Scalable RFID Systems: a Privacy-Preserving Protocol with Constant-Time Identification

Basel Alomair and Radha Poovendran
Network Security Lab (NSL), University of Washington
e-mail: {alomair,rp3}@u.washington.edu

Abstract. In RFID literature, most "privacy-preserving" protocols require the reader to search all tags in the system in order to identify a single tag. In another class of protocols, the search complexity is reduced to be logarithmic in the number of tags, but it comes with two major drawbacks: it requires a large communication overhead over the fragile wireless channel, and the compromise of a tag in the system reveals secret information about other, uncompromised, tags in the same system. In this work, we take a different approach to address time-complexity of private identification in large-scale RFID systems. We utilize the special architecture of RFID systems to propose the first symmetric-key privacy-preserving authentication protocol for RFID systems with constant-time identification. Instead of increasing communication overhead, the existence of a large storage device in RFID systems, the database, is utilized for improving the time efficiency of tag identification.

Keywords. RFID, privacy, identification complexity, scalability, authentication

1. Introduction

The ability to trace RFID tags, and ultimately the individuals carrying them, is a major privacy concern in RFID systems. Privacy activists have been worried about the invasion of users' privacy by RFID tags, calling for the delay or even the abandonment of their deployment. In extreme cases, companies have been forced to repudiate their plans for RFID deployment in response to the threat of being boycotted [1]. Consequently, significant effort has been made in the direction of designing RFID systems that preserve users' privacy.

Two main objectives of typical RFID systems are identification and privacy. Identification, by itself, can be as straightforward as broadcasting tags' identifiers in clear text. When combined with the privacy requirement, however, transmitting identifiers in clear text is obviously unacceptable. For RFID tags capable of performing asymmetric cryptography, such as public-key encryption [2,3] or trapdoor functions [4,5], private identification can be achieved easily (for instance, by encrypting a randomized version of the tag's ID with the reader's public key).

Public-key operations, however, are beyond the computational capabilities of low-cost tags. Hoping Moore's law will eventually render tags capable of performing public-key operations, one might consider the computational limitations of RFID tags a temporary problem. The price of tags, however, will be a determining factor in the deployment of RFID systems in real life applications. When RFID systems are to replace barcodes to identify tagged items, the price of tags will contribute to the tagged products' prices. When retailers are to choose between tags that can perform sophisticated cryptographic operations and cheaper tags that cannot, it seems highly likely that the cheaper tags will prevail. Consequently, low-cost RFID systems are restricted to the use of symmetric-key cryptography in most practical scenarios.

Privacy-preserving symmetric-key protocols are faced with the following paradox. On one side, a tag must encrypt its identity with its secret key so that only authorized readers can extract the identity. On the opposite side, authorized readers must first determine the identity of the tag in order to know which key is to be used for decryption. Therefore, given that tags' responses are randomized (to protect users' privacy), and that the length of tags' responses is sufficiently long (so that easy to implement attacks such as random guessing and exhaustive search will have small probability of success), searching the database for those responses is a nontrivial task.

Most RFID protocols trade-off identification efficiency for the sake of privacy. That is, private identification is accomplished, but the reader is required to perform a linear search among all tags in the system in order to identify the tag being interrogated (see, e.g., [6,7,8,9,10]). In a typical protocol of this class, the reader interrogates a tag by sending a random nonce, r_1. The tag generates another nonce, r_2, computes $h(ID, r_1, r_2)$, where h is a cryptographic hash function, and responds with $s = (r_2, h(ID, r_1, r_2))$. (Different protocols implement variants of this approach; but this is the main idea of this class of protocols.) Upon receiving the tag's response, the reader performs a linear search of all the tags in the system, computing the hash of their identifiers with the transmitted nonces, until it finds a match. Obviously, unauthorized observers cannot correlate different responses of the same tag, as long as the nonce is never repeated.

Although protocols of this class have been shown to provide private identification, their practical implementation has a scalability issue. In a large-scale RFID system, performing a linear search for every identification run can be a cumbersome task, especially in applications requiring identification of multiple tags simultaneously (which is the typical scenario in many RFID applications). Moreover, denial of service attacks can be launched by giving authorized readers false identifiers causing them to perform exhaustive search amongst all tags in the system before realizing that the received response is invalid. Hence, for an RFID system to be practical, one must aim for a scheme that can break the barrier of *linear-time* identification complexity.

A big step towards solving the scalability issue in privacy-preserving RFID systems was proposed by Molnar and Wagner in [11]. This new approach traded-off computational and communication overhead on tags to speed up the identification process. The authors utilized a tree data structure, where each edge in the tree corresponds to a unique secret key, each leaf of the tree corresponds to a unique tag, and each tag carries the set of keys on the corresponding path from the root of the tree to its leaf. When a reader interrogates a tag, the tag responds with a message encrypted with its first key. By decrypting the tag's response with the keys corresponding to all edges of the first level of the tree, the reader can determine to which edge the tag belongs. By traversing the tree from top to bottom, the tag can be identified in $O(\log N_T)$ time using $O(\log N_T)$ reader-tag interactions, where N_T is the number of tags in the system.

