_i^{before} and u_i^{after}, obtaining responses r_{before} and r_{after}.
2. **Testing the results:** The adversary interacts with $\mathcal{R}$ at time u_i^{before} and u_i^{after} and replays responses r_{before} and r_{after}. If the reader accepts r_{before} and rejects r_{after}, then it is almost certain that $\mathcal{T} = \mathcal{T}_i$.

Note that this strategy does not require an active man in the middle. $\mathcal{A}$ can interact with the tag and the reader at separate times. This adversarial algorithm can be extended, of course, to mark multiple tags uniquely. In the real-world, an attacker could identify an individual on the basis of a single marked tag $\mathcal{T}_i$—regardless of how many other tags the consumer is carrying.

In the setting for which YA-TRAP was designed, where a reader batches tag information exclusively for processing by a back-end system that does not reveal information to an adversary about the results of its identification of tags, YA-TRAP is not subject to the attacks we describe.

Remark: If the granularity of time in the system is small, then it may be difficult for $\mathcal{A}$ to interact with $\mathcal{R}$ at exactly time u_i^{before} or u_i^{after}. To increase its likelihood of success, $\mathcal{A}$ can query the tag at *multiple* times around u_i^{before} and u_i^{after}, i.e., harvest a staggered set of responses from the target tag.

3.4 A "Zero-Knowledge" RFID Protocol

Engberg, Harning, and Jensen [9] (EHJ) have proposed a privacy-preserving RFID protocol that they call "zero-knowledge device authentication." (The protocol, as they note, is not zero-knowledge in the received sense of the term.) Media reports suggest that this protocol is serving as the basis of a commercial offering by a company called RFIDSec [16]. As we show, their protocol does not provide strong privacy according to our definition.

The authors do not fully explain a core aspect of their system. Their authentication protocol involves reader-side initiation, but they do not indicate how a reader is to determine which tag it is communicating with (and thus which key to use). We therefore assume that the number of tags in the vicinity of the reader is small, such that it can try all keys in attempting to communicate with a tag. We also gloss over the question of how a reader singulates a tag, as this has little bearing on our analysis.

The protocol involves two passes. To use the notation of EHJ, DT is a timestamp, and RSK is a random nonce selected by the reader. The value $SSDK$ is a tag-specific secret key. We let $\oplus$ denote the XOR operation:

1. $\mathcal{R} \to \mathcal{T}$: $[DT, (RSK \oplus h(DT \oplus SSDK)), h(RSK \oplus SSDK)]$
2. $\mathcal{T} \to \mathcal{R}$: $h(RSK \oplus SSDK \oplus DT)$

A tag stores the timestamp DT' that it received in the last successful protocol execution. When it receives a new authentication message based on some DT, it only responds if the message is valid (with respect to $SSDK$) and if $DT > DT'$. Otherwise the tag outputs no response.

Privacy vulnerability in EHJ: A privacy vulnerability in this system lies in the way that a tag decides whether or not to respond to a first-flow message. The first flow of the protocol is meant to permit the reader to authenticate to the tag. This flow, however, is keyed according to the tag with which the reader communicates. Suppose that adversary intercepts and stores a first flow message M directed to $\mathcal{T}_i$. If the adversary transmits M to $\mathcal{T}_i$ (while M is still fresh), then $\mathcal{T}_i$ will output a response. Any other tag will disregard M.

In Figure 7, we specify an adversarial algorithm that shows that EHJ is not (r, s, t)-private for any $r \geq 1$ and $t \geq 2$.

EHJ Adversarial Algorithm

1. In Phase 1, $\mathcal{A}$ selects a pair of distinct tags $\mathcal{T}_i$ and $\mathcal{T}_j$ uniformly at random.
2. The attacker causes the reader to issue an authentication message M to $\mathcal{T}_i$ at time DT; $\mathcal{A}$ stores M and does not allow it to reach the tag.
3. In Phase 2, the attacker transmits M to tag $\mathcal{T}_b^*$.
4. If $\mathcal{T}_b^*$ outputs a response, $\mathcal{A}$ guesses $b = 0$, i.e. $\mathcal{T}^* = \mathcal{T}_i$. Else $\mathcal{A}$ guesses $b = 0$, i.e. $\mathcal{T}^* = \mathcal{T}_j$.

