This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Refresh: Weak Privacy Model for RFID Systems

Li Lu*, Yunhao Liu†, and Xiang-Yang Li‡

***Abstract*—Privacy-Preserving Authentication (PPA) is crucial for Radio Frequency Identifcation (RFID)-enabled applications. Without appropriate formal privacy models, it is difficult for existing PPA schemes to explicitly prove their privacy. Even worse, RFID systems cannot discover potential security flaws that are vulnerable to new attacking patterns. Recently, researchers propose a formal model, termed as Strong Privacy, which strictly requires tags randomly generate their output. Adopting the Strong Privacy model, PPA schemes have to employ brute-force search in tags' authentications, which incurs unacceptable overhead and delay to large-scale RFID systems. Instead of adopting Strong Privacy, most PPA schemes improve the authentication efficiency at the cost of the privacy degradation. Due to the lack of proper formal models, it cannot be theoretically proven that the degraded PPA schemes can achieve acceptable privacy in practical RFID systems. To address these issues, we propose a weak privacy model, Refresh, for designing PPA schemes with high efficiency as well as acceptable privacy. Based on Refresh, we show that many well-known PPA schemes do not provide satisfied privacy protection, even though they achieve relatively high authentication efficiency. We further propose a Light-weight privAcy-preServing authenTication scheme, LAST, which can guarantee the privacy based on the Refresh model and realize $\mathcal{O}(1)$ authentication efficiency, simultaneously.**

***Index Terms*—RFID, Privacy, Model, Authentication.**

I. Introduction

DUE to the low cost and easy deployment, Radio Frequency Identification (RFID) becomes a promising technology in everyday's applications, such as logistics, access control [11], and supply chain management systems [16], [12]. In RFID systems, the reader keeps interrogating the nearby tags via RF waves. The RFID tags, which are usually attached to people or objects, reply the queries with their IDs or other stored information. The interaction pattern, however, raises a security concern to RFID systems. Within the interrogating range, any reader can retrieve the information, sometimes sensitive and private data, from the tags. A malicious reader thereby is able to access the secret information stored in the tags, such as the personal information of customers, the items they purchased, the customers' health condition inferred by the purchased pharmaceutical, and etc. Moreover, the tags can be located via their emitted information even semantic meaning has been encoded, so that they are subject to the tracking attack. All above drawbacks could be exploited by an adversary to violate users' privacy significantly.

To address the problems, Privacy-Preserving Authentication (PPA) becomes an immediate need for protecting the interactive procedure between the RFID reader and tags [19]. Recently, many PPA schemes have been proposed [8], [6], [22], [17], [2], [14], [15], [20], [13], [5], [4]. Nevertheless, the research on RFID is still short of appropriate formal models that can explicitly define the privacy whereas maintain the authentication efficiency in a general way. Lacking such models, existing PPA schemes have to employ ad hoc notions of security and privacy [9], and then heuristically analyze the security and privacy via those notions. The heuristic analysis, however, only allows those PPA schemes examine the privacy under the known attacks using the ad hoc defined notations. It is difficult to explore the potential vulnerabilities and flaws that are vulnerable to newly emerging attacks. To our best knowledge, most existing works, if not all, suffers one or several attacks.

Juels proposes a privacy model [10], named "Strong Privacy", to meet the demands on privacy in RFID systems. Strong Privacy employs *indistinguishability* to represent the privacy. Indistinguishability means that RFID tags should not be distinguished from each other according to their output. To achieve indistinguishability, tags randomize their output such that adversaries cannot compromise their privacy. However, to authenticate a tag, the legitimate reader cannot be directly aware of which tag it is interrogating due to the random output of the tags. It has to search all tags in the system to identify the tag instead. Under the Strong Privacy model, such a brute-force search makes the authentication efficiency linear to the number of tags in the system. In other words, the PPA scheme might not be practical and applicable in large-scale RFID systems due to the linear-search authentication, although the Strong Privacy model can guarantee the privacy.

To meet the large-scale deployments, most PPA schemes focus on high authentication efficiency. Their privacy, however, surely degrades because of the trade-off between the privacy and authentication efficiency. They are often vague about that how much cost on the privacy degradation brings back how much improvement on the authentication efficiency.

In this paper, we propose a weak formal privacy model, termed as Refresh. Different from Strong Privacy, Refresh loosens the strict constrains on the output of tags, such as the randomization and unpredictability. Refresh allows a tag to contain a temporally constant field in the output for stating the tag's identity. The field remains unchanged during the time interval between two consecutive interrogations from the legitimate reader. Thus, the field is predictable for the valid reader and used to accelerate authenticating the tag. After a successful authentication with the valid reader, this field will be refreshed (or updated) for next authentication procedure. The tag encrypts the field using the secret keys shared with the valid reader for preventing adversaries from identifying the tag. By this means, RFID systems can achieve acceptable privacy protection as well as highly efficient authentica-

*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Email: luli2009@uestc.edu.cn
†Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, P.R. China. Email: liu@cse.ust.hk
‡Dept. of Computer Science, TongJi University, China, and Dept. of Computer Science, Illinois Institute of Technology, Chicago, IL, USA. Email: xli@cs.iit.edu

978-1-4244-5837-0/10/$26.00 ©2010 IEEE

tion. Based on Refresh, we propose a Light-weight privAcy-preServing authenTication scheme, LAST which showcases our exploration that a little privacy degradation can bring significant benefit on the authentication efficiency. We highlight our contributions as follows.

- We propose a new model, Refresh, which sets forth the concept "Refresh Privacy" for PPA schemes. This concept closes the gap between security model and practical PPA schemes.
- Based on Refresh, we present a methodology of formally analyzing privacy. With this methodology, we examine well-known PPA schemes and discover their potential security flaws.
- We also present a demonstrative PPA scheme LAST based on Refresh. The proposed scheme achieves privacy and is highly efficient with an $\mathcal{O}(1)$ authentication complexity.

The rest of this paper is organized as follows. We review the related work in Section II. We discuss the Strong Privacy model in Section III and propose Refresh in Section IV. We examine the privacy of several existing well-known PPA schemes based on Refresh in Section V. In Section VI, we present a new PPA scheme, named LAST, to satisfy Refresh. We analyze the storage and authentication efficiency of LAST, and then prove that LAST achieves privacy. We conclude this work in Section VII.

II. Related Work

There are two categories of formal privacy models in RFID systems, non-cryptography based and cryptography based. The work in the non-cryptography based category is for low-cost tags that cannot perform some cryptographic operations, including symmetric-key encryption, pseudo-random number generation and hash computations. Without cryptographic operations, tags cannot share any secret with the legitimate reader. The knowledge about tags known by adversaries thereby is similar to that known by the legitimate reader. Therefore, the capability of adversaries is not less than that of the legitimate reader. Loosely speaking, adversaries can do anything that the legitimate reader can do. Two representative models, "*Minimalist*" [8] and "*Universal Re-Encryption*" [6], of this kind restrict adversaries' capabilities. Minimalist sets bounds on the number of queries that an adversary can make when mounting a man-in-the-middle attack [10]. Universal Re-Encryption excludes certain forms of active attacks, such as the cloning and replay attack. Based on those non-cryptographic formal models, PPA schemes cannot defend against some attacks in practice such as the man-in-the-middle attack and cloning attack.