Arranging tags in a tree based on secret keys they possess, however, introduced a new security threat to the RFID system: every compromised tag will reveal the secret keys from the root of the tree to its leaf. Since these keys are shared by other tags in the system, compromising one tag will reveal secret information about all tags sharing a subset of those keys. In [8], the tree structure is analyzed showing that in a tree with a branching factor of two, compromising 20 tags in a system of 2^{20} tags leads to the identification of uncompromised tags with an average probability close to one.

Researchers who believe that reducing identification complexity from $O(N_T)$ to $O(\log N_T)$ cannot be overlooked as a result of the vulnerability it introduced have been making significant effort to mitigate the tag compromise problem in tree based systems [12,13,14]. The idea shared by all such attempts is to employ key updating mechanisms to mitigate the effect of tag capture. Other researchers, however, believe that the new threat overweighs the reduction in identification complexity, thus, proceeding with the linear-time class of protocols and trying to improve on its performance (see, e.g., [8,9,10]).

Another major drawback of the tree based class of protocols is the increase in communication and computation overhead on tags. In a typical RFID system, the reader interrogates multiple tags simultaneously. Consequently, even in the linear-time identification protocols, where communication overhead is $O(1)$, collision avoidance and medium access control are among the most challenging problems in the design of efficient RFID systems [15,16,17,18,19]. Increasing the communication overhead to $O(\log N_T)$ can only complicate collision avoidance even further. Moreover, extra computation overhead can also be problematic for passive tags as it leads to more energy consumption.

We point out here that this work takes only into account protocols that are both secure and provide private identification. That is, although there exist protocols that can provide constant-time identification, they either fail to provide privacy against active adversaries or fail to provide secure authentication. Discussion of such protocols is deferred to Section 2 (the related work section).

In this paper, we address the private identification problem in large-scale RFID systems. We propose a protocol that, in addition to being *resilient to tag compromise attacks*, allows *constant-time identification*, without imposing extra *communication or computation overhead* on the resource limited tags. The main drive behind devising our protocol is the intuition that, in order to overcome the problems in both linear and logarithmic time identification classes, one must aim for a solution that is fundamentally different than both of them. We do not resort to tree structure, nor do we incur more communication overhead. Instead, we utilize resources that are already available in RFID systems to improve identification efficiency. That is, since in any RFID system there is a database, to store information about tags in the system, and since storage is relatively cheap in today's technology, we tradeoff storage for the sake of better identification efficiency. To the best of our knowledge, the proposed protocol is the first symmetric-key privacy-preserving protocol that allows constant-time tag identification. Table 1 compares our protocol to the class of linear-time identification protocols, Class 1, and to the class of log-time identification protocols, Class 2.

The rest of the paper is organized as follows. In Section 2, we discuss some related work in the design of RFID systems. In Section 3 we describe our system model, adversarial model, and security model. The proposed system is described in Section 4. In Section 5, we prove our claim of constant-time identification and provide a case study

Table 1. Performance comparison as a function of the number of tags in the system, N_T. Class 1 represents protocols with linear-time identification, while Class 2 represents protocols with log-time identification. The overhead in the last column refers to computation and communication overhead on the tags' side.

	Search time	Key size	Database size	Overhead
Class 1	$O(N_T)$	$O(1)$	$O(N_T)$	$O(1)$
Class 2	$O(\lg N_T)$	$O(\lg N_T)$	$O(N_T)$	$O(\lg N_T)$
Proposed	$O(1)$	$O(1)$	$O(N_T)$	$O(1)$

in Section 6. Section 7 is dedicated to the security proofs of the proposed system. The robustness against tag capture attacks is detailed in Section 8. In Section 9, we discuss desynchronization attacks against the proposed system and extend our system to prevent such attacks. We conclude our paper in Section 10.

2. Related Work

Many protocols have been designed to meet the stringent computational capabilities of low-cost RFID systems. The class of stateful protocols is an example of such protocols. In stateful protocols, the tag maintains a state that can allow authorized readers to identify it. To avoid tag impersonation, the state gets updated with the completion of an identification run with an authorized reader. Since every tag's state must be the same as the state stored at the database, such protocols must be designed to be secure against desynchronization attacks. Obviously, since tags are identified via their states, which are sent in the clear, readers can identify tags responses in constant time.

However, such protocols are not designed to provide privacy against active adversaries. To see this, recall that the tag cannot update its state without completing an identification run with an authorized reader (since the tag must always be synchronized with the database). Therefore, an active adversary interrogating the same tag multiple times, in the absence of an authorized reader, will receive the same response (the tag's current state). In other words, such stateful protocols, although might be suitable for some applications, cannot be used in applications where privacy against active adversaries is required. Examples of such protocols include, but are not limited to, [20,21,22,23,24]. This class of protocols, however, remains attractive due its fast identification and the fact that it requires less computation effort on the tags. Consequently, many noncryptographic techniques have been proposed to increase tags privacy. Such techniques include the use of a blocker tag [25], the use of a watchdog tag [26], the use of an RFID guardian [27], and the use of an RFID Enhancer Proxy (REP) [28].

Another class of protocols that claim to achieve constant-time identification is the class based on time stamps. Protocols of this class, however, have been analyzed and shown to lack some required security propeeties (see, e.g., [29,30]). Protocols of this class include [31,32,33].

To overcome the lack of privacy against active adversary in stateful protocols, linear-time identification protocols were introduced. If properly designed, linear-time identification protocols can provide privacy against both passive and active adversaries as well as secure reader-tag mutual authentication. One advantage of the linear class of protocols is that it can be designed without any synchronization requirement. This lack of synchronization, however, is the main reason why authorized readers have to perform a linear search amongst all tags in the system to identify each response. Due to its desirable security and privacy characteristics, however, this class has attracted the most attention from researchers in the RFID technology. Protocols of this class include, but are not limited to, [6,7,8,9,10].

The main drive behind the introduction of tree based protocols is to decrease the extensive amount of computational power required to identify tags in the linear-time protocols. Most RFID tags, however, are not tamper resistant. Hence, capturing an RFID tag and obtaining its secret information is not a complicated task. For the linear-time protocols this does not pose a real problem, since the secret information of each tag is independent of the others. For the logarithmic-time protocols, this is not the case. Consequently, many protocols have been proposed to address the tag compromise vulnerability in logarithmic-time identification systems. Protocols of this class include, but are not limited to, [11,12,34,13,14].

3. Model Assumptions

In this section, we state the system, adversarial, and security model used to develop this protocol. The comprehensive details of the adversarial and the security models are not needed for the description of the protocol and can be skipped for a better flow of ideas.

3.1. System Model

RFID systems are typically composed of three main components: tags, readers, and a database. In our model, the tag is assumed to have limited computing power: hash computations are the most expensive operations tags can perform. The reader is a computationally powerful device with the ability to perform sophisticated cryptographic operations. The database is a storage resource at which information about tags in the system is stored. Readers-database communications are assumed to be secure.

We assume that tags have nonvolatile memory so they can retain their keying information and carry out necessary updates. Although this assumption is already practical for passive tags with today's technology, depending on the reader's transmission power and its distance from the tag, technology will only improve and make this assumption even more practical. Indeed, most RFID protocols are based on the same assumption (see, e.g., [10,12,13,14]).

3.2. Adversarial Model

We assume adversaries with complete control over the communication channel. Adversaries can observe all exchanged messages, modify exchanged messages, block exchanged messages and replay them later, and generate messages of their own. We do not consider an adversary whose only goal is to jam the communication channel. Distinguishing tags by the physical fingerprints of their transmissions requires sophisticated devices and cannot be solved using cryptographic solutions. It is out of the scope of this work as in the majority of similar proposals.

The adversary $\mathcal{A}$ is modeled as a polynomial-time algorithm. Given a tag, T, and a reader, R, we assume $\mathcal{A}$ has access to the following oracles:

- *Query* (T, m_1, x_2, m_3): $\mathcal{A}$ sends m_1 as the first message to T; receives a response, x_2; and then sends the message $m_3 = f(m_1, x_2)$. This oracle models the adversary's ability to interrogate tags in the system.
- *Send* (R, x_1, m_2, x_3): $\mathcal{A}$ receives x_1 from the reader R; replies with $m_2 = f(x_1)$; and receives the reader's response x_3. This oracle models the adversary's ability to act as a tag in the system.
- *Execute* (T, R): The tag, T, and the reader, R, execute an instance of the protocol. $\mathcal{A}$ eavesdrops on the channel, and can also tamper with the messages exchanged between T and R. This oracle models the adversary's ability to actively monitor the channel between tag and reader.
- *Block* $(\cdot)$: $\mathcal{A}$ blocks any part of the protocol. This query models the adversary's ability to launch a denial of service attack.
- *Reveal* (T): This query models the exposure of the tags' secret parameters to $\mathcal{A}$. The oracle simulates the adversary's ability to physically capture the tag and obtain its secret information.

$\mathcal{A}$ can call the oracles *Query*, *Send*, *Execute*, and *Block* any polynomial number of times. The *Reveal* oracle can be called only once (on the same tag), at which the tag is considered compromised and, thus, there is no point of calling the *Reveal* oracle on the same tag multiple times. To model tag compromise attacks, however, the adversary is allowed to call other oracles after the *Reveal* oracle on the same tag; detailed discussion about this is provided in Section 8.

3.3. Security Model

The security model presented in this section does not consider the adversary's ability to perform pre-processing before engaging in the games. In Section 8, however, we will modify the security model to give the adversary such ability to perform pre-processing that involves calling the *Reveal* oracle on tags in the system. The main purpose of this modification is to allow modeling tag compromise attacks.

The two main security goals of our protocol are tags' privacy and tag-reader mutual authentication. There are different notions of privacy in the RFID literature (see, e.g., [35,29,36]). In this paper, privacy is measured by the adversary's ability to trace tags by means of their responses in different protocol runs. We define three notions of untraceability, *universal*, *forward*, and *existential*.

Definition 1 (Universal Untraceability) *In an RFID system, tags are said to be universally untraceable if an adversary cannot track a tag based on information gained before the tag's last authentication with a valid reader. In other words, there is no correlation between a tag's responses before and after completing a protocol run with a valid reader.*

Universal untraceability is modeled by the following game between the challenger C (an RFID system) and a polynomial time adversary $\mathcal{A}$.

1. C selects two tags, T_0 and T_1, and a valid reader, R.
2. $\mathcal{A}$ makes queries on T_0, T_1, and R using the *Query*, *Send*, *Execute*, and *Block* oracles for a number of times of its choice.
3. $\mathcal{A}$ stops calling the oracles and notifies C.

4. C carries out an instance of the protocol with T_0 and T_1, during which mutual authentication of both tags with R is achieved.
5. C selects a random bit, b, and sets $T = T_b$.
6. $\mathcal{A}$ makes queries of T and R using the *Query*, *Send*, *Execute*, and *Block* oracles.
7. $\mathcal{A}$ outputs a bit, b', and wins the game if $b' = b$.

The second notion of privacy, forward untraceability, is defined as follows.

Definition 2 (Forward Untraceability) *In an RFID system with forward untraceability, an adversary capturing the tag's secret information cannot correlate the tag with its responses before the last complete protocol run with a valid reader.*

Forward untraceability is modeled by the following game between C and $\mathcal{A}$.

1. C selects two tags, T_0 and T_1, and a valid reader, R.
2. $\mathcal{A}$ makes queries of T_0, T_1, and R using the *Query*, *Send*, *Execute*, and *Block* oracles for a number of times of its choice.
3. $\mathcal{A}$ stops calling the oracles and notifies C.
4. C carries out an instance of the protocol with T_0 and T_1, during which mutual authentication of both tags with R is achieved.
5. C selects a random bit, b, and sets $T = T_b$.
6. $\mathcal{A}$ calls the oracle *Reveal* (T).
7. $\mathcal{A}$ outputs a bit, b', and wins the game if $b' = b$.

Finally, the third notion of privacy, existential untraceability, is defined as follows.

Definition 3 (Existential Untraceability) *Tags in an RFID system are said to be existentially untraceable if an* active *adversary cannot track a tag based on its responses to multiple interrogation, even if the tag has not been able to accomplish mutual authentication with an authorized reader.*

Existential untraceability is modeled by the following game between C and $\mathcal{A}$.

1. C selects two tags, T_0 and T_1.
2. $\mathcal{A}$ makes queries of T_0 and T_1 using the *Query* oracle for at most $C - 1$ number of times for each tag, where C is a pre-specified system security parameter.
3. $\mathcal{A}$ stops calling the oracles and notifies C.
4. C selects a random bit, b, and sets $T = T_b$.
5. $\mathcal{A}$ makes a query of T using the *Query* oracle.
6. $\mathcal{A}$ outputs a bit, b', and wins the game if $b' = b$.

To quantify the adversary's ability to trace RFID tags, we define the adversary's advantage of successfully identifying the tag in the previous games as

$$Adv_{\mathcal{A}} = 2\Big(\Pr[b' = b] - \frac{1}{2}\Big). \tag{1}$$

If the adversary cannot do any better than a random guess, then $\Pr(b' = b) = 1/2$. Consequently, the adversary's advantage, $Adv_{\mathcal{A}}$, is zero, at which point we say that tags are untraceable.

The other security goal of our protocol is mutual authentication. An *honest protocol run* is defined as follows [24]: A mutual authentication protocol run in the symmetric key setup is said to be honest if the parties involved in the protocol run use their shared key to exchange messages, and the messages exchanged in the protocol run have been relayed faithfully (without modification).

Another term that will be used for the reminder of the paper is the definition of negligible functions: A function $\gamma : \mathbb{N} \to \mathbb{R}$ is said to be negligible if for any nonzero polynomial $\wp$, there exists N_0 such that for all $N > N_0$, $|\gamma(N)| < (1/|\wp(N)|)$. That is, the function is said to be negligible if it converges to zero faster than the reciprocal of any polynomial function.

We now give the formal definition of secure mutual authentication for RFID systems as appeared in [24].

Definition 4 (Secure Mutual Authentication) *A mutual authentication protocol for RFID systems is said to be secure if and only if it satisfies all the following conditions:*
1. No information about the secret parameters of an RFID tag is revealed by messages exchanged in protocol runs.
2. ***Authentication*** $\Rightarrow$ ***Honest protocol:*** *the probability of authentication when the protocol run is not honest is negligible in the security parameter.*
3. ***Honest protocol*** $\Rightarrow$ ***Authentication:*** *if the protocol run is honest, the tag-reader pair must authenticate each other with probability one.*

To model the adversary's attempt to authenticate herself to a reader (tag), we propose the following game between the challenger C and adversary $\mathcal{A}$.

1. C chooses a tag, T, at random, and a reader, R.
2. $\mathcal{A}$ calls the oracles *Query*, *Send*, *Execute*, and *Block* using T and R for a number of times of its choice.
3. $\mathcal{A}$ decides to stop and notifies C.
4. $\mathcal{A}$ calls the oracle *Send* (*Query*) to impersonate a tag (reader) in the system.
5. If $\mathcal{A}$ is authenticated as a valid tag (reader), $\mathcal{A}$ wins the game.

Definition 4 implies that the protocol achieves secure mutual authentication only if the adversary's probability of winning the previous game is negligible.

4. System Description

4.1. Protocol Overview

In our system, each tag has an internal counter, c, and is preloaded with a unique *secret* pseudonym, ψ, and a secret key, k. The secret key and the secret pseudonym are updated whenever mutual authentication with a valid reader is accomplished, while the counter is incremented every time authentication fails.

When an RFID reader is to identify and authenticate a tag within its range, it generates a random nonce, $r \in_R \{0,1\}^L$, and transmits it to the tag. Upon receiving r, the tag computes $h(\psi, c)$ and $\tilde{r} := h(0, \psi, c, k, r)$, where ψ is the tag's current pseudonym, k is the tag's current secret key, c is the tag's internal counter, and r is the received nonce. The tag then increments its counter, $c \leftarrow c + 1$. With $h(\psi, c)$, the reader accesses the database to identify the tag and obtain its information, including its pseudonym, ψ, its secret key,

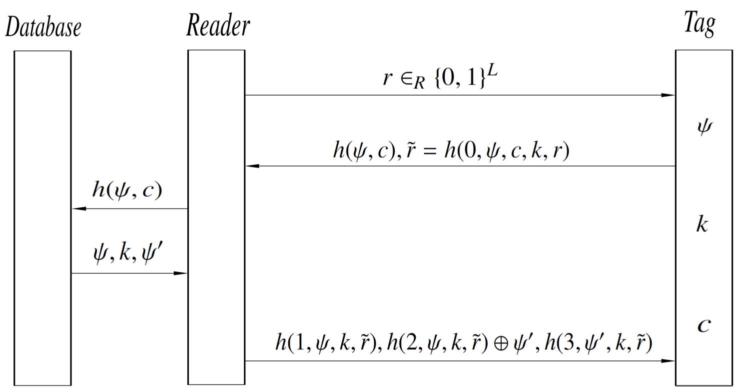

Figure 1. A schematic of one instance of the protocol.

k, and a new pseudonym, ψ', to update the tag. With $\tilde{r}$, the reader authenticates the tag by confirming its knowledge of the secret key, k, obtained from the database.

Once the tag has been identified and authenticated, the reader responds with $h(1, \psi, k, \tilde{r})$, $h(2, \psi, k, \tilde{r}) \oplus \psi'$, and $h(3, \psi', k, \tilde{r})$. With $h(1, \psi, k, \tilde{r})$, the tag authenticates the reader (by verifying its knowledge of its secret key, k). If the reader is authenticated, the tag uses $h(2, \psi, k, \tilde{r}) \oplus \psi'$ to extract its new pseudonym, ψ'. Once the new pseudonym has been computed, the tag verifies its integrity using $h(3, \psi', k, \tilde{r})$. The tag and the reader then update the tag's secret key to $k' = h(k)$ truncated to the required length, ℓ. Figure 1 depicts a single protocol run between an RFID reader-tag pair.

4.2. Database Overview

As mentioned above, the tag is identified by its randomized response, $h(\psi, c)$, which is an L-bit long string. Since security requires that L is sufficiently long, it is infeasible to construct a physical storage that can accommodate all possible 2^L responses, for direct addressing. (This is the reason why previous schemes resorted to linear search amongst all tags in the system to identify a response.) For ease of presentation, the structure of the database is divided into three logical parts, M-I, M-II, and M-III.

To allow for constant-time identification, with feasible storage, we truncate the L-bit identifiers to their s most significant bits, where s is small enough so that a storage of size 2^s is feasible. Of course, many identifiers will share the same s most significant bits (to be exact, 2^{L-s} possible identifiers will share the same truncated value). M-I is a table of size $O(2^s)$, with addresses ranging from 0 to $2^s - 1$, and each table entry contains a pointer to an entry in M-II (similar to a hashtable data structure, with truncation instead of hashing). All identifiers with the same s most significant bits will be stored in a smaller table in M-II, and the pointer at address s in M-I will point to the head of this smaller table. Finally, actual information about tags in the system is stored in M-III. Detailed construction of the database and description of the identification process will be the focus of the remainder of this section.

The proposed protocol can be broken into four main phases: parameters selection phase, system initialization phase, tag identification phase, and identity randomization and system update phase. Each phase is detailed below.

Table 2. A list of parameters and used notations.

Symbol	Definition
N_T	The total number of tags in the system
N	The total number of pseudonyms in the system
ψ_i	The pseudonym corresponding to the i^{th} tag
C	The maximum counter value
ℓ	The length of the secret parameter in bits
h	Cryptographic hash function
L	The output length of the used hash function
n	The length of the truncated hash values
$\Psi_{i,c}$	A tag identifier, $\Psi_{i,c} := h(\psi_i, c)$
$\Psi^n_{i,c}$	The n most significant bits of $\Psi_{i,c}$

4.3. Parameters Selection

During this phase, the database is initialized and each tag is loaded with secret information. The secret information includes the tag's secret key, which the tag and reader use to authenticate one another, and the tag's pseudonym, which is used for tag identification.

Given the total number of tags the RFID system is suppose to handle, N_T, and predefined security and performance requirements (more about this later), the system designer chooses the following parameters to start the initialization phase:

- The total number of pseudonyms, N. Since pseudonyms will be used as unique tag identifiers, there must be at least one pseudonym for every tag in the system. Furthermore, since tags are assigned new identifiers following every successful mutual authentication process with an authorized reader, the total number of pseudonyms must be greater than the total number of tags in the system, i.e., $N > N_T$.
- The maximum counter value, C. The counter is used by RFID tags to mitigate traceability by active adversaries; the larger the counter is, the more difficult it will be for active adversaries to track the tag; on the downside, the size of the database will grow linearly with the counter (the database size is $O(NC)$). Therefore, the size of the counter is a trade-off between tags' privacy and system complexity.
- The length, ℓ, in bits, of the tags' secret parameters (pseudonyms and keys). As in any symmetric key cryptosystem, ℓ should be chosen properly to prevent easy-to-implement attacks, such as exhaustive search and random guessing. Obviously, ℓ must be long enough to generate N distinct pseudonyms, i.e., $\ell \geq \lceil \log_2 N \rceil$. In practice, however, ℓ will be much longer.
- The hash function, h. In particular, the output length of the hash values, L, is of special importance. The length must be chosen large enough so that there are no collisions during database initialization, which is described below.
- The length, n, of the truncated hashes. The size of n is the key for constant-time identification and practicality of the system. It will be determined in Section 5.

Table 2 summarizes the list of system parameters and used notations.

4.4. System Initialization

Once the system parameters have been chosen, the initialization phase can start. The initialization phase can be summarized in the following steps.

1) Given the number of pseudonyms, N, and the length of each pseudonym, ℓ, the system designer draws, *without replacement*, N pseudonyms randomly from the set of all possible ℓ-bit strings. That is, N *distinct* pseudonyms, $\psi_1, \psi_2, \ldots, \psi_N$, are chosen at random from $\{0, 1\}^\ell$. Each tag is given a unique pseudonym and a secret key, and each tag's counter is initially set to zero. *We emphasize that the drawn pseudonyms are not publicly known; otherwise, tags' privacy can be breached.*

2) For each pseudonym, ψ_i, the hash value $h(\psi_i, c)$ is computed for all $i = 1, \ldots, N$ and all $c = 0, \ldots, C - 1$. That is, a total of NC hash operations must be performed, as depicted in Figure 2. Each row of the table in Figure 2 corresponds to the same pseudonym. Therefore, all entries in the i^{th} row must point to the same memory address carrying information about the tag identified by the pseudonym ψ_i.

In order for tags to be identified uniquely, the hash values in the table of Figure 2 *must be distinct*. This can be achieved by choosing the hash function, h, to be an expansion function, as opposed to the usual use of hash functions as compression functions, so that collision will occur with small probability.[1] We will assume that the output of the hash function has length L bits, which must be at least equal to $\lceil \log_2 NC \rceil$ so that the table in Figure 2, which is of size NC, can be constructed without collisions (L will be much larger in practice). If a pseudonym that causes a collision in Figure 2 is found, the pseudonym is replaced by another one that does not cause a collision. (Observe that the pool of possible pseudonyms is of size 2^ℓ, which is much larger than the required number of pseudonyms N, giving the system designer a sufficient degree of freedom in constructing the system.) With the appropriate choice of the hash function, a table of hash values with no collisions can be constructed. *Note that this operation is performed only once during the initialization phase, thus, it does not undermine the performance of the system.*

Since the length of $h(\psi_i, c)$ (the tags' identifiers), L, is large to avoid collision, it would be infeasible to have a physical storage that can accommodate all possible L-bit strings (for direct addressing). For example, if $L = 128$, a database of size in the order of 4×10^{28} *Gigabyte* will be required. Previously proposed privacy-preserving schemes solve this problem in one of two approaches. The first approach requires $O(N_T)$ memory space to store information about each tag in the system, and requires the reader to perform a linear search among tags in the system to identify tags' responses; thus requiring $O(N_T)$ space and $O(N_T)$ time for identification. The other method identifies tags based on their key information and requires the reader to perform logarithmic search to identify tags' responses; thus requiring $O(N_T)$ space and $O(\log N_T)$ time for identification.

3) For ease of presentation, we will divide the database into three logical parts, M-I, M-II, and M-III. The first part, M-I, consists of a single table of size $O(2^n)$.The second

[1]For example, this can be accomplished by concatenating multiple hash functions, i.e., $h(x) = h_1(x) \| \cdots \| h_m(x)$, so that $h(x)$ has the required length.

$h(\psi_1, 0)$	$h(\psi_1, 1)$	$\cdots$	$h(\psi_1, C-1)$
$h(\psi_2, 0)$	$h(\psi_2, 1)$	$\cdots$	$h(\psi_2, C-1)$
$\vdots$	$\vdots$		$\vdots$
$h(\psi_N, 0)$	$h(\psi_N, 1)$	$\cdots$	$h(\psi_N, C-1)$

Figure 2. During database initialization, all values of $h(\psi, c)$ are computed.

part, M-II, consists of multiple smaller tables; the total size of all the tables in M-II is $O(NC)$. Finally, the last part, M-III, is of size $O(N)$.

The table in M-I is a table of pointers. The addresses of M-I range from 0^n to 1^n; each entry in the table points to the head of one of the mini tables in M-II (according to a specific relation explained below).

Each entry of M-II contains two fields. In the first field, the hash values obtained in the table of Figure 2 are stored (i.e., $h(\psi_i, c)$ for all $i = 1, \dots, N$ and all $c = 0, \dots, C-1$). M-II is organized based on the hash values stored in the first field. We say that two hash values $h(\psi_1, c_1)$ and $h(\psi_2, c_2)$ are in the same *position*, b, if their n most significant bits are the same (recall that the output length of the hash function is $L > n$). All hash values that have the same position, i.e., share the n most significant bits, are stored in the same *mini* table in M-II (e.g., the hash values with $b = s$ in Figure 3). Hash values with distinct positions are stored in different tables (e.g., hash values with $b = 0^n, s, 1^n$ in Figure 3). (Recall that Figure 2 contains the computed hash values; hence, table M-II can be viewed as a reorganized version of the two-dimensional table in Figure 2 into a one-dimensional table of size $O(NC)$.) The second field of each entry of M-II stores a pointer to an entry in M-III containing information about a tag in the system (depending on the value of the first field). For example, if the value stored in the first field is $h(\psi_i, c)$, then the value in the second field will be a pointer to the data entry in M-III where information about the tag with pseudonym ψ_i can be found.

After M-II has been constructed, the pointers at M-I are chosen to satisfy the following: the pointer stored at address a in M-I must point to the mini table in M-II that stores identifiers with position a. In other words, each pointer in M-I must point to the identifiers with position equal to the address of the pointer.

Finally, M-III is the actual storage where tags' information is stored. Figure 3 depicts the architecture of the database with the three logical partitions. The identification phase below will further illustrate the structure of the database.

4.5. Tag Identification

Tags in a protocol run of the system are identified by the hash of their pseudonyms concatenated with their internal counters. Denote by $\Psi_{i,c}$ the hash value of the i^{th} pseudonym concatenated with a counter c; that is, $\Psi_{i,c} := h(\psi_i, c)$. Furthermore, we will denote by

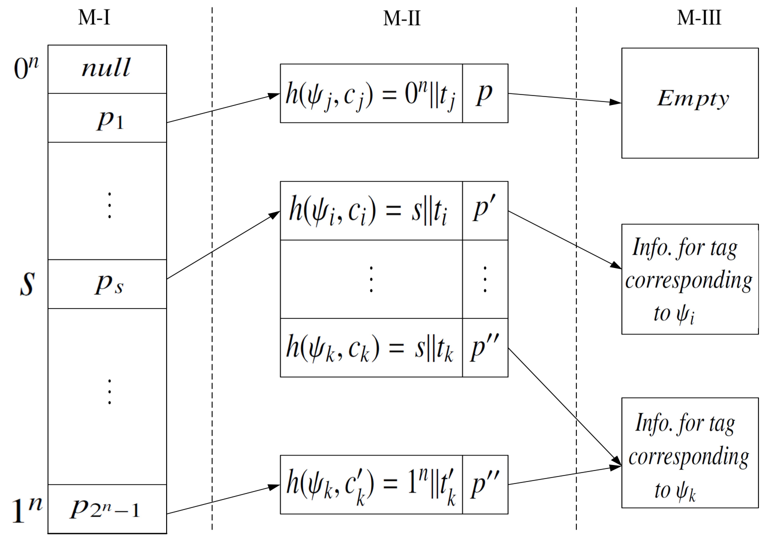

Figure 3. The architecture of the database. Each entry in M-I points to another, smaller table in M-II. The entries of the smaller tables in M-II point to tags' information.

$\Psi^n_{i,c}$ the truncated value of $\Psi_{i,c}$; more precisely, $\Psi^n_{i,c}$ represents the n most significant bits of $\Psi_{i,c}$ (i.e., the position of $\Psi_{i,c}$).

Once $\Psi_{i,c}$ has been received, the reader accesses the data entry at address $\Psi^n_{i,c}$ in M-I. This table entry is actually a pointer, p, to one of the tables in M-II. There are three possible scenarios here:

1) The value at address $\Psi^n_{i,c}$ in M-I is a *null*. This implies that, during the construction of the table in Figure 2, no identifier with position $\Psi^n_{i,c}$ is constructed. Therefore, either the tag is not a valid one or the tag's response has been modified. In the example of Figure 3, if the n most significant bits of the received $\Psi_{i,c}$ are *zeros*, then no valid tag matches this response.

2) The pointer, p, at address $\Psi^n_{i,c}$ points to a table in M-II with exactly one entry. In this scenario, the first field of the entry pointed at by p must be the entire (untruncated) $\Psi_{i,c}$; the value at the second field will be a pointer to the entry in M-III that contains information about the interrogated tag. In the example of Figure 3, if the n most significant bits of the received $\Psi_{i,c}$ are *ones*, then the pointer at address 1^n in M-I will point to the entry at M-II at which $\Psi_{k,c'_k} = 1^n||t'_k$ and the pointer, p'', are stored. In turn, p'' will point to the entry at M-III where information about the tag with pseudonym ψ_k is stored.

3) The pointer at address $\Psi^n_{i,c}$ of M-I points to a table in M-II with more than one entry. In this scenario, the reader searches the first fields of the mini table in M-II until it reaches the entry that matches the complete (untruncated) received identifier, $\Psi_{i,c}$; and then follows the pointer (in the corresponding second field) to get the tag's information. In the example of Figure 3, if the received identifier is $\Psi_{k,c_k} = s||t_k$, the reader will follow the pointer at address s of M-I. The pointer, however, points to a table in M-II with more than one entry. Therefore, the reader must search until it reaches the last entry of the table to find a match for the received $\Psi_{k,c_k} = s||t_k$. Once the match is found, the reader can follow the pointer, p'', to the entry in M-III containing information about the tag with pseudonym ψ_k.

The identification process allows for unique identification of tags in the system. This is due to the requirement that, in the initialization phase, the values in the table of Figure 2 are distinct. Consequently, the entries in M-II are distinct, allowing for the unique identification of tags.

Remark 1 Recall that the pseudonyms drawn in the initialization are not publicly known. If the pseudonyms were published, an adversary can, in principle, construct her own system and identify tags in constant-time. Further discussion about the adversary's ability to expose secret pseudonyms is provided in Section 8.

4.6. Identity Randomization and System Update

Once a tag has been authenticated, the reader draws one of the unoccupied pseudonyms generated in the initialization phase. (Recall that the number of pseudonyms is greater than the number of tags in the system; consequently, there will always be unused pseudonyms available for identity randomization.) Once an unoccupied pseudonym has been chosen, it is to be transmitted to the tag in a secret and authenticated way.

To allow for correct identification of a tag after its pseudonym has been updated, the database must be updated accordingly. A straightforward way of updating the database is by updating the pointers corresponding to the outdated and updated pseudonyms. For example, if the tag's outdated pseudonym is ψ_i and its updated pseudonym is ψ_k, then all pointers in M-II corresponding to entries $\Psi_{i,0}, \Psi_{i,1}, \ldots, \Psi_{i,C-1}$ must point to a null; and all pointers in M-II corresponding to entries $\Psi_{k,0}, \Psi_{k,1}, \ldots, \Psi_{k,C-1}$ must point to the entry in M-III containing information about the tag. This method, however, requires $O(C)$ updates.

An alternative method that allows a faster update is depicted in Figure 4. Instead of updating the pointers as in the previous method, the tag's information is moved to the entry in M-III pointed at by the pointers corresponding to the updated pseudonym in M-II. The only price to pay for this method over the previous one is that the size of M-III will increase from $O(N_T)$ to $O(N)$ (asymptotically, N and N_T are of the same size). In the example of Figure 4, instead of changing all entries in M-II with pointer p' to p, and changing entries with pointer p to *null*, the tag's information is moved to the entry in M-III pointed at by p' and the entry pointed at by p is emptied.

5. Performance Analysis

For the proposed scheme to be practical, we must show that a set of parameters can be chosen such that our claim of constant-time identification can be achieved with feasible resources (namely, feasible database size). This section is devoted to showing that, with a set of appropriately chosen parameters, the proposed technique can achieve constