Modeling Privacy for Off-line RFID Systems*

Flavio D. Garcia** and Peter van Rossum

Institute for Computing and Information Sciences,
Radboud University Nijmegen, The Netherlands.
{flaviog,petervr}@cs.ru.nl

Abstract. This paper establishes a novel model for RFID schemes where readers are not continuously connected to the back office, but only periodically. Furthermore, adversaries are not only capable of compromising tags, but also of compromising readers. This more properly models large scale deployment of RFID technology such as in public transport ticketing systems and supply-chain management systems. In this model we define notions of security (only legitimate tags can authenticate) and of privacy (no adversary is capable of tracking legitimate tags). We show that privacy is always lost at the moment that a reader is compromised and we develop notions of forward and backward privacy with respect to reader corruption. This models the property that tags cannot be traced, under mild additional assumptions, for the time slots before and after reader corruption. We exhibit two protocols that only use hashing that achieve these security and privacy notions and give proofs in the random oracle model.

1 Introduction

During the last decade, the use of RFID technology has expanded enormously. It is currently deployed in electronic passports, tags for consumer goods, public transport ticketing systems, race timing, and countless other applications.

The widespread use of RFID has raised privacy concerns. Since most RFID tags will send a unique identifier to every reader that attempts to communicate with it, an adversary could build an "RFID profile" of an individual, i.e., the collection of unique identifiers of the RFID tags that the individual usually carries. This profile could be used to track this person, or to infer behavior such as spending or traveling patterns, jeopardizing this person's privacy.

For simple RFID applications (for instance the tagging of products in stores to enable RFID based cash registers), the privacy problems could be solved by sending the kill-command to the RFID tags (upon leaving the store). This,

* The work described in this paper has been partially supported by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II.

** Partially supported by the research program Sentinels (www.sentinels.nl), project PEARL (7639). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

however, is useless for situations such as access control, where legitimate readers need to verify the authenticity of tags.

Various RFID schemes using cryptographic techniques that guarantee both security (authenticity of accepted tags) and privacy have been proposed. Because RFID tags are low-cost and low-power devices, they are limited in the type of cryptography that can be used. In particular, it is not possible to use public key cryptography in a way that is both cheap and fast enough. Additionally, most RFID tags are not tamper-resistant and do not have the ability to keep time, since they lack an independent energy source.

Protocols that have been proposed to achieve both security and privacy are, among others, OSK/AO [OSK03,AO05], NIBY [NIBY06], and YA-TRAP [Tsu06]. In the literature, RFID schemes are typically modeled as a multi-threaded reader that enjoys an always-active secure communication channel with the back office [JW07,Vau07,Avo05]. Although this approach is simple and practical, it cannot model several widely deployed RFID systems nowadays.

In practice, in a number of large-scale RFID systems, readers remain off-line most of the time and have only periodic connection with the central back office. During that connection, the readers and the back office synchronize. Typical examples of this configuration are transport ticketing systems such as the London Oyster card and the Dutch OV-chipkaart, where readers in buses and trains connect to a central database during the night and remain off-line during the day. This configuration enforces the migration of sensitive information from the back office to the readers, since readers now have to be able to decide by themselves whether to grant access to a passenger or not, to name an example. This configuration brings new security threats: readers might de-synchronize with other readers, tags or with the back office itself. Besides that, if an attacker steals or tampers with a reader that now contains sensitive information it should not compromise the security and privacy of the whole system. Concurrently and independently this issue has also been studied by Avoine et al. in [ALM09]; both results were presented at RFIDSec'09.

Our contribution. In this paper, we propose to explicitly model the existence of multiple readers, that, just as tags, can be compromised by the adversary and their secret information obtained. With respect to tag corruption, we consider a "forward" notion of privacy: if a tag is corrupted in the future, its past behavior is still private. With respect to reader corruption, we consider both a "forward" and a "backward" notion of privacy: if a reader is corrupted in the future, privacy of past communication should still be guaranteed, but also if some reader has been corrupted in the past, privacy should still be guaranteed in the future. See Figures 1 and 2.

Because readers only periodically connect to the back office, one cannot expect to retain privacy of tags (or security of the system) at the moment a reader is destroyed. More precisely, during a time slot (a period between two successive synchronizations of the whole system) in which a reader gets corrupted, security and privacy cannot be guaranteed. There is the additional point that tags do not have an intrinsic notion of time. For forward privacy, since the adversary can corrupt a reader in the future, privacy is only guaranteed if the last commu-

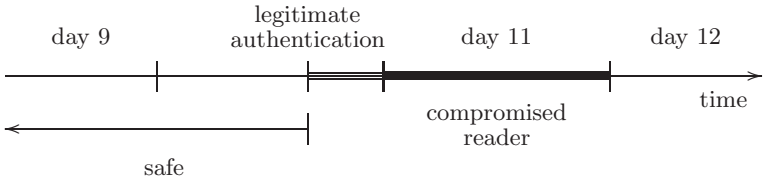

Fig. 1. Self-stabilizing Forward Privacy

nication of the tag (before the time slot where reader corruption takes place) is with a legitimate reader. When there is no reader corruption, this condition is not necessary and our notion of forward privacy reduces to the standard one.

For backward privacy, note that a corrupted reader will always be able to communicate with a tag that has not communicated with the system since the reader corruption took place. It is not until a tag communicates with a legitimate reader that one can expect to regain privacy guarantees.