To enhance the security, the cryptography based work assumes: (1) an RFID tag is able to perform some cryptographic operations, such as symmetric-key encryption, hash computation, and pseudo-random number generation; (2) there are a number of distinct secret keys shared between each tag and the valid reader. It is believed that the next generation of RFID tags will be able to carry out those operations [10]. There are two representative and similar models in this category, "*Untraceability*" [1] and "*Strong Privacy*" [10]. We choose Strong Privacy to introduce this kind of models. Strong Privacy employs *indistinguishability* to represent the privacy. That is, an adversary cannot distinguish two tags from each other in a given system according to their output. Hence, the adversary could not compromise the privacy of the two tags. To achieve indistinguishability, Strong Privacy requires tags to randomize their output. Moreover, the tags' output should also be unpredictable for the adversary as well as the legitimate reader. By realizing Strong Privacy, RFID systems can achieve a strong privacy protection. The adversary cannot gain any useful information about tags according to the output. To authenticate a tag, however, the valid reader has to search all tags in the given system to determine which tag the reader is interrogating. Clearly, the authentication efficiency is linear to the number of tags in the system. In this way, satisfying Strong Privacy means a PPA scheme is not scalable in large-scale systems, due to the linear search complexity. For example, HashLock [22] is proven to fulfill Strong Privacy, its authentication efficiency, however, is $\mathcal{O}(n)$ (without loss of generality, we use n to denote the number of tags in a given system in this paper). Other schemes, although having higher efficiency than HashLock, are proven not to be private under Strong Privacy [10]. Basically, we find that there is an instinctive trade-off between the authentication efficiency and the privacy for RFID PPA schemes.

III. Strong Privacy Model

In this section, we discuss the Strong Privacy model [10], in which the privacy of RFID systems is formally defined as *indistinguishability* among tags. Strong Privacy integrates three components: RFID Scheme, Adversaries, and Privacy Game. The RFID Scheme component depicts the different behaviors of different parties in RFID systems. The Adversaries component characterizes the capability of adversaries. The Privacy Game component describes the stratagem of adversaries to attack the RFID Scheme. We briefly overview these components and then summarize the Strong Privacy model to educe our Refresh model in the next section.

A. Overview

Generally, an RFID system consists of two types of devices, tags and a reader. A tag $\mathcal{T}$ is a transponder with well-designed circuits that can respond the signals emitted from the RFID reader with its own unique ID. Since there is no battery available on a tag, the communication range is very limited, usually within a meter [18]. Although an RFID tag has constrained capability of computations and storage, it is believed that a tag of next generation is able to perform some cryptographic operations, such as symmetric-key encryption, hash computation, and pseudo-random number generation [21], [22]. Strong Privacy also assumed that a tag $\mathcal{T}$ shares a number of secret keys with the reader. In addition, $\mathcal{T}$ cannot defend against tag-compromising [10].

The RFID reader $\mathcal{R}$ in an RFID system has one or more transceivers. The reader interrogates a tag periodically or on

a user's demand, and then reports responses of the tag to the back-end database of the system. The goal of the RFID system is to distinguish legitimate tags (tags which are registered in the back-end database) from unknown tags, and further authenticate them (infer their IDs). RFID system usually uses the back-end database to perform computations and store the information of tags in the system. Generally, the channel between the reader and back-end database is assumed to be secure. Hereafter in this paper, we take the reader device and the back-end database as a whole. Thus, we denote the reader device and back-end database by the "reader" for simplicity.

In an RFID scheme, $\mathcal{T}$ and $\mathcal{R}$ are initiated with a shared secret by the system. There is an interactive protocol describing the manner of message exchanges between $\mathcal{T}$ and $\mathcal{R}$.

Adversaries in RFID systems can be specified by three characteristics: the actions that they can perform (i.e., the *oracles* they can query), the goal of their actions (i.e., the *game* they can play), and the manner in which they can access the system (i.e., the *rules* of the game).

The Privacy Game of Strong Privacy, which describes the methodology of an adversary used to attack an RFID scheme, is derived from the classic INDistinguishability under Chosen-Plaintext Attack (IND-CPA) and under Chosen-Ciphertext Attack (IND-CCA) cryptosystem security games. The idea is that an RFID protocol can be considered private for some parameters if no polynomial-time adversary can win the game with a overwhelming probability. The goal of the adversary $\mathcal{A}$ in the game is to distinguish between two different tags within his computational bounds (i.e., how many times he can query oracles). Due to the page limitation, we do not go through the details of the game, which can be found in Juels's work [10].

The basic idea of Strong Privacy is that tags randomize their output such that adversaries cannot compromise their privacy. However, to authenticate a tag, the legitimate reader has to search all tags in the system to authenticate the tag. Such a brute-force search makes the authentication efficiency linear to the number of tags in the system.

B. Issues of Strong Privacy

In essence, the Strong Privacy model can achieve a strong sense of privacy with an implicate assumption that both adversaries and the legitimate reader perform algorithms in polynomial time. It is well known that the polynomial time algorithms are considered as being efficient in theory, but not really applicable to practice applications in many protocols due to large hidden constant, or large polynomial degree. For example, to preserve privacy, the output of a tag are encrypted or hashed with a key shared between the reader and the tag. Without the key, an adversary cannot determine which tag is interrogated by the reader. On the other hand, the reader has to search all tags' keys to determine which tag can generate the similar output such that it can authenticate this tag. Obviously, authenticating the tag is a brute-force search with the efficiency of authentication being $\mathcal{O}(n)$. For a large scale RFID system, the authentication efficiency $\mathcal{O}(n)$ may not be acceptable.

Existing PPAs can be classified into two categories: (1) the variants of HashLock, and (2) tree-based approaches.

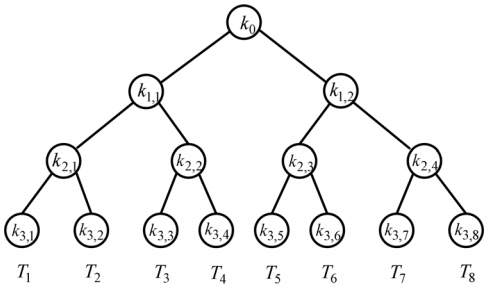

Fig. 1. A binary key tree with eight tags

HashLock is proven to be private under Strong Privacy, but its efficiency of authentication is $\mathcal{O}(n)$ [10]. Therefore, it is not practical for a large scale system. Although HashLock's variants [17], [2], [14], [20], [4] employ some techniques to improve the identification efficiency, it has been proven that those schemes are not private under the Strong Privacy model. For example, they may suffer the de-synchronization attack [10]. Furthermore, their authentication complexity is still not less than $\mathcal{O}(n^{2/3})$ [2], which is not very piratical.