Fig. 7. Adversarial algorithm for EHJ

Of course, an adversary can extend this strategy to discriminate easily among multiple tags, $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_{n-1}$. The adversary need simply harvest first-flow messages $M_0, \ldots M_{n-1}$ for each of the tags. To identify a given tag, the adversary tests each of its harvested messages.

One might think that if a tag does not respond to a read request that no privacy issue arises. This idea is misleading for two reasons. First of all, the absence of a tag can be conspicuous. For

example, if the tag is a disabled proximity card, the fact of an individual being unable to enter a building is an easily observable event. More importantly, though, in the EHJ scheme, a tag *must provide some indication of its existence at the outset*. If it does not, then the reader cannot determine that it should initiate the first flow. Any practical embodiment of EHJ therefore would probably need at a minimum to involve a "liveness" signal from tags.

Vulnerability in EHJ with shared keys: Even if SSDK is shared across tags (which would make the protocol more scalable), a privacy-infringing attack is still possible. In particular, the adversary causes the reader to set timestamps DT_0 and DT_2 in tags $\mathcal{T}_i$ and $\mathcal{T}_j$ respectively by causing two authentication sessions to run to completion. The adversary also causes the reader to output a first-flow message M at time DT_1, where $DT_0 < DT_1 < DT_2$. The message M will then be fresh with respect to $\mathcal{T}_i$, but stale with respect to $\mathcal{T}_j$, permitting the adversary to distinguish between them. This adversarial strategy too can be extended in an obvious manner to distinguish among multiple tags.

Repairing EHJ: The EHJ protocol can easily be modified to alleviate the privacy vulnerability we present here. Rather than ignoring an authentication message with a stale timestamp, a tag can output a random or pseudorandom response (computationally indistinguishable from a legitimate one). In the absence of a carefully specified protocol, however, we do not make any formal claims about the security of this variant. Indeed, the inevitable presence of clock skew and the naturally slow response times of RFID tags make timestamp-based protocols like this one tricky to analyze.

One might think that tag silence confers privacy protection by concealing the presence of RFID tags on a given person or in a given environment. We note, however, that tag silence probably confers minimal privacy protection. Even if a tag does not reply to a reader query, its presence is still likely to be detectable through simple power analysis of reflected RF reader emissions. Recent work by Oren and Shamir [21] suggests the potency of such attacks. Logical-layer silence does not necessarily imply RF silence.

4 Hash-Locks

Weis, Sarma, Rivest, and Engels [26] (WSRE) first advanced the general approach of key search for RFID-tag identification. They proposed two basic schemes dubbed "hash-locks" and "randomized hash-locks". We now show that deterministic hash-locks are not private by our definition. We will show, however, that randomized hash locks are (r, s, t)-private for any r, s, and t polynomial in k. We propose a new improved randomized hash lock that is both private and resistant to replay attacks, and also satisfies our cross-reader privacy definition.

4.1 Deterministic Hash-Locks

Hash-locks are an access control mechanism designed for efficiency. The general idea is as follows. A tag normally resides in a locked state in which it only responds to reader queries with a temporary $metaID$, rather than some permanent identifier. This $metaID$ is the hash of some random nonce selected by $\mathcal{R}$ and programmed into the tag. $\mathcal{R}$ stores both the $metaID$ and its pre-image for its later interaction with the tag.

$\mathcal{R}$ may unlock a tag by sending it the pre-image of its $metaID$. The tag verifies that this value hashes to the $metaID$. Alternatively, the tag need not use a hash function at all. It can instead be locked with some secret $metaKey$. The tag is programmed with a $(metaID, metaKey)$ pair when locked, and unlocked with the $metaKey$. The hash-function based scheme may be justified,

however, if the hardware cost of implementing a hash function is less than EEPROM costs of $metaKey$ storage or if the $metaID$ needs to be assigned in the clear.

Using the notation of this paper, the session identifier sid will play the role of the $metaID$. Let $h : \{0,1\}^* \to \{0,1\}^{2k}$ be a random oracle. Hash-locks work as follows:

Setup:

(1) GEN outputs a uniformly chosen vector of k-bit keys $(key_1, \ldots, key_n)$.
(2) $\mathcal{R}$ is initialized with GEN's output.
(3) Each $\mathcal{T}_i$ is initialized by a SETKEY call with a unique key_i.
(4) $\mathcal{R}$ makes a TAGINIT call to each $\mathcal{T}_i$ with $sid = h(nonce)$ for a random k-bit $nonce$.
(5) $\mathcal{R}$ stores each $(sid, nonce)$ pair internally.

System execution:

(6) $\mathcal{R}$'s challenge c_0 to a tag $\mathcal{T}$ is empty.
(7) $\mathcal{T}$'s response $r_0 = sid$.
(8) $\mathcal{R}$'s challenge $c_1 = nonce$.
(9) If $h(c_1) = sid$, then $\mathcal{T}$'s r_1 is some identifier determined by key.
(10) Otherwise $\mathcal{T}$'s r_1 is empty.

Clearly, this protocol is not private by our definition since a tag's response r_0 is always its current sid. Or $\mathcal{A}$ can simply eavesdrop on the protocol between $\mathcal{T}_b^*$ and $\mathcal{R}$ to obtain some identifier. Thus, deterministic hash-locks are not (r, s, t)-private for any non-trivial (r, s, t) values. In fact, any protocol where tags respond deterministically is necessarily not private by our definition.

4.2 Randomized Hash-Locks

In contrast, WRSE's *randomized* hash-lock scheme is a mechanism for private tag identification that is in fact (r, s, t)-private in the random oracle model. We present a short proof of this fact. Letting $h : \{0,1\}^* \to \{0,1\}^{2k}$ be a random oracle, randomized hash-locks work as follows:

Setup:

(1) GEN outputs a uniformly chosen vector of k-bit keys $(key_1, \ldots, key_n)$.
(2) $\mathcal{R}$ is initialized with GEN's output.
(3) Each $\mathcal{T}_i$ is initialized by a SETKEY call with a unique key_i.
(4) Tags do not store any session state or sid.

System execution:

(5) $\mathcal{R}$'s challenges c_0 are empty.
(6) $\mathcal{T}$'s response $r_0 = (nonce, h(nonce||key))$, for a random k-bit $nonce$.
(7) $\mathcal{R}$ searches for key_i among GEN's output such that r_0 contains $h(nonce||key_i)$.
(8) $\mathcal{R}$ accepts if such a key_i exists and rejects otherwise.

A simple illustration is given in Figure 8.

Theorem 1 (Randomized Hash-Lock Privacy). *Randomized hash-locks are (r, s, t)-private in the random oracle model, for any polynomially bounded $\mathcal{A}$, i.e., any r, s, t polynomial in k.*

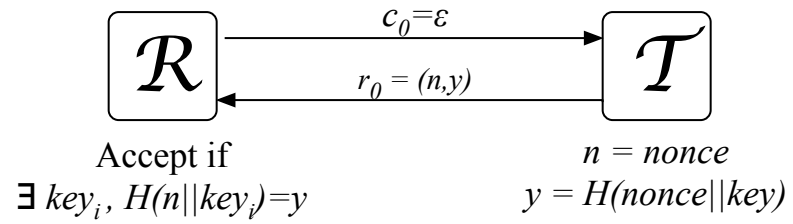

Fig. 8. Illustration of the randomized hash-lock protocol.

Proof: We specify a simulator Sim for $\mathcal{T}_b^*$ in the privacy experiment $\mathbf{Exp}^{priv}$. Sim does not possess knowledge of the value of b or any keys k_i. $\mathcal{A}$'s interaction with Sim will be computationally indistinguishable from an interaction with $\mathcal{T}_b^*$. Thus, we demonstrate that $\mathcal{A}$ gains no knowledge from its interaction with $\mathcal{T}_b^*$ in the real RFID system $\mathcal{S}$.