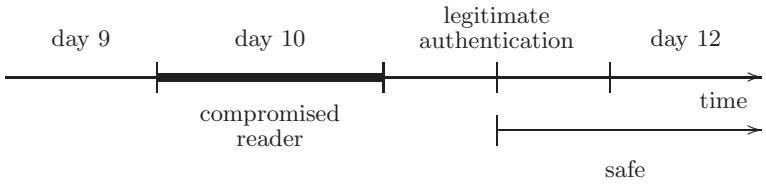

Fig. 2. Self-stabilizing Backward Privacy

We formulate three notions of privacy: forward privacy with respect to tag corruption, self-stabilizing forward privacy with respect to reader destruction, and self-stabilizing backward privacy with respect to reader destruction. "Self-stabilizing" refers to the fact that privacy is not immediately guaranteed outside the time slot where reader destruction takes place, but only after communicating with a legitimate reader.

As in [Vau07], we consider several classes of adversaries (depending on their ability to corrupt or destroy tags, destroy readers, or see the result of an authentication protocol between a tag and a reader) and the privacy (and security) notions are parameterized by the class of adversaries under consideration. We model the privacy requirements as in [JW07]. An attacker generates two uncorrupted tags. Later, he get access to only one of them and has to guess which one it is. The privacy requirement is that he only guesses correctly with probability negligibly larger than $\frac{1}{2}$. The difference between the three privacy notions is expressed in the different capabilities the adversary has. For privacy with respect to tag corruption, the adversary cannot destroy readers. For self-stabilizing forward privacy with respect to reader destruction, the adversary can only corrupt readers after being given access to the challenge tag. For self-stabilizing backward

privacy with respect to reader corruption, it can only corrupt readers before it is given access to the challenge tag.

Finally, we analyze three protocols. First, because we have slightly modified the notion of privacy from the literature combining ideas from [JW07] and [Vau07], we present a slightly modified version of the OSK protocol that obtains security and privacy with respect to tag corruption. We then describe two other protocols, using only hashing, that additionally achieve self-stabilizing forward and backward privacy with respect to reader corruption. In all three cases, we prove the security and privacy of the scheme in the random oracle model.

Structure of the paper In Section 2 we formally model an RFID system with offline readers. We model adversaries, as usual, as probabilistic polynomial-time algorithms that interact with an RFID system by means of oracles. In Section 3, we define the notions of security and privacy, using a game-based approach. Section 4 describes the three protocols achieving our security and privacy notions and the proofs in the random oracle model. Finally, Section 5 concludes.

2 System Model

To model this scenario, consider a scheme where readers have a secure communication channel with the back office that is only active during synchronization. We assume that readers are single threaded, i.e., can only have one active protocol instance with a tag at a time. After running a protocol with a tag, the reader has an output that is typically the identity of the tag. New readers and tags can be added to the system at will. The formal definition follows.

Definition 2.1 (RFID scheme)**.** An *RFID scheme* Π consists of:

- a probabilistic polynomial-time algorithm SetupSystem that takes as input the security parameter 1^η and outputs the public key pair $(\mathrm{sk}, \mathrm{pk})$ of the system.
- a probabilistic polynomial-time algorithm SetupReader that takes as input the secret key of the system sk and outputs the initial state of the reader s and the reader's secret k.
- a probabilistic polynomial-time algorithm SetupTag that takes as input the secret key of the system sk and outputs the initial state of the tag s and the tag's secret k.
- a polynomial-time interactive protocol Sync between the readers and the back-office.
- a polynomial-time interactive protocol between a reader and a tag, where the reader returns Output. Output is typically the identity of the tag.

An adversary is a probabilistic polynomial-time algorithm that interacts with the system by means of different oracles. The environment keeps track of the state of each element in the system and answers the oracle queries according to the protocol. Besides adding new tags and readers to the system and being able to communicate with them, an adversary can also corrupt tags. This models

techniques like differential power analysis and chip slicing. By corrupting a tag an adversary retrieves its internal state. Something similar happens with readers, although in this case we assume that the system can detect that. In the example of the transport ticketing system we assume that the company would detect if a bus gets stolen or a missing gate at the metro station. An adversary can also initiate the synchronization protocol which models waiting until the next time-period. Additionally, an adversary might be capable of seeing the result of an authentication attempt by external means, for instance, by looking whether a door opens or not. The formal definition of adversary follows.

Definition 2.2 (Adversary)**.** An *adversary* is a probabilistic polynomial-time algorithm that takes as input the system public key pk and has access to the following oracles:

- $\mathsf{CreateReader}(\mathcal{R})$ creates a new reader by calling $\mathsf{SetupReader}(\mathrm{sk})$ and updates the state of the back-office. This new reader is referenced as $\mathcal{R}$.
- $\mathsf{DestroyReader}(\mathcal{R})$ destroys reader $\mathcal{R}$ and returns its internal state s to the adversary. After calling $\mathsf{DestroyReader}$, oracle calls with this reference are no longer valid.
- $\mathsf{CreateTag}(\mathcal{T})$ creates a new tag $\mathcal{T}$ by calling $\mathsf{SetupTag}(\mathrm{sk})$ and updates the state of the back-office. This new tag is referenced as $\mathcal{T}$.
- $\mathsf{CorruptTag}(\mathcal{T})$ returns the internal state s of the tag $\mathcal{T}$.
- $\mathsf{Launch}(\mathcal{R})$ attempts to initiate a new protocol instance at reader $\mathcal{R}$. If $\mathcal{R}$ has already an active protocol instance then Launch fails and returns zero. Otherwise it starts a new protocol instance and returns one.
- $\mathsf{Send}(m, A)$ sends a message m to the entity A and returns its response m'. The entity A can either be a reader $\mathcal{R}$ or a tag $\mathcal{T}$.
- $\mathsf{Result}(\mathcal{R})$ outputs whether or not the output of the last finished protocol instance at reader $\mathcal{R}$ is not $\perp$, i.e., $\mathsf{Output} \neq \perp$.
- $\mathsf{Sync}()$ initiates the interactive protocol Sync between the readers and the back-office.