In tree-based approaches [15], [14], [5], [13], each tag's keys are correlated to those of other tags. These approaches usually employ a virtual balanced tree to organize keys. Each node in the tree is allocated a random key with each leaf node assigned to a tag. Hence, there is a unique path from the root to each leaf node in the tree, and the keys along this path will be distributed to the tag corresponding to the leaf node. When being interrogated, a tag will response a message containing a sequence of encrypted random numbers with its keys. At each level in the key tree, the reader performs a search and locates a node holding a key that can generate the same encrypted message as the one in the tag's response sequence. For example, at the root level, the reader can determine whether a given tag is belonging to the left subtree or the right subtree (we assume that the key tree is a binary tree in this example). Repeating this binary search level by level until a leaf node is reached, the reader can identify and authenticate the tag.

Thus, tree-based approaches can improve the key search efficiency from linear complexity (HashLock and its variants) to logarithmic complexity. However, they are proven not to be private under the Strong Privacy model. The reason is that each tag's keys can be correlated to other tags' keys. If an adversary compromises a tag, the keys stored in the tag is exposed to the adversary. Indeed, part of those keys are still used by other tags. As the case depicted in Fig. 1, each pair of tags shares some inner nodes more or less in the tree. For example, $\mathcal{T}_1$ and $\mathcal{T}_2$ share $k_{2,1}$, while $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$, and $\mathcal{T}_4$ share $k_{1,1}$. If the adversary compromises some tags, it obtains several paths from the root to the leaf nodes corresponding to the compromised tags, as well as the keys at those paths. As a result, the adversary captures the keys still used by uncompromised tags. With these keys, the adversary can identify uncompromised tags by eavesdropping on the output of these tags. The detail analysis of the compromising attack can be found in [13].

The above discussion reflects the tradeoff between the

authentication efficiency and privacy of RFID systems. Generally speaking, more secure privacy requires more complicated authentication algorithms. Thus, we attempt to improve the identification efficiency by slightly degrading the privacy protection of tags, which is still acceptable in real RFID applications. To this end, we propose a weak privacy model for explicitly defining the privacy.

IV. Refresh Model

In this section, we propose a new model, named *Refresh*, for RFID systems. In this paper, we consider RFID systems with only one reader, which is the case in most real application scenarios.

The basic idea of Refresh is that a tag generates independent output each time when being interrogated by the legitimate reader, making adversaries very hard, if not impossible, to correlate the current output with previous ones. Refresh consists of three components: RFID Scheme, Adversaries, and Privacy Game.

A. RFID Scheme

In Refresh, an RFID scheme is defined as following:

Definition 1 (RFID Scheme): An RFID scheme consists of

- a polynomial-time algorithm $KeyGen(1^s)$ which generates all key materials $k_1, \ldots, k_n$ for the system depending on a security parameter s.
- a setup scheme $SetupTag(ID)$ which allocates a specific secret key k and a distinct ID to a tag. Each legitimate tag should have a pair (ID, k) stored in the back-end database.
- a setup scheme $SetupReader$ which stores all pairs (ID, k) in the reader's back-end database for all legitimate tags ID in the system.
- a polynomial-time interactive protocol P between the reader and a tag in which the reader owns the common inputs, the database and the secrets. If the reader fails, it outputs $\perp$; otherwise, outputs some ID and may update the database.

An RFID scheme has a correct output if the reader executes the protocol P honestly and then infers the ID of a legitimate tag except with a negligible probability (A function in terms of a security parameter s is called negligible if there exists a constant $x > 0$ such that it is $\mathcal{O}(x^{-s})$). Otherwise, the reader outputs $\perp$ when the tag is not legitimate.

B. Adversaries

Similar to Strong Privacy, adversaries in Refresh have three characteristics: the oracles they can query, the goal of their actions, and the rules of their actions. According to those characteristics, we define adversaries in RFID systems below.

Definition 2 (Adversaries): An adversary $\mathcal{A}$ in an RFID system is a polynomial-time algorithm which performs attacking behaviors by querying five oracles.

- $Launch \rightarrow \pi$: Execute a new protocol instance π between the reader and a tag.
- $TagQuery(m, \pi, \mathcal{T}) \rightarrow m'$: Send a message m to a given protocol session π on the tag $\mathcal{T}$. The oracle returns a message m' as the output of $\mathcal{T}$.
- $ReaderSend(m, \pi, \mathcal{R}) \rightarrow m'$: Send a message m to a given protocol session π on the reader $\mathcal{R}$. The oracle returns a message m' as the output of $\mathcal{R}$.
- $Corrupt(\mathcal{T})$: Compromise the tag $\mathcal{T}$, and obtain the secret stored in $\mathcal{T}$. The tag $\mathcal{T}$ is no longer used after this oracle call. In this case, we say that the tag $\mathcal{T}$ is destroyed.
- $Result(\pi)$: When the π is complete, the oracle returns 1 if the scheme has the correct output; otherwise, it returns 0.

The adversary starts a game by setting up the RFID system and feeding the adversary with the common parameters. The adversary uses the oracles above following a privacy game, which will be described in next subsection, and produces the output. Depending on the output, the adversary wins or loses the game.

C. Privacy Game

We introduce our "Privacy Game" in terms of the classic indistinguishability under chosen-plaintext attack (IND-CPA) cryptosystem security experiment. In IND-CPA experiment, a cryptosystem is considered to be secure if no adversary can distinguish the ciphertexts of two known plaintexts. In RFID system, due to the challenge-response communication model between a reader and tags, the output of tags can be considered as ciphertexts of challenge messages from the reader. Therefore, similar to IND-CPA, an RFID protocol may be considered private for some security parameters if no adversary has a noneligible advantage in this privacy 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.

The game experiences three phases: Learning, Challenging and Re-learning.

As shown in Fig. 2, in the Learning phase, $\mathcal{A}$ is able to issue any message and perform any polynomial-time computation (i.e., query oracles in polynomial times). After the Learning phase, $\mathcal{A}$ selects two uncorrupted tags as challenge candidates in the Challenge phase. One of these challenge candidates is then randomly chosen by the system (the *Challenger* $\mathcal{C}$) and presented to the adversary (the oracles of this tag can be queried by the adversary except the *Corrupt* oracle). After that, similar to the Learning phase, $\mathcal{A}$ is offered the oracles of all the tags in the RFID system by $\mathcal{C}$ except the two challenge candidates. This phase is named Re-learning. At the end of Re-learning, $\mathcal{A}$ outputs a guess about which candidate tag is selected by $\mathcal{C}$. If the guess is correct, $\mathcal{A}$ wins the game; otherwise, $\mathcal{A}$ loses.