Recall that $\mathcal{A}$ selects two challenge tags $\mathcal{T}_i$ and $\mathcal{T}_j$ from the population of uncorrupted tags. Let L be the full list of pairs $\{(nonce, h(nonce \parallel key))\}$ output by $\mathcal{T}_i$ and $\mathcal{T}_j$ during the learning phase of the experiment. Let $H()$ represent the random oracle for $h()$ in our system.

During the challenge phase, Sim simulates the result of a TagInit call to $\mathcal{T}_b^*$ by generating a random nonce/hash pair (n, y) and appending it to a list L' (empty at the beginning of the challenge change). In addition to any valid tag pair, $\mathcal{R}$ accepts any pair in L'.

In order for $\mathcal{A}$ to distinguish between the simulated challenge phase and a real challenge phase, $\mathcal{A}$ must determine that some pair $(n, y) \in L'$ is invalid for both $\mathcal{T}_i$ and $\mathcal{T}_j$. As a necessary condition for this determination, $\mathcal{A}$ must identify a pair (n, z) that is valid for $\mathcal{T}_i$ or $\mathcal{T}_j$, but such that $z \neq y$. (In other words, $\mathcal{A}$ must rule out (n, y) as a valid pair in order to determine that Sim is present.) Consequently, one of the following three conditions must occur at some point in the course of the experiment:

1. *There is a nonce value n such that $(n, y) \in L'$ and $(n, z) \in L$ for some pair (y, z)*: Since $\mathcal{A}$ may make at most t calls to tags, we have $|L'|, |L| \leq t$. As nonces are random k-bit values, and thus the space of nonces is 2^k, it follows that this condition occurs with probability at most $t^2/2^k$.
2. *$\mathcal{A}$ submits to $H()$ a query of the form $n \parallel k_i$ or $n \parallel k_j$*: Given that $H()$ is a random oracle, its outputs reveal no information about secret keys. Hence the probability that $\mathcal{A}$ can successfully submit a query of the form $n \parallel k_i$ or $n \parallel k_j$ is at most $2s/2^k$, and thus negligible.
3. *For a pair $(n, y) \in L'$, $\mathcal{A}$ submits and $\mathcal{R}$ accepts a pair (n, z), where $z \neq y$*: Barring condition 1 and condition 2, and given that $H()$ is a random oracle, $\mathcal{A}$ can submit a pair $(n, z) \notin L, L'$ only by guessing z by brute force. The probability of this event is at most $r/2^k$.

Thus $\mathcal{A}$ can distinguish Sim from $\mathcal{T}_b^*$ with probability at most $(r+2s+t^2)/2^k$, which is negligible for polynomially bounded $\mathcal{A}$. □

4.3 Improved Randomized Hash-Locks

Since r_0 does not depend on any reader input, tag outputs are *vulnerable to replay*. Consequently, as explained in section 2.8, the WRSE system does *not* provide privacy in a multi-verifier model. In particular, it does not provide cross-reader (r, s, t)-privacy.

Fortunately, this is an easily addressable flaw. Instead of sending an empty c_0, $\mathcal{R}$ can instead send $c_0 = nonce_{\mathcal{R}}$, where $nonce_{\mathcal{R}}$ is generated uniformly at random. (Alternatively, *sid* could

contain this nonce value.) A tag will generate its own nonce $nonce_{\mathcal{T}}$ and send the value $r_0 = (nonce_{\mathcal{T}}, h(nonce_{\mathcal{R}}||nonce_{\mathcal{T}}||key))$. The reader $\mathcal{R}$ will then search for a key among GEN's output that would generate r_0.

Adversaries cannot replay a previously used r_0 to a $\mathcal{R}$, since with high probability, it will not match the $nonce_{\mathcal{R}}$ value generated by $\mathcal{R}$ for that session. Unlike the OSK/AO or NIBY protocol, the tag does not store internal state based on $nonce_{\mathcal{R}}$. Thus there is no counter or timestamp that can be manipulated by an adversary. Unlike EHJ, the tag always yields an output value.

We do not formally analyze this variant protocol here.