Definition 2.3. We denote by $\mathcal{O}$ the set of oracles $\{\mathsf{CreateReader}, \mathsf{CreateTag}, \mathsf{CorruptTag}, \mathsf{Launch}, \mathsf{Send}, \mathsf{Sync}, \mathsf{Result}\}$ and $\mathcal{O}^+ = \mathcal{O} \cup \{\mathsf{DestroyReader}\}$.

3 Security Definitions

This section elaborates on the security and privacy definitions from the literature, adapting them to our model. Then, it also discusses (when applicable) the relations among them.

The main goal of an RFID system is security, which means that readers are able to authenticate legitimate tags. Throughout this paper we focus on privacy. For the sake of self containment, we include here the following security definition which is an adapted version of the security definition proposed in [Vau07].

Definition 3.1 (Security)**.** An RFID scheme is *secure* if for all adversaries $\mathcal{A}$ and for all readers $\mathcal{R}$, the probability that $\mathcal{R}$ outputs the identity of a legitimate tag while the last finished protocol instance at reader $\mathcal{R}$ and this tag did not have any

matching conversation, is a negligible function of η. Matching conversation here means that $\mathcal{R}$ and the tag (successfully) executed the authentication protocol.

Next we define privacy with respect to tag corruption. We compose the definitions of Juels and Weis [JW07] and Vaudenay [Vau07] since each of them has its advantages: the former is indistinguishability based, which makes it more practical; the latter has the drawback of being simulation based but is stronger and allows for a variety of adversaries with custom capabilities. Privacy is defined in an IND-CCA like fashion where the adversary tries to win the privacy game. In this game, the environment creates system parameters by calling $\mathsf{SetupSystem}$. Then it gives the public key of the system pk to the adversary $\mathcal{A}_0$. This adversary has access to the set of oracles $\mathcal{O}$. Eventually, $\mathcal{A}_0$ must output two uncorrupted challenge tags $\mathcal{T}_0^\star$ and $\mathcal{T}_1^\star$. Then, the environment chooses a random bit b and gives the adversary $\mathcal{A}_1$ access to $\mathcal{T}_b^\star$. At this point, the original references to $\mathcal{T}_0^\star$ and $\mathcal{T}_1^\star$ are no longer valid. Again, the adversary has access to all oracles $\mathcal{O}$. Finally, the adversary outputs a guess bit b'. The adversary wins the game if $b = b'$. The formal definition follows.

Definition 3.2 (Privacy game)**.**

$$
\begin{array}{l}
\textbf{Priv-Game}_{\Pi,\mathcal{A}}(\eta) \ : \\
\quad (\mathrm{sk}, \mathrm{pk}) \leftarrow \mathsf{SetupSystem}(1^\eta) \\
\quad \mathcal{T}_0^\star, \mathcal{T}_1^\star \leftarrow \mathcal{A}_0^{\mathcal{O}}(\mathrm{pk}) \\
\quad b \leftarrow \{0,1\} \\
\quad b' \leftarrow \mathcal{A}_1^{\mathcal{O}}(\mathcal{T}_b^\star) \\
\quad \textbf{winif } \text{if } b = b'.
\end{array}
$$

The challenge tags $\mathcal{T}_0^\star$ and $\mathcal{T}_1^\star$ must be uncorrupted, which means that no $\mathsf{CorruptTag}(\mathcal{T}_{\{0,1\}}^\star)$ query has been made. Adversaries implicitly pass state.

In general, it is hard to define a realistic adversarial model as different applications have different requirements. Following the lines of Vaudenay [Vau07], we consider different classes of adversaries depending on their capabilities. The notions of forward, weak and narrow adversaries are due to Vaudenay. Intuitively, a *forward* adversary is an adversary that observes communication between tags and readers and later on acquires one of these tags and tries to link it with some of the past sessions, compromising its privacy. If the adversary succeeds to do so, with non-negligible probability, we say that is a *winning* adversary. A *weak* adversary is an adversary that is unable to corrupt tags. In real life scenarios it is often realistic to assume that an adversary can see the outcome of an authentication attempt. For instance, this is the case of transport ticketing systems where an adversary could observe whether the gate of the metro opens or not, for a specific tag. An adversary that is unable to do so is called *narrow*.

We introduce the notion of *reader-destructive* adversary which is a forward adversary, additionally empowered with a $\mathsf{DestroyReader}$ oracle.

Definition 3.3 (Types of adversaries)**.** An adversary who has access to all oracles $\mathcal{O}$ is called *forward*. Note that $\mathcal{A}_1$ is allowed to perform $\mathsf{CorruptTag}$ queries on $\mathcal{T}_b^\star$. An adversary is called *weak* if it does not perform any $\mathsf{CorruptTag}$ query.

An adversary is called *narrow* if it does not perform any Result query. An adversary is called *reader-destructive* if it additionally has access to a DestroyReader oracle.

Remark 3.4. Note that this notion of forward adversary is stronger than the one proposed by Vaudenay and closer to the notion of Juels and Weis.

Definition 3.5 (Privacy)**.** Let C be a class of adversaries in $\{\text{forward}, \text{weak}, \text{narrow}\}$. An RFID scheme is said to be C*-private* if for all probabilistic polynomial-time adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1) \in C$

$$\mathbb{P}[\textbf{Priv-Game}_{\Pi,\mathcal{A}}(\eta)] - 1/2$$

is a negligible function of η.

Next, we want to generalize this privacy definition to the off-line setting, where readers can be subdued. A first attempt would be to take Definition 3.5 and additionally empower the adversary with a DestroyReader oracle. Unfortunately, the resulting definition is not achievable since the following adversary wins the privacy game with probability one, regardless of the particular scheme.

$\mathcal{A}_0^{\mathcal{O}^+}(\text{pk})$:
  $\mathsf{CreateReader}(\mathcal{R})$
  $\mathsf{CreateTag}(\mathcal{T}_0^\star)$
  $\mathsf{CreateTag}(\mathcal{T}_1^\star)$
  $\mathsf{Sync}()$
  $s \leftarrow \mathsf{DestroyReader}(\mathcal{R})$
  $id \leftarrow \mathsf{Execute}(s, \mathcal{T}_0^\star)$
  return $\mathcal{T}_0^\star, \mathcal{T}_1^\star$

$\mathcal{A}_1^{\mathcal{O}^+}(\mathcal{T}_b^\star)$:
  if $id = \mathsf{Execute}(s, \mathcal{T}_b^\star)$ **then return** 0
  else return 1

where $\mathsf{Execute}(s, \mathcal{T})$ runs the authentication protocol with tag $\mathcal{T}$ using s as the internal state of the reader.

A setup with off-line readers is inherently insecure during the time-period where reader destruction takes place. The following definition captures the notion of self-stabilizing forward privacy with respect to reader destruction. Intuitively, is the same definition as before. However, before reader destruction takes place, we must guarantee that readers and tags have the same notion of time. This is achieved by having the tags communicate with a legitimate reader followed by a Sync, without the adversary interfering. This means that once the time moves forward, the privacy of past sessions is ensured, even if the adversary retrieves the internal state of a reader, see Figure 3.

Definition 3.6 (Self-stabilizing forward privacy game)**.**

$\mathbf{SS\text{-}Fwd\text{-}Priv\text{-}Game}_{\Pi,\mathcal{A}}(\eta)$:
$(\mathrm{sk}, \mathrm{pk}) \leftarrow \mathsf{SetupSystem}(1^\eta)$
$\mathcal{T}_0^\star, \mathcal{T}_1^\star \leftarrow \mathcal{A}_0^{\mathcal{O}}(\mathrm{pk})$
$b \leftarrow \{0,1\}$
$\mathcal{R}_0^\star, \mathcal{R}_1^\star \leftarrow \mathcal{A}_1^{\mathcal{O}}(\mathcal{T}_b^\star)$
$\mathsf{Execute}(\mathcal{R}_0^\star, \mathcal{T}_0^\star)$
$\mathsf{Execute}(\mathcal{R}_1^\star, \mathcal{T}_1^\star)$
$\mathsf{Sync}()$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}^+}(\mathcal{T}_b^\star)$
winif if $b = b'$.

where $\mathsf{Execute}(\mathcal{R}, \mathcal{T})$ runs the authentication protocol between the reader $\mathcal{R}$ and the tag $\mathcal{T}$. The challenge tags $\mathcal{T}_0^\star$ and $\mathcal{T}_1^\star$ must be uncorrupted, which means that no $\mathsf{CorruptTag}(\mathcal{T}_{\{0,1\}}^\star)$ query has been made. Adversaries implicitly pass state.

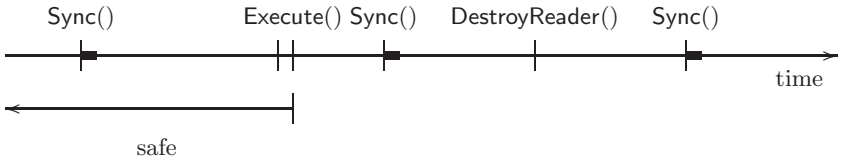

Fig. 3. Self-stabilizing Forward Privacy

Definition 3.7 (Self-stabilizing forward privacy)**.** Let C be a class of adversaries in {forward, weak, narrow, reader-destructive}. An RFID scheme is said to be C-*forward private* w.r.t. tag corruption and reader destruction if for all probabilistic polynomial-time adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2) \in C$

$$\mathbb{P}[\mathbf{SS\text{-}Fwd\text{-}Priv\text{-}Game}_{\Pi,\mathcal{A}}(\eta)] - 1/2$$

is a negligible function of η.

Theorem 3.8. *Let Π be a self-stabilizing forward private RFID system. Then, Π is private with respect to Definition 3.5.*

Proof. By inspection. Note that the games are the same up to the Execute call. The new adversary just has extra power, namely, the distinguishing capability of $\mathcal{A}_2^{\mathcal{O}^+}$, i.e., a winning adversary against **Priv-Game** is also a winning adversary against **SS-Fwd-Priv-Game**, where $\mathcal{A}_2$ just outputs the bit b chosen by $\mathcal{A}_1$. □

Next we introduce the notion of backward privacy with respect to reader destruction. Backward privacy is the time-transposed analogy of forward privacy, see Figure 4. Therefore, the same limitations on privacy during the time-period where reader destruction takes place still apply. Moreover, since tags lack a

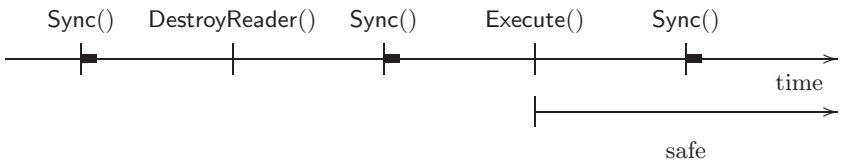

Fig. 4. Self-stabilizing Backward Privacy

timing device, the lapse extends beyond the compromised time slot. From the tag's perspective it is impossible to know that time has passed and therefore when an attacker interacts first with the tag during a later time slot, the tag is in an inherently insecure situation. In such a situation, the best one can hope for is that the tag gets back to a secure state (it self-stabilizes) after interacting with a legitimate reader. The following security definition captures this notion.

Definition 3.9 (Self-stabilizing backward privacy game)**.**

$$
\begin{array}{l}
\textbf{SS-Back-Priv-Game}_{\Pi,\mathcal{A}}(\eta) \ : \\
\quad (\mathrm{sk}, \mathrm{pk}) \leftarrow \mathsf{SetupSystem}(1^\eta) \\
\quad \mathcal{T}_0^\star, \mathcal{T}_1^\star \leftarrow \mathcal{A}_0^{\mathcal{O}^+}(\mathrm{pk}) \\
\quad \mathsf{Sync}() \\
\quad \mathcal{R}^\star \leftarrow \mathcal{A}_1^{\mathcal{O}}() \\
\quad b \leftarrow \{0,1\} \\
\quad \mathsf{Execute}(\mathcal{R}^\star, \mathcal{T}_b^\star) \\
\quad b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(\mathcal{T}_b^\star) \\
\quad \textbf{winif } \text{if } b = b'.
\end{array}
$$

The challenge tags ($\mathcal{T}_0^\star$ and $\mathcal{T}_1^\star$) must be uncorrupted, i.e., no $\mathsf{CorruptTag}(\mathcal{T}_{\{0,1\}}^\star)$ query has been made. Adversaries implicitly pass state.

Definition 3.10 (Self-stabilizing backward privacy)**.** Let C be a class of adversaries in {forward, weak, narrow, reader-destructive}. An RFID scheme is said to be *C-self-stabilizing backward private* w.r.t. tag corruption and reader destruction if for all probabilistic polynomial-time adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2) \in C$

$$\mathbb{P}[\textbf{SS-Back-Priv-Game}_{\Pi,\mathcal{A}}(\eta)] - 1/2$$

is a negligible function of η.

4 Protocol Description

In this section we first recall a slightly modified version of the OSK protocol [OSK03] and prove it narrow-forward private in the random oracle model, as a proof-of-concept. Then, we propose two new protocols that achieve self-stabilizing forward and backward privacy. Both security proves are in the random oracle model.

4.1 The OSK protocol

The modified version of the OSK protocol is depicted in Figure 5. The protocol uses two hash functions f and g. The state of the tag consists of a symmetric key k that gets hashed with every authentication attempt. The reader has a table T consisting of pairs of tag identities id and keys k. When the reader gets an answer c to a challenge n, it will search in T for any matching key, and will literately hash the keys when no match is found.

	$\mathcal{T}$ state: k		$\mathcal{R}$ state: $T = [id, k]$
0			$n \leftarrow \{0,1\}^l$
1		$\xleftarrow{\quad n \quad}$	
2	$c \leftarrow h(k, n)$		
3	$k \leftarrow g(k)$		
4		$\xrightarrow{\quad c \quad}$	
5			$\exists (id, k) \in T, i < t\colon h(g^i(k), n) = c$
6			**then** $k \leftarrow g^i(k)$; **return** $\mathcal{T}$
7			**else return** $\perp$

Fig. 5. Slightly modified version of the OSK protocol

Theorem 4.1. *The modified version of the OSK protocol depicted in Fig. 5 is narrow-forward private in the random oracle model.*

Proof. Suppose that there is an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ that wins the **Priv-Game** with non-negligible probability. Then we build the following simulator $\mathcal{S}$. $\mathcal{S}$ initializes the system and then runs the adversary $\mathcal{A}_0$ simulating all oracle calls. The random oracles $\mathcal{H}$ and $\mathcal{G}$ are simulated as usual by having a tables $T_{\mathcal{H}}$ and $T_{\mathcal{G}}$ storing previous queries and answers. Eventually $\mathcal{A}_0$ finishes and outputs tags $(\mathcal{T}_0^\star, \mathcal{T}_1^\star)$. Let k_0, k_1 be respectively the secret keys of $\mathcal{T}_0^\star$ and $\mathcal{T}_1^\star$ just before the last authentication. As in the game, $\mathcal{S}$ will draw a random bit b. Next, $\mathcal{S}$ runs $\mathcal{A}_1^{\mathcal{O}}(\mathcal{T}_b^\star)$ which eventually outputs a guess bit b'. By hypothesis we get that $b' = b$ with probability significantly higher than $1/2$. Now $\mathcal{T}$ rewinds the adversary $\mathcal{A}_0$ until it performs the first call to the random oracle $\mathcal{G}$ on input k_0 or k_1. Then it runs $\mathcal{A}_1^{\mathcal{O}}(\mathcal{T}_{1-b}^\star)$ and swaps in $T_{\mathcal{G}}$ all occurrences of k_0 and k_1. By hypothesis we get that $\mathcal{A}_1$ outputs $b' = 1-b$ with probability significantly higher than $1/2$. Since $\mathcal{A}_1$ is narrow, its view is exactly the same as in the previous run, which leads to a contradiction. □

4.2 A self-stabilizing private protocol

Figure 6 depicts our protocol for self-stabilizing forward and backward privacy. The core of the protocol is the OSK protocol plus some modifications for backward privacy. The state of the tag consists of two keys k and k'. Intuitively, the former key is used for communication with the readers and the latter is used for (indirect) communication with the back office. The state of the reader includes

a table T consisting of a tag identity id; the last-known key k; the fist-key-of-the-day $\tilde{k}$; a MAC $h(k', C_0)$ and a bit u that tells whether or not the tag has authenticated during the current time period. The MAC constitutes a proof of knowledge of k' and C_0 is a system-wide constant.