In addition, there are two requirements for our privacy game to work properly. First, at the step (7) in the privacy game, the challenger $\mathcal{C}$ refreshes the private information of the two challenge candidates $\mathcal{T}_0^*$ and $\mathcal{T}_1^*$. Thus, the adversary cannot correlate the output of $\mathcal{T}_b^*$ at the Re-learning phase with the

$\mathbf{Game}_{\mathcal{A}}^{ref-priv}(s,n,r,t,c)$:
Setup:
(1) $KeyGen(1^s) \rightarrow (k_0,\ldots,k_n)$.
(2) Initiate $\mathcal{R}$ by $SetupReader$ with $(k_0,\ldots,k_n)$.
(3) Initiate each tag $\mathcal{T}_i$ by $SetupTag(\mathcal{T}_i,k_i)$ with the key k_i.
Learning:
(4) $\mathcal{A}$ may perform the following actions in any interleaved order.
(a) Query $ReaderSend$ oracle at most r times.
(b) Query $TagQuery$ oracle at most t times.
(c) Query $Corrupt$ oracles of $n-2$ tags arbitrarily selected from n tags.
(d) Do computations without exceeding c overall steps. Note that $\mathcal{A}$ may query $Launch$ oracle to initiate an instance of the protocol in communications and computations.
Challenge:
(5) $\mathcal{A}$ selects two tags $\mathcal{T}_i$ and $\mathcal{T}_j$ to which he did not query $Corrupt$ oracles, then $\mathcal{A}$ presents these two tags to challenger $\mathcal{C}$.
(6) Let $\mathcal{T}_0^* = \mathcal{T}_i$ and $\mathcal{T}_1^* = \mathcal{T}_j$.
(7) *Challenger $\mathcal{C}$ queries $TagQuery$ oracles of $\mathcal{T}_0^*$ and $\mathcal{T}_1^*$ for one time, respectively.*
(8) Challenger $\mathcal{C}$ picks up a bit b from $\{0,1\}$ uniformly at random. Then provide $\mathcal{A}$ access to $\mathcal{T}_b^*$.
Re-learning:
(9) $\mathcal{A}$ may perform the following actions in any interleaved order.
(a) Query $ReaderSend$ oracle without exceeding r overall queries.
(b) Query $TagQuery$ oracle without exceeding t overall queries.
(c) Query $Corrupt$ oracles of any tag except $\mathcal{T}_b^*$.
(d) Do computations without exceeding c overall steps. Note that $\mathcal{A}$ may query $Launch$ oracle to initiate an instance of the protocol in communications and computations.
(10) $\mathcal{A}$ outputs a guess bit b'.
(11) $\mathcal{A}$ wins the game if $b' = b$.

Fig. 2. Privacy game of Refresh

output of $\mathcal{T}_0^*$ and $\mathcal{T}_1^*$ at the Learning phase. Second, if an adversary can corrupt $n-1$ tags and get the keys of these tags, then he can retrieve the output of these corrupted tags. Therefore, any tag in the system can be definitely distinguished from others with the output of the tag. That is why at least two tags need to be uncorrupted.

We denote such a privacy game for an RFID system as $\mathbf{Game}_{\mathcal{A}}^{ref-priv}(s,n,r,t,c)$. Here s is a security parameter, for example, the length of keys, and n, r, t and c are respective parameters for number of tags, number of $ReaderSend$ queries, number of $TagQuery$ queries, and computation steps. An adversary $\mathcal{A}$ with parameters r, t, and c is denoted by $\mathcal{A}[r,t,c]$.

Based on above privacy game, we define the Refresh Privacy of an RFID scheme in Def. 3.

Definition 3 (RFID $(r,t,c)-Refresh-Privacy$): A protocol P of an RFID system achieves $(r,t,c)-Refresh-Privacy$ with parameter s, if for any polynomial-time $\mathcal{A}$, the probability of $\mathcal{A}$ wining under $\mathbf{Game}_{\mathcal{A}}^{ref-priv}(s,n,r,t,c)$ satisfies:

$$\forall \mathcal{A}(r,t,c), \mathbf{Pr}[\mathcal{A} \quad wins] \leq \frac{1}{2} + 1/ploy(s)$$

where $poly(s)$ denotes any polynomial of parameter s.

For a given protocol P in an RFID system, we define the *advantage* of an adversary by:

$$\mathbf{Adv}_P(\mathcal{A}) = \mathbf{Pr}[\mathcal{A} \quad wins] - \frac{1}{2}$$

In $\mathbf{Game}_{\mathcal{A}}^{ref-priv}(s,n,r,t,c)$, the adversary $\mathcal{A}$ can win the game in a trivial way. That is $\mathcal{A}$ picks up a bit b' from $\{0,1\}$ uniformly at random, i.e., $\mathbf{Pr}[b'=b] = \frac{1}{2}$. In this case, $\mathcal{A}$ attacks the system without any knowledge about the tags in the system, the successful attacking probability is the lower bound of all attacking activities. Therefore, we define the advantage of any polynomial-time adversary by $\mathbf{Pr}[b'=b] - \frac{1}{2}$.

Different with the Strong Privacy, Refresh Privacy only requires that tags change their output after being interrogated by the valid reader. Thus, between authentication sessions with the legitimate reader, the output value of a tag is static. Tags are therefore subject to tracking during such intervals of time. On the other hand, tags' output are generated by secret keys shared with the legitimate reader. After a valid interrogation, the secret keys are refreshed. The output of tags therefore is changed accordingly. Without the secret keys, the adversary cannot predict the output of tags, and thus fail to track tags. The privacy degradation is acceptable in many applications, which requires that the valid reader interrogates tags frequently, such as access control and retailing. With limited privacy degradation, the efficiency of authentication can be improved significantly, since the legitimate reader need not perform brute-force search to authenticate each tag.

D. Implications in Refresh

We have a set of important implications of the Refresh model.

First, the $KeyGen$ function should not generate low-entropy or strongly correlated keys. Otherwise, a PPA scheme might suffer the *compromising* attack. As the adversary $\mathcal{A}$ can make $Corrupt$ calls to some tags, $\mathcal{A}$ may infer some information of keys in uncorrupted tags from those corrupted tags. It may gain a significant advantage to win the Privacy Game. The detail of the compromising attack can be found in [13].

Second, whether or not $\mathcal{R}$ accepts a certain tag should not be history-dependent; otherwise, a PPA scheme will be vulnerable to the *de-synchronization* attack. For example, we assume that a tag only be interrogated for a maximum m times. After the adversary $\mathcal{A}$ makes m legitimate queries to a certain tag in the Learning phase, $\mathcal{A}$ chooses this tag as one of the two candidates in the challenge phase. Then $\mathcal{A}$ observes whether or not $\mathcal{R}$ accepts the challenge tag $\mathcal{T}_b^*$ in the Challenge phase. Therefore, $\mathcal{A}$ can determine the challenge tag is the selected one or not. Hence, $\mathcal{A}$ can definitely win $\mathbf{Game}_{\mathcal{A}}^{ref-priv}(s,n,r,t,c)$.

Third, the reader $\mathcal{R}$ should not differ significantly in its acceptance rate across tags; otherwise, $\mathcal{A}$ can just pick up one tag which $\mathcal{R}$ accepts frequently and one that $\mathcal{R}$ accepts less frequently as its challenge candidates in the Challenge phase. Thus, $\mathcal{A}$ might have a significant advantage in $\mathbf{Game}_{\mathcal{A}}^{ref-priv}(s,n,r,t,c)$. This attack requires that $\mathcal{R}$ must treat all tags in the system equally.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

V. Case Study

In this section, we use our refresh privacy to examine several representative RFID schemes: (1) the OSK scheme [17] and its variant AO [2]; (2) the YA-TRAP scheme [20]; (3) and tree-based approaches [15], [14], [13], [5]. We choose these schemes based on following observations: first, they are well-known and efficient; second, they lack of formal privacy proofs. We show their vulnerabilities and present the methodology of the formal analysis on PPA schemes under Refresh.

A. OSK and AO Schemes

In OSK, a tag $\mathcal{T}_i$ is initiated by the reader $\mathcal{R}$ with a secret key k_i. Let f_1 and f_2 be two independent pseudo-random functions. The tag $\mathcal{T}_i$ outputs $M_{i,t} = f_2(f_1^{(t)}(k_i))$ when interrogated by the reader $\mathcal{R}$, where $f_1^{(t)}$ denotes the functional powers of f_1 for natural t. Then $\mathcal{T}_i$ updates its key k_i as $f_1(k_i)$.

Upon receiving $M_{i,t}$, $\mathcal{R}$ performs a brute-force search to authenticate $\mathcal{T}_i$. Note that there is a predefined upper bound on the number of interrogations for each tag, denoted as m. After being accessed for m times, a tag will either yield no output or just output random nonce until the reader $\mathcal{R}$ re-initializes the tag, i.e., $1 \leq t \leq m$. To improve the efficiency of authentication, the reader $\mathcal{R}$ stores all pre-computed output in a giant table $T = \{M_{i,t} | 1 \leq i \leq n, 1 \leq t \leq m\}$. By looking up $M_{i,t}$ in the table, the reader $\mathcal{R}$ can immediately authenticate the tag $\mathcal{T}_i$.

Avoine et al propose a variant scheme AO to improve the storage efficiency of OSK by using Hellman tables [7]. The storage efficiency can be improved from $\mathcal{O}(N)$ to $\mathcal{O}(N^{2/3})$, where $N = n \times m$.

The vulnerability of OSK or AO lies in the upper bound m. An adversary can launch the de-synchronization attack on a tag. Interrogating the tag for m times, the adversary can exhaust the valid output of the tag. As a result, the valid reader will reject the tag even though the tag is valid. Algorithm 1 shows the attack procedure. Based on the de-synchronization attack, the adversary can always distinguish a tag from others and win the privacy game with the probability one. Hence, OSK and AO schemes do not achieve Refresh Privacy.

Algorithm 1 De-synchronization attack to OSK/AO

1: In the Learning phase, the adversary $\mathcal{A}$ selects two distinct tags $\mathcal{T}_i$ and $\mathcal{T}_j$ uniformly at random.
2: $\mathcal{A}$ queries the oracle $TagQuery$ of $\mathcal{T}_i$ for m times.
3: $\mathcal{A}$ submits $\mathcal{T}_i$ and $\mathcal{T}_j$ as its challenge candidates.
4: In the Re-learning phase, $\mathcal{A}$ first queries the oracle $TagQuery$ of $\mathcal{T}_b^*$ and then relays the response to $\mathcal{T}_b^*$'s $ReaderSend$ oracle, $\mathcal{A}$ finally queries $Result$ oracle.
5: If $\mathcal{T}_b^*$ is valid, the $Result$ oracle of $\mathcal{T}_b^*$ returns 1, then $\mathcal{A}$ guesses $b = 1$, i.e., $\mathcal{T}_b^* = \mathcal{T}_i$; otherwise, $\mathcal{A}$ guesses $b = 0$, i.e., $\mathcal{T}_b^* = \mathcal{T}_j$.

B. YA-TRAP Scheme

In YA-TRAP, besides the secret key k_i issued by the reader $\mathcal{R}$, a tag $\mathcal{T}_i$ stores an internal timestamp t_i to record the time of last interrogation of $\mathcal{R}$.

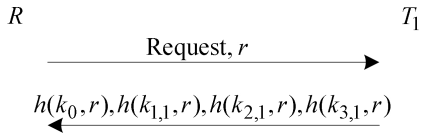

Fig. 3. A basic authentication procedure of tree-based approaches

During each interrogating, $\mathcal{R}$ sends the current timestamp t to a tag $\mathcal{T}_i$. Upon receiving t, $\mathcal{T}_i$ compares t with its internal timestamp t_i. If $t > t_i$, then $\mathcal{T}_i$ outputs $M = h(t||k_i)$ and sets $t_i = t$, where h is a cryptographic hash function and '$||$' denotes concatenation; otherwise, $\mathcal{T}_i$ outputs a random response.

To authenticate the tag, $\mathcal{R}$ conducts the brute-force search in its database to find a secret key k_i that can generate a message $h(t||k_i)$ equal to M. If such a key exists, $\mathcal{R}$ accepts the tag; otherwise, $\mathcal{R}$ rejects the tag.

Similar to OSK/AO, YA-TRAP is also vulnerable to the de-synchronization attack. An adversary can query a tag with a certain future timestamp t_{max} that indicates an extremely distant future time. The tag thereby sets its internal timestamp as t_{max}, and outputs random responses in all subsequential interrogations. Consequently, this tag will always be rejected by the legitimate reader in the following sessions. We show the attack in Alg. 2. In this scenario, the adversary can win the Privacy Game with probability one. Thus, YA-TRAP is not private under Refresh.

We can see that OSK/AO and YA-TRAP are not private under Refresh because they violate the second and third implications described in Subsection IV-D. Hence, an adversary can mark a target tag by modifying its inner state (i.e., the upper bound m in OSK/AO, and the timestamp in YA-TRAP) to win the Privacy Game.

Algorithm 2 De-synchronization attack on YA-TRAP

1: In the Learning phase, the adversary $\mathcal{A}$ selects two distinct tags $\mathcal{T}_i$ and $\mathcal{T}_j$ uniformly at random.
2: $\mathcal{A}$ queries the oracle $TagQuery$ of $\mathcal{T}_i$ with t_{max}.
3: $\mathcal{A}$ submits $\mathcal{T}_i$ and $\mathcal{T}_j$ as its challenge candidates.
4: In the Re-learning phase, $\mathcal{A}$ first queries the oracle $TagQuery$ of $\mathcal{T}_b^*$, and then relays the response to $\mathcal{T}_b^*$'s $ReaderSend$ oracle, finally makes a query to $Result$ oracle.
5: If $\mathcal{T}_b^*$ is valid, the $Result$ oracle of $\mathcal{T}_b^*$ returns 1, then $\mathcal{A}$ guesses $b = 1$, i.e., $\mathcal{T}_b^* = \mathcal{T}_i$; otherwise $\mathcal{A}$ guesses $b = 0$, i.e., $\mathcal{T}_b^* = \mathcal{T}_j$.