5 Conclusion: Further Research Directions

We have proposed new privacy definitions strong enough to capture highly stringent application-level design requirements for RFID systems in the real world. We believe that the relative simplicity of these definitions is useful for the design and analysis of privacy-preserving RFID protocols. In this paper we have examined several published systems that fail to fulfill our privacy definition. In so doing, we believe that we have highlighted potential design flaws.

That said, our proposed privacy definitions are just a starting point. They certainly do not capture the full spectrum of real-world needs. We propose two important areas for further work.

Stronger definitions: In one sense, even our definition of strong privacy is not strong enough, as it does not capture the effects of side information. Side-channel attacks are pertinent to newly emerging protocol proposals. For example, in independent and contemporaneous work, Burmester, Le, and de Medeiros [7] propose a new protocol called O-TRAP that is similar in flavor to hash locks. They propose a definition of privacy and security in the Universal Composability framework and prove that O-TRAP fulfills their definition. We believe that O-TRAP is also provably secure with respect to our definition of strong privacy.

O-TRAP, however, is *not* secure according to a variant of our definition in which the reader functionality outputs timing information. The reader (or back-end server) in the O-TRAP protocol performs a single table lookup in the optimistic case where a tag is synchronized with a reader; otherwise, it performs a search of the full key space for the system. If an adversary can distinguish between these two cases based on timing—which seems a serious, practical threat—she can "mark" tags by de-synchronizing them, much as we do in the attacks we describe above for OSK/AO and NIBY. (The reader in O-TRAP could process tags in uniform time, but then the system would be no more efficient than brute-force key search, like the improved hash-lock scheme we propose above.) Thus, a small extension of our model in which the reader outputs information about computation time can shed critical light on protocol design. Modeling other forms of side information may prove equally valuable.

Weaker definitions: On the other hand, some aspects of our definitions (even the cross-reader one) may actually be *too* strong. For RFID tags capable of only symmetric-key cryptography, we believe that our definition may require the reader to perform brute-force search to identify tags—at least in cases of de-synchronization. Such a system scales poorly. Obviously public-key cryptography would alleviate this burden, but is impractical for the vast majority of RFID devices. (Indeed, many classes of RFID tags will be incapable of executing even standard symmetric-key protocols for some years.) While the cryptographic literature often aims at increasingly strong definitions and protocols, we emphasize like [14] that a fertile and essential area of investigation is definitions and protocols for RFID privacy that are *weaker*, but more practical and useful.

Acknowledgments:

Thanks to Gildas Avoine, Burt Kaliski, Shingo Kinoshita, Miyako Ohkubo, Koutarou Suzuki, and Gene Tsudik for their very helpful comments on this work.

References

1. Alien Technology. Alien Technology Corporation achieves another step toward pervasive, economic RFID with announcement of 12.9 cent RFID labels, September 2005. Alien Technology Press Release. Referenced 2006 at http://www.alientechnology.com.
2. G. Avoine. Adversarial model for radio frequency identification, 2005. Cryptology ePrint Archive, Report 2005/049. Referenced 2006 at http://eprint.iacr.org.
3. G. Avoine. Security and privacy in RFID systems. http://lasecwww.epfl.ch/∼gavoine/rfid/, 2006.
4. G. Avoine, E. Dysli, and P. Oechslin. Reducing time complexity in RFID systems. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography – SAC 2005*, Lecture Notes in Computer Science. Springer-Verlag, 2005.
5. G. Avoine and P. Oechslin. RFID traceability: A multilayer problem. In A. Patrick and M. Yung, editors, *Financial Cryptography – FC '05*, volume 3570 of *Lecture Notes in Computer Science*, pages 125–140. Springer-Verlag, 2005.
6. G. Avoine and P. Oechslin. A scalable and provably secure hash based RFID protocol. In F. Stajano and R. Thomas, editors, *The 2nd IEEE International Workshop on Pervasive Computing and Communication Security – PerSec 2005*, pages 110–114. IEEE, IEEE Computer Society Press, 2005.
7. M. Burmester, T. van Le, and B. de Medeiros. Provably secure ubiquitous systems: Universally composable RFID authentication protocols, 2006. Referenced 2006 at http://eprint.iacr.org/2006/131.pdf.
8. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. IACR ePrint Report 2000/067. Referenced 2006 at http://eprint.iacr.org/2000/067. Extended version of FOCS '01 paper.
9. S.J. Engberg, M.B. Harning, and C.D. Jensen. Zero-knowledge device authentication: Privacy and security enhanced RFID preserving business value and consumer convenience. In *Second Annual Conference on Privacy, Security, and Trust*, 2004. Referenced 2006 at http://www.obivision.com/Papers/PST2004_RFID_ed.pdf.
10. K. P. Fishkin, S. Roy, and B. Jiang. Some methods for privacy in RFID communication. In *1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004)*, 2004.
11. P. Golle, M. Jakobsson, A. Juels, and P. Syverson. Universal re-encryption for mixnets. In T. Okamoto, editor, *RSA Conference - Cryptographers' Track (CT-RSA)*, volume 2964 of *Lecture Notes in Computer Science*, pages 163–178, 2004.
12. M. Hellman. A cryptanalytic time-memory tradeoff. *IEEE Transactions on Information Theory*, IT-26:401–406, 1980.
13. A. Juels. Minimalist cryptography for low-cost RFID tags. In C. Blundo and S. Cimato, editors, *The Fourth International Conference on Security in Communication Networks – SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 149–164. Springer-Verlag, 2004.
14. A. Juels. RFID security and privacy: A research survey. *IEEE Journal on Selected Areas in Communication*, 24(2), February 2006.
15. A. Juels, R.L. Rivest, and M. Szydlo. The blocker tag: Selective blocking of RFID tags for consumer privacy. In V. Atluri, editor, *8th ACM Conference on Computer and Communications Security*, pages 103–111. ACM Press, 2003.
16. F. Kahn. Can zero-knowledge tags protect privacy? *RFID Journal*, September 2005. Referenced 2006 at http://www.rfidjournal.com/article/articleview/1891/1/1/.
17. A. Kinoshita, M. Ohkubo, F. Hoshino, G. Morohashi, O. Shionoiri, and A. Kanai. Privacy enhanced active RFID tag. In *International Workshop on Exploiting Context Histories in Smart Environments*, May 2005.

18. D. Molnar, A. Soppera, and D. Wagner. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography – SAC 2005*, Lecture Notes in Computer Science. Springer-Verlag, 2005.
19. D. Molnar and D. Wagner. Privacy and security in library RFID : Issues, practices, and architectures. In B. Pfitzmann and P. McDaniel, editors, *ACM Conference on Communications and Computer Security*, pages 210 – 219. ACM Press, 2004.
20. Y. Nohara, S. Inoue, K. Baba, and H. Yasuura. Quantitative evaluation of unlinkable ID matching schemes. In *Workshop on Privacy in the Electronic Society (WPES)*, 2005.
21. M.C. O'Connor. EPC tags subject to phone attacks. *RFID Journal*, February 2006. Referenced 2006 at http://www1.rfidjournal.com/article/articleview/2167/1/1/.
22. P. Oechslin. Making a faster cryptanalytic time-memory trade-off. In D. Boneh, editor, *Advances in Cryptology – CRYPTO '03*, pages 617–630. Springer-Verlag, 2003. LNCS no. 2729.
23. M. Ohkubo, K. Suzuki, and S. Kinoshita. Efficient hash-chain based RFID privacy protection scheme. In *International Conference on Ubiquitous Computing – Ubicomp, Workshop Privacy: Current Status and Future Directions*, 2004.
24. G. Tsudik, 2006. Personal commuications.
25. G. Tsudik. YA-TRAP: Yet another trivial RFID authentication protocol. In *PerCom '06*, 2006. To appear. Referenced 2006 at http://lasecwww.epfl.ch/∼gavoine/download/papers/WIP-12.pdf.
26. S. Weis, S. Sarma, R. Rivest, and D. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In D. Hutter, G. Müller, W. Stephan, and M. Ullmann, editors, *International Conference on Security in Pervasive Computing – SPC 2003*, volume 2802 of *Lecture Notes in Computer Science*, pages 454–469. Springer-Verlag, 2003.