	$\mathcal{T}$ state: k, k'		$\mathcal{R}$ state: $T = [id, k, \tilde{k}, h(k', C_0), u]$
0			$n \leftarrow \{0,1\}^l$
1		$\xleftarrow{\quad n \quad}$	
2	$c \leftarrow h(k, n)$		
3	$k \leftarrow h(k)$		
4		$\xrightarrow{\quad c \quad}$	
5			$\exists\, k \in T, i \leq N : h(h^i(k), n) = c$
6			**then** $m \leftarrow h(h^{i+1}(k), h(k', C_0))$
7			$u \leftarrow 1$
8			$\exists\, \tilde{k} \in T, i \leq N : h(h^i(\tilde{k}), n) = c$
9			**then** $m \leftarrow \{0,1\}^l$
10			$\tilde{k} \leftarrow h^{i+1}(\tilde{k})$
11			**otherwise** $m \leftarrow \{0,1\}^l$
12		$\xleftarrow{\quad m \quad}$	
13	**if** $h(k, h(k', C_0)) = m$		
14	**then** $k \leftarrow h(k' + 1)$		
15	$k' \leftarrow h(k')$		

Fig. 6. Self-stabilizing Forward and Backward Private Protocol.

The $\mathsf{Sync}()$ protocol gathers the tables $T_{\mathcal{R}}$ from each reader $\mathcal{R}$. Then it computes, for each tag, the latest key k used. If there is a table $T_{\mathcal{R}}$ for which $u = 1$ then it sets $u \leftarrow 0, \tilde{k} \leftarrow h(k' + 1)$ and it updates k' in the back office by computing $k' \leftarrow h(k')$. Finally, it distributes the updated tables to all readers.

Theorem 4.2. *The protocol depicted in Fig. 6 is narrow self-stabilizing forward private in the random oracle model.*

Proof. Suppose that there is an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, A_2)$ that wins the **SS-Fwd-Priv-Game** with non-negligible probability. We build the following simulator S. S first creates system parameters by calling $(\mathrm{sk}, \mathrm{pk}) \leftarrow \mathsf{SetupSystem}(1^\eta)$. Then, it proceeds as in the **SS-Fwd-Priv-Game**, invoking $\mathcal{A}$ when specified. Again, the random oracle $\mathcal{H}$ is simulated by having a table $T_{\mathcal{H}}$ storing previous query-answer pairs. At some point $\mathcal{A}_0$ outputs challenge tags $\mathcal{T}_0^\star$ and $\mathcal{T}_1^\star$. Let $t_0^\dagger$ and $t_1^\dagger$ be respectively the time when $\mathcal{T}_0^\star$ and $\mathcal{T}_1^\star$ initiated the last successful authentication, see Figure 7. Define $k_i^\dagger, k_i'^\dagger$ as the secret keys of $\mathcal{T}_i^\star$ at time $t_i^\dagger$, for $i = 0, 1$. Let $t_i^\ddagger$ be the time of the last Sync call before $t_i^\dagger$ and let $k_i^\ddagger, k_i'^\ddagger$ be the secret keys of $\mathcal{T}_i^\star$ at time $t_i^\ddagger$, for $i = 0, 1$. Next $\mathcal{S}$ will rewind the adversary $\mathcal{A}_0$ and resume its execution with a modified random oracle $\tilde{h}$. $\tilde{h}$ is defined as h, except for the following four points: $\tilde{h}(k_i'^\ddagger) := h(k_{1-i}'^\ddagger)$ and $\tilde{h}(k_i^\dagger) := h(k_{1-i}^\dagger)$ for $i = 0, 1$. Note that this modification does not affect the

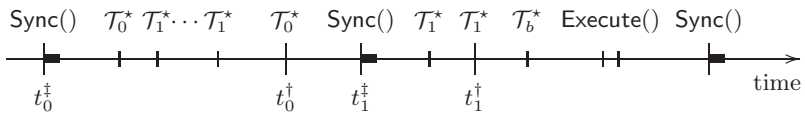

Fig. 7. Timeline of events

view of $\mathcal{A}_0$ but with negligible probability. Next, $\mathcal{S}$ calls $\mathcal{A}_1^{\mathcal{O}}(\mathcal{T}_{1-b}^\star)$. Note that $\mathcal{T}_{1-b}^\star$ has exactly the same state that $\mathcal{T}_b^\star$ had in the previous execution of $\mathcal{A}_1^{\mathcal{O}}$, therefore the view of $\mathcal{A}_1$ remains unchanged. Finally, $\mathcal{S}$ calls $\mathsf{Execute}(\mathcal{R}_0^\star, \mathcal{T}_0^\star)$, $\mathsf{Execute}(\mathcal{R}_1^\star, \mathcal{T}_1^\star)$ and $\mathsf{Sync}()$ followed by $\mathcal{A}_2^{\mathcal{O}^+}(\mathcal{T}_{1-b}^\star)$. It remains to show that $\mathsf{DestroyReader}$ calls do not change the view of the adversary. This is easy to see since the internal state of tags and readers at any time are the same, with exception of $h(k_i^\ddagger)$ and $h(k_i'^\ddagger)$. For example, the information on the readers about $\mathcal{T}_i^\star$ at time $\mathsf{max}(t_0^\ddagger, t_1^\ddagger)$ is $id_i, k = h(k_i^\dagger), \tilde{k} = h(h(k_i'^\ddagger)+1), h(k_i'^\ddagger, \tilde{d}), 0$ in one view, and $id_{1-i}, k = h(k_i^\dagger), \tilde{k} = h(h(k_i'^\ddagger)+1), h(k_i'^\ddagger, \tilde{d}), 0$ in the other. But this is not a problem since the adversary never has access to these values, due to the fact that the $\mathsf{Execute}$ and Sync calls destroy this information.

Note that since $\mathcal{A}$ is narrow, it does not have access to a Result oracle.

This time, $\mathcal{A}$ must output $1-b$ with probability significantly higher than $1/2$ but since its views are indistinguishable, this leads to a contradiction. □

Theorem 4.3. *The protocol depicted in Fig. 6 is narrow self-stabilizing backward private in the random oracle model.*

Proof. The main idea of the proof is similar to the one of Theorem 4.2. Suppose that there is an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, A_2)$ that wins the **SS-Back-Priv-Game** with non-negligible probability. As before, we build the following simulator S. S first creates system parameters by calling $(\mathrm{sk}, \mathrm{pk}) \leftarrow \mathsf{SetupSystem}(1^\eta)$. Then, it proceeds as in the **SS-Back-Priv-Game**, invoking $\mathcal{A}$ when specified. Again, the random oracle $\mathcal{H}$ is simulated by having a table $T_{\mathcal{H}}$ storing previous query-answer pairs. At some point $\mathcal{A}_0$ outputs challenge tags $\mathcal{T}_0^\star$ and $\mathcal{T}_1^\star$. Let $t_i^\dagger$ be the time when $\mathcal{T}_i^\star$ initiated the last successful authentication, for $i = 0, 1$, see Figure 8.

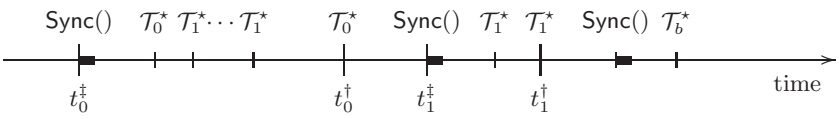

Fig. 8. Timeline of events

Let $t_i^\ddagger$ be the time of the last Sync call before $t_i^\dagger$ and let $k_i^\ddagger, k_i'^\ddagger$ be the secret keys of $\mathcal{T}_i^\star$ at time $t_i^\ddagger$, for $i = 0, 1$. Next $\mathcal{S}$ will rewind the adversary $\mathcal{A}_0$ and

resume its execution with a modified random oracle $\tilde{h}$ defined as h except for the points $\tilde{h}(k_0'^{\ddagger}) := h(k_1'^{\ddagger})$ and $\tilde{h}(k_1'^{\ddagger}) := h(k_0'^{\ddagger})$. Note that this modification does not affect the view of $\mathcal{A}_0$ but with negligible probability. Next, $\mathcal{S}$ calls $\mathsf{Sync}()$ and then $\mathcal{A}_1^{\mathcal{O}}()$, which does not have access to either $\mathcal{T}_0^{\star}$ or $\mathcal{T}_1^{\star}$. Eventually $\mathcal{A}_1$ outputs a reader $\mathcal{R}^{\star}$ and then $\mathcal{S}$ runs $\mathsf{Execute}(\mathcal{R}^{\star}, \mathcal{T}_{1-b}^{\star})$. Finally, $\mathcal{S}$ calls $\mathcal{A}_2^{\mathcal{O}}(\mathcal{T}_{1-b}^{\star})$. Note that $\mathcal{T}_{1-b}^{\star}$ has exactly the same state that $\mathcal{T}_b^{\star}$ in the previous execution of $\mathcal{A}_2$ and therefore the view of $\mathcal{A}_2$ remains unchanged. This time, $\mathcal{A}$ must output $1-b$ with probability significantly higher than $1/2$ but since its views are indistinguishable, this leads to a contradiction. □

Remark 4.4. This protocol suffers from de-synchronization when reader corruption has taken place.An adversary can use the data from a corrupted reader to move forward the key k' both in a tag or in the back office. Throughout this paper we focus on privacy and the purpose of this protocol is to show achievability of our privacy notions. In this protocol there is still a trade-off possible between de-synchronization resistance and privacy. The reader could additionally store the keys $\tilde{k}$ from adjacent time-slots. This, of course, extends the unsafe window from one to three time slots but it allows re-synchronization.

	$\mathcal{T}$ state: k, k'		$\mathcal{R}$ state: $T = [id, k, \tilde{k}, h(k', C_0), u]$
0			$n \leftarrow \{0,1\}^l$
1		$\xleftarrow{\quad n \quad}$	
2	$c \leftarrow h(k, n)$		
3	$k \leftarrow h(k)$		
4		$\xrightarrow{\quad c \quad}$	
5			$\exists\, k \in T, i \leq N : h(h^i(k), n) = c$
6			**then** $m \leftarrow h(h^{i+1}(k), h(k', C_0))$
7			flag $\leftarrow$ true
8			$\exists\, \tilde{k} \in T, i \leq N : h(h^i(\tilde{k}), n) = c$
9			**then** $m \leftarrow \{0,1\}^l$
10			$\tilde{k} \leftarrow h^{i+1}(\tilde{k})$
11			**otherwise** $m \leftarrow \{0,1\}^l$
12		$\xleftarrow{\quad m \quad}$	
13	**if** $h(k, h(k', C_0)) = m$		
14	**then** $k \leftarrow h(k'+1)$		
15	$k' \leftarrow h(k')$		
16	$c' \leftarrow h(k, n)$		
17	$k \leftarrow h(k)$		
18		$\xrightarrow{\quad c' \quad}$	
19			**if** flag **and** $h(\tilde{k}, n) = c'$
20			**then** update k' in BO $(u \leftarrow 1)$

Fig. 9. Improved Self-stabilizing Forward and Backward Private Protocol.

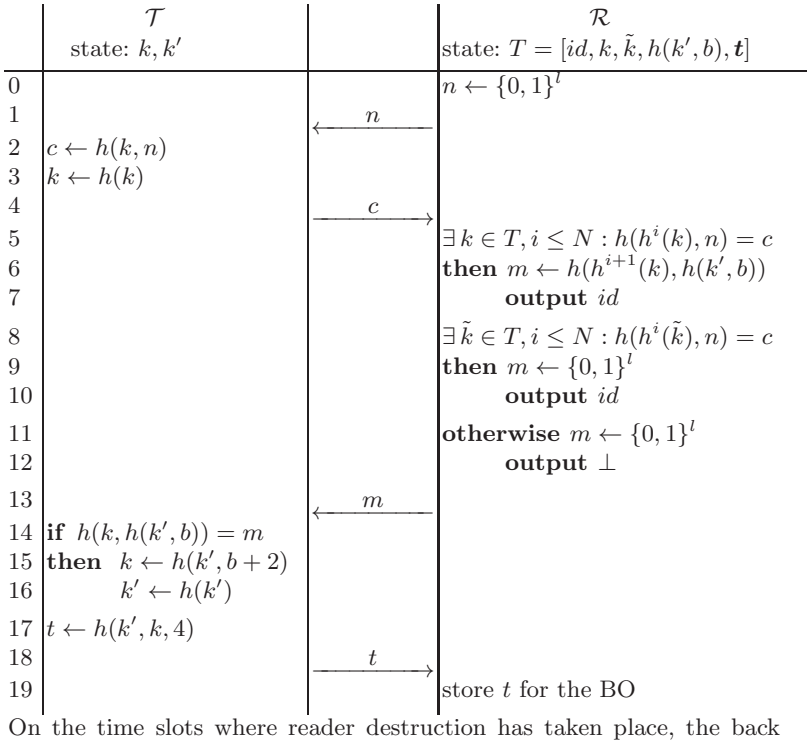

Fig. 10. Improved Self-stabilizing Forward and Backward Private Protocol.

4.3 Improving synchronization

Figure 9 depicts an improved version of the protocol from Section 4.2. By adding a second authentication it is possible to address the de-synchronization issue exposed in Remark 4.4. This allows the reader to verify whether or not the key update has been successful and only report successful key updates to the back office. Unfortunately this protocol is still vulnerable to de-synchronization when reader corruption takes place. Figure 10 shows a protocol that prevents this by storing witnesses of each authentication for later analysis at the back office.

Theorem 4.5. *The protocol depicted in Fig. 10 is narrow self-stabilizing forward and backward private in the random oracle model.*

Proof. The proof closely follows the lines of the ones of Theorems 4.2 and 4.3. Note that authenticating twice does not compromise privacy, otherwise that would be a valid attack against the protocol from Section 4.2, contradicting Theorem 4.2 or 4.3. □

5 Conclusions

We have proposed a new model for RFID privacy that considers off-line systems and the potential threat of reader subversion. We have elaborated on the privacy notions from the literature and adapted the standard notions of forward and backward security to this setting. We have shown that the straightforward generalization is unachievable. We have proposed the notions of self-stabilizing forward and backward privacy, which are the strongest one can expect to attain.

We have designed two authentication protocols that achieve self-stabilizing forward and backward privacy. We have proven the security of these protocols in the random oracle model. This protocols use only a hash function as a cryptographic primitive, which makes it suitable to be implemented using some of the many lightweight functions proposed in the literature [Sha08,PHER07].

References

ALM09. Gildas Avoine, Cédric Lauradoux, and Tania Martin. When Compromised Readers Meet RFID. In H.Y. Youm and M. Yung, editors, *Workshop on Information Security Applications – WISA'09*, volume 5932 of *Lecture Notes in Computer Science*, pages 36–50, Busan, Korea, August 2009. Springer-Verlag.

AO05. Gildas Avoine and Philippe Oechslin. RFID Traceability: A Multilayer Problem. In Andrew Patrick and Moti Yung, editors, *Financial Cryptography – FC'05*, volume 3570 of *Lecture Notes in Computer Science*, pages 125–140, Roseau, The Commonwealth Of Dominica, February–March 2005. IFCA, Springer-Verlag.

Avo05. Gildas Avoine. Adversary Model for Radio Frequency Identification. Technical Report LASEC-REPORT-2005-001, Swiss Federal Institute of Technology (EPFL), Security and Cryptography Laboratory (LASEC), Lausanne, Switzerland, September 2005.

JW07. Ari Juels and Stephen Weis. Defining Strong Privacy for RFID. In *International Conference on Pervasive Computing and Communications – PerCom 2007*, pages 342–347, New York City, New York, USA, March 2007. IEEE, IEEE Computer Society Press.

NIBY06. Yasunobu Nohara, Sozo Inoue, Kensuke Baba, and Hiroto Yasuura. Quantitative Evaluation of Unlinkable ID Matching Schemes. In *Workshop on Privacy in the Electronic Society – WPES*, pages 55–60, Alexandria, Virginia, USA, November 2006. ACM, ACM Press.

OSK03. Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic Approach to "Privacy-Friendly" Tags. In *RFID Privacy Workshop*, MIT, Massachusetts, USA, November 2003.

PHER07. P. Peris-Lopez, J.C. Hernandez-Castro, J.M. Estevez-Tapiador, and A. Ribagorda. An efficient authentication protocol for rfid systems resistant to active attacks. In *Emerging Directions in Embedded and Ubiquitous Computing*, volume 4809 of *Lecture Notes in Computer Science*, pages 781–794. Springer, 2007.

Sha08. Adi Shamir. SQUASH - A New MAC With Provable Security Properties for Highly Constrained Devices Such as RFID Tags. In *Fast Software Encryption – FSE 2008*, pages 144–157, 2008.

Tsu06. Gene Tsudik. YA-TRAP: Yet Another Trivial RFID Authentication Protocol. In *International Conference on Pervasive Computing and Communications – PerCom 2006*, Pisa, Italy, March 2006. IEEE, IEEE Computer Society Press.

Vau07. Serge Vaudenay. On Privacy Models for RFID. In *Advances in Cryptology - Asiacrypt 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87, Kuching, Malaysia, December 2007. Springer-Verlag.