C. Tree-based Approaches

In tree-based approaches [15], [14], [5], [13], PPA schemes use a balanced tree to organize and store the keys for all tags.

In the key tree structure, we find that each tag shares some inner nodes, more or less, with other tags in the key tree as we discussed in Subsection III-B. Therefore, the tree-based approaches are vulnerable to compromising. If the adversary deliberately corrupts some tags, it can obtain several paths related to the corrupt tags, as well as the keys along those paths. Since those keys are shared among tags, some secret keys of uncorrupted tags will be exposed to the adversary. Only via eavesdropping, the adversary may have significant advantages of distinguishing other uncorrupted tags. Lu et al

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

[13] shows that in a tree-based RFID system containing 2^{20} tags, an adversary can identify any tag with the probability approximate to 90% by tampering with only 20 tags. We define the compromising attack in Alg. 3.

We can see that tree-based approaches are not private under Refresh due to the correlation between the keys of tags, which violate the first implication in Subsection IV-D.

Algorithm 3 Compromising attack on tree-based approaches

1: In the Learning phase, the adversary $\mathcal{A}$ selects a number of tags, and queries $Corrupt$ oracles of these tags.
2: $\mathcal{A}$ chooses two distinct tags $\mathcal{T}_i$ and $\mathcal{T}_j$ arbitrarily from the set of uncompromised tags.
3: $\mathcal{A}$ submits $\mathcal{T}_i$ and $\mathcal{T}_j$ as its challenge candidates.
4: In the Re-learning phase, $\mathcal{A}$ queries all oracles of $\mathcal{T}_b^*$, except $Corrupt$ oracle.
5: $\mathcal{A}$ analyzes the output of $\mathcal{T}_b^*$, then guesses b accordingly.

Through above formal analysis, we find that some well-known PPA schemes are not private under our Refresh model, since they do not satisfy the implications of Refresh (see Subsection IV-D). Specifically, the output of tags in OSK/AO and YA-TRAP is history-dependent, they are thereby inevitably vulnerable to the de-synchronization attack. On the other hand, due to the correlation of tags' keys, the tree based approaches fail to defend against the compromising attack. Therefore, a challenging issue emerges in designing PPA schemes: achieving the privacy as well as high authentication efficiency.

VI. LAST Design

To address the issue raised in Section V, we propose a Light-weight privAcy-preServing authenTication scheme, LAST, based on the Refresh model. We not only analyze the efficiency of LAST, but also prove its privacy under Refresh. We show that LAST is Refresh Private with extremely high efficiency.

A. Scheme

Based on Def. 1, LAST consists of following components.

- $KeyGen(1^s)$: generates two sequences of keys, $k_1, \ldots, k_n$ and $Index_1, \ldots, Index_n$, depending on a security parameter s. Each key is generated independently to all others, here $Index_i$ is the index of tag $\mathcal{T}_i$ in the back-end database, and k_i is the key used in the authentication procedure.
- $SetupTag(\mathcal{T})$: returns a specific secret pair $(Index, k)$ for a particular tag $\mathcal{T}$. The tuple $(\mathcal{T}_i, Index_i, k_i)$ exists in the back-end database when the tag $\mathcal{T}_i$ is legitimate.
- $SetupReader$: stores all tuples $(\mathcal{T}_i, Index_i, k_i)$, $1 \leq i \leq n$, in the back-end database.
- P is a polynomial-time interactive protocol between $\mathcal{R}$ and $\mathcal{T}_i$ in which $\mathcal{R}$ uses common parameters and the secret tuples. If the reader failed, it outputs $\perp$; otherwise, $\mathcal{R}$ outputs the ID of tag $\mathcal{T}_i$ and updates the corresponding tuple in database.

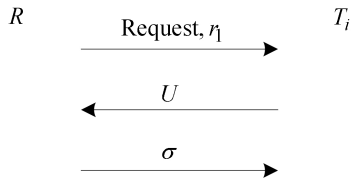

Fig. 4. Authentication Procedure in LAST. Upon U, $\mathcal{R}$'s operations are: 1. authenticating $\mathcal{T}_i$ and key-updating; 2. computing σ and sending it to $\mathcal{T}_i$. $\mathcal{T}_i$ also updates its keys after checking σ.

The protocol P includes three rounds, as illustrated in Fig. 4. In the first round, $\mathcal{R}$ generates a random number r_1 (a nonce), and then sends r_1 with the "Request" message to $\mathcal{T}_i$. In the second round, $\mathcal{T}_i$ generates a random number r_2 and computes $h(r_1, r_2, k_i)$, where $h(r_1, r_2, k)$ denotes the output of a cryptographic hash function h on three inputs: a key k of $\mathcal{T}$ and two random numbers r_1 and r_2. $\mathcal{T}_i$ computes $V = h(r_1, r_2, k_i)$, and then replies $\mathcal{R}$ with a message $U = (r_2, Index_i, V)$. $\mathcal{R}$ authenticates $\mathcal{T}_i$ according to U.

Upon U, $\mathcal{R}$ searches for an $Index_i$ in the back-end database, and then output $\perp$ if $Index_i$ does not exist; otherwise, $\mathcal{R}$ picks up the k_i belonging to the tuple $(\mathcal{T}_i, Index_i, k_i)$, and computes $V' = h(r_1, r_2, k_i)$. If $V' = V$, $\mathcal{R}$ accepts the tag and identifies it as $\mathcal{T}_i$; otherwise, $\mathcal{R}$ outputs $\perp$.

After authenticating $\mathcal{T}_i$, $\mathcal{R}$ will update the keys of $\mathcal{T}_i$. $\mathcal{R}$ computes $Index_i' = h(r_1, r_2, Index_i, k_i)$ and $k_i' = h(r_1, r_2, k_i)$, respectively, and then replaces the tuple $(\mathcal{T}_i, Index_i, k_i)$ in the database with $(\mathcal{T}_i, Index_i', k_i')$. After the key-updating procedure, $\mathcal{R}$ replies $\mathcal{T}_i$ the message $\sigma = h(r_1, r_2, k_i')$.

Upon receiving σ, $\mathcal{T}_i$ first computes $k_i' = h(r_1, r_2, k_i)$ and $\sigma' = h(r_1, r_2, k_i')$. Then $\mathcal{T}_i$ checks whether σ' equals σ. If yes, $\mathcal{T}_i$ updates original k_i and $Index_i$ to k_i' and $Index_i' = h(r_1, r_2, Index_i, k_i)$, respectively; otherwise, $\mathcal{T}_i$ discards σ and keeps $Index_i$ and k_i unchange.

B. Efficiency

We first investigate the storage efficiency of LAST, and then analyze the authentication efficiency by estimating the number of hash computations in each authentication.

An RFID tag normally has very tiny memory to store user's information as well as keys. The storage efficiency thereby is critical in designing PPA schemes. LAST is efficient in key storage. Specifically, LAST only allocates two keys for each tag, the index and authentication key. On the reader side, the storage is $2n$. The storage efficiency is similar to that of HashLock (one key stored on tag side and n keys on reader side), whereas higher than that of tree-based approaches ($\mathcal{O}(\log n)$ on tag side and $\mathcal{O}(n \log n)$ on reader side [13]).

The basic operation in LAST is hash computation. Therefore, we can use the number of hash computations to evaluate the authentication efficiency of LAST. According to the authentication procedure, $\mathcal{R}$ just needs four hash computations: one for authentication, three for key-updating. Thus, the authentication of LAST has the complexity $\mathcal{O}(1)$ on the authentication efficiency, which is much more efficient than previous schemes.

C. Privacy

Based on the Refresh model, we formally prove that LAST can achieve Refresh Privacy.

Theorem 1: LAST achieves $(r, t, c) - Refresh - Privacy$ in random oracle model, for any polynomial-time adversary $\mathcal{A}$, i.e., for any r, t, and c polynomial in the security parameter s.

Proof: In this proof, we employ the *random orale* (RO) model [3], in which cryptographic hash functions are treated as arbitrary random functions. We denote the game $\mathbf{Game}_{\mathcal{A}}^{ref-priv}$ (depicted in Fig. 2) between the challenger $\mathcal{C}$ and the polynomial adversary $\mathcal{A}$ as $\mathbf{G}_0$.

We specify a simulator **Sim**, who plays a game $\mathbf{G}_1$ with $\mathcal{A}$, to simulate the real challenger $\mathcal{C}$. Here $\mathbf{G}_1$ is derived from $\mathbf{G}_0$ by replacing the hash function and some oracles in $\mathbf{G}_0$ with an RO and random functions, respectively. **Sim** does not have any knowledge about the value of b or any pair ($Index_i$ and k_i) (according to $\mathbf{G}_0$, these values are hold by $\mathcal{C}$). Hence, $\mathcal{A}$'s advantage in $\mathbf{G}_1$ is zero. In other word, $\mathcal{A}$ gains no knowledge from **Sim** in $\mathbf{G}_1$. Based on the random oracle model, if we can construct **Sim** that is indistinguishable with the challenger $\mathcal{C}$, we can safely claim that $\mathcal{A}$ will not gain any knowledge about tags in $\mathbf{G}_0$. Thus, the privacy of tags in the real RFID systems is preserved.

According to $\mathbf{G}_0$, $\mathcal{A}$ chooses two tags $\mathcal{T}_i$ and $\mathcal{T}_j$ from uncorrupted tags. Let $\mathcal{H}$ be the random oracle for the hash function h. $\mathcal{H}$ is a list of hash values, H_list, maintained by **Sim**. H_list is initialized as empty. The format of each entry in H_list is (u, v). For a query u to $\mathcal{H}$, **Sim** first checks whether u exists in H_list. If yes, **Sim** returns the corresponding v as the hash value of $h(u)$; otherwise, **Sim** generates a v uniformly at random, then returns v as the value of $h(u)$, and appends (u, v) into H_list.

In $\mathbf{G}_1$, **Sim** simulates the result of $TagQuery$ call to $\mathcal{T}_b^*$:

After receiving the "Request" and r_1, **Sim** simulates the $TagQuery$ oracle as follows:

- In the Learning phase,
  - if $TagQuery$ is queried for the first time, **Sim**
    - generates r_2 uniformly at random, and then queries $\mathcal{H}$ to get $Index$ and V;
    - returns $U = (r_2, Index, V)$;
  - otherwise,
    - generates r_2 uniformly at random and queries $\mathcal{H}$ to get V;
    - returns $U = r_2, Index, V$.
- In the Re-learning phase,
  - for $\mathcal{T}_b^*$,
    - if $TagQuery$ is queried for the first time in the Learning and Re-learning phases, **Sim**
      - generates r_2 uniformly at random, and then queries $\mathcal{H}$ to get $Index_b$ and V;
      - returns $U = (r_2, Index_b, V)$, where $Index_b$ is the existing index key of $\mathcal{T}_b^*$;
    - otherwise,
      - generates r_2 uniformly at random and queries $\mathcal{H}$ to get V;
      - returns $U = r_2, Index_b, V$, where $Index_b$ is the existing index key of $\mathcal{T}_b^*$.
  - For other tags: same operations as those in the Learning phase.

Sim simulates the $ReaderSend$ oracle by generating a random σ, and returning it to $\mathcal{A}$.

$\mathbf{G}_1$ is similar to $\mathbf{G}_0$ except the random oracle and the constructions of the $TagQuery$ and $ReaderSend$ oracles. In $\mathbf{G}_1$, from the view point of $\mathcal{A}$, **Sim** simulates $\mathcal{C}$ perfectly except following events happens:

1) Collisions in the input of the hash function h. For example, in $\mathbf{G}_0$, the hash function h will treat $(r_1, r_2, \cdot)$ and $(r_2, r_1, \cdot)$ as same input, therefore the output of h should be identical. However, in $\mathbf{G}_1$, according to the definition of random oracle, $\mathcal{H}$ considers that $(r_1, r_2, \cdot)$ and $(r_2, r_1, \cdot)$ are different, the output of $\mathcal{H}$ of course is different. Thus, **Sim** cannot answer $\mathcal{A}$'s query correctly. We denote this event as $Event_1$.
2) Collisions in the output of $\mathcal{H}$. Since $\mathcal{A}$ performs at most c computations, the number of $\mathcal{H}$ is not more than c. We denote this event as $Event_2$.
3) $\mathcal{A}$ guesses the correct key of k_i or k_j. In this case, $\mathcal{A}$ owns the correct keys of $\mathcal{T}_i$ or $\mathcal{T}_j$. Thus, $\mathcal{A}$ can find that the output from **Sim** are not correct. We denote this event as $Event_3$.

The probabilities of $Event_1$ and $Event_2$ are bounded by the birthday paradox:

$$\mathbf{Pr}[Event_1 \vee Event_2] \leq \frac{(r+t)^2 + c^2}{2 \cdot 2^s}$$

For $Event_3$, given that $\mathcal{H}$ is a random oracle, its output reveals no information about the secret keys. Hence, the probability that $\mathcal{A}$ can successfully guess k_i or k_j is at most $\frac{2c}{2^s}$.

As discussed at the beginning of the proof, the advantage of $\mathcal{A}$ in $\mathbf{G}_1$ is zero. Considering the probabilities of any event happening, the advantage of $\mathcal{A}$ in $\mathbf{G}_0$ is bounded by:

$$\begin{aligned} & \mathbf{Adv}_{LAST}(\mathcal{A}) = \mathbf{Pr}[Event_1 \vee Event_2 \vee Event_3] \\ \leq \quad & \frac{(r+t)^2 + c^2}{2 \cdot 2^s} + \frac{2c}{2^s} = \frac{(r+t)^2 + c^2 + 4c}{2^{s+1}} \end{aligned}$$

Obviously, the advantage of $\mathcal{A}$ is negligible. Based on the Def. 3, LAST is (r, t, c)-Refresh-Privacy. ■

D. Other Security Objects

In this subsection, we show that LAST also achieves other security objectives, which are required in the PPA scheme design [13], [5].

Cloning Resistance: An adversary may impersonate a valid tag via repeatedly forwarding valid responses to the reader, which is termed as the cloning attack [13]. In LAST, two random numbers r_1 and r_2 are exchanged in the authentication procedure to defend against the cloning attack. Since the random numbers r_1 and r_2 are generated uniformly at random and varied in every authentication procedure, it is infeasible for an adversary to predicate. In addition, if the length of r_1 and r_2 in LAST are sufficiently long (more than 64 bits),

the probability that the adversary can successfully guess the random numbers is negligible. Thus, LAST is not subject to the cloning attack.

Forward Secrecy: The Forward Secrecy guarantees that $\mathcal{A}$ cannot reveal the previous messages sent from the tags even if $\mathcal{A}$ obtains the current keys used by those tags. In LAST, the keys stored in a tag are updated after each interrogation from the legitimate reader. By this means, LAST prevents the adversary, even it obtains the current keys used by a tag, from retrieving any useful information through the previous output sent by the tag. The only way that the adversary can recover the past messages is to successfully invert the one-way cryptographic hash function, which is computationally infeasible. On the contrary, many balanced tree based protocols [5], [15] cannot update the keys of tags, the adversary can easily reveal all past authentication messages sent by a corrupted tag.

VII. Conclusions and Future Work

In this paper, we propose a weak privacy model, Refresh, which meets the privacy requirements of RFID systems in practice. We also examine several popular PPA schemes and show that they are not private under our model. We further propose a secure PPA scheme, LAST, and prove its privacy. LAST achieves extremely high authentication efficiency with acceptable privacy.

Following the current work, there are still much room to make our model more robust. First, we now consider an RFID system with only one reader. In the case of multiple readers in the system, there may exist several problems. For example, if different readers have different acceptance rates for some tags, an adversary can mark a tag with the acceptance rate of a certain reader. Thus, the adversary can distinguish this tag from others and compromise the tag's privacy. In the future, we will take multiple readers into consideration. Second, as we mentioned in Subsection IV-D, Refresh is a weak model suitable for those applications in which tags are interrogated frequently. For those applications focusing on the location privacy, for example logistics, tags may be tracked in a short time. In the future, we will extend Refresh to fit such applications.

Acknowledgment

This work is supported in part by the NSF CNS-0832120, National Natural Science Foundation of China under Grant No.60903155, and No. 60828003, NSFC/RGC N_HKUST602/08, National High Technology Research and Development Program of China (863 Program) under grant No.2007AA01Z180, National Basic Research Program of China (973 Program) under Grant No.2006CB303000, and No. 2010CB328100, Hong Kong ITF GHP /044/07LP and NSF China Key Project 60736016.

References

[1] G. Avoine. Adversary Model for Radio Frequency Identification. Technical Report, LASEC-REPORT-2005-001, EPFL, 2005.

[2] G. Avoine and P. Oechslin. A Scalable and Provably Secure Hash Based RFID Protocol. In *IEEE PerCom, Workshop Security*, 2005.

[3] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security – CCS*, 1993.

[4] T. Dimitriou. A Lightweight RFID Protocol to Protect against Traceability and Cloning attacks. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm*, 2005.

[5] T. Dimitriou. A Secure and Efficient RFID Protocol that Could make Big Brother (partially) Obsolete. In *International Conference on Pervasive Computing and Communications – PerCom*, 2006.

[6] P. Golle, M. Jakobsson, A. Juels, and P. Syverson. Universal Re-Encryption for Mixnets. In *The Cryptographers' Track at the RSA Conference – CT-RSA*, 2004.

[7] M. Hellman. A Cryptanalytic Time-Memory Trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, July 2006.

[8] A. Juels. Minimalist Cryptography for Low-Cost RFID Tags. In *International Conference on Security in Communication Networks – SCN*, 2004.

[9] A. Juels. RFID Security and Privacy: A Research Survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, 2006.

[10] A. Juels and S. Weis. Defining Strong Privacy for RFID. In *IEEE PerCom, Workshop Security*, 2007.

[11] T. Kriplean, E. Welbourne, N. Khoussainova, V. Rastogi, M. Balazinska, G. Borriello, T. Kohno, and D. Suciu. Physical Access Control for Captured RFID Data. *IEEE Pervasive Computing*, 6(4):48–55, 2007.

[12] Y. Li and X. Ding. Protecting RFID Communications in Supply Chains. In *ACM Symposium on Information, Computer and Communications Security – AsiaCCS*, 2007.

[13] L. Lu, J. Han, Y. Liu, L. Hu, and L. M. Ni. Dynamic Key-updating: Privacy-Preserving Authentication for RFID Systems. In *IEEE PerCom*, 2007.

[14] D. Molnar, A. Soppera, and D. Wagner. A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags. In *Selected Areas in Cryptography – SAC*, 2005.

[15] D. Molnar and D. Wagner. Privacy and Security in Library RFID: Issues, Practices, and Architectures. In *ACM CCS*, 2004.

[16] L. M. Ni, Y. Liu, Y. Lau, and A. P. Patil. LANDMARC: Indoor Location Sensing Using Active RFID. In *IEEE PerCom*, 2003.

[17] M. Ohkubo, K. Suzuki, and S. Kinoshita. Efficient Hash-Chain Based RFID Privacy Protection Scheme. In *International Conference on Ubiquitous Computing – Ubicomp, Workshop Privacy*, 2004.

[18] R. Paise and S. Vaudenay. Mutual Authentication in RFID: Security and Privacy. In *ACM Symposium on Information, Computer and Communications Security – ASIACCS*, 2008.

[19] P. Robinson and M. Beigl. Trust Context Spaces: An Infrastructure for Pervasive Security in Context-Aware Environments.

[20] G. Tsudik. YA-TRAP: Yet Another Trivial RFID Authentication Protocol. In *IEEE PerCom*, 2006.

[21] S. Vaudenay. On Privacy Models for RFID. In *Advances in Cryptology – Asiacrypt*, 2007.

[22] S. Weis, S. Sarma, R. Rivest, and D. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *International Conference on Security in Pervasive Computing – SPC*, 2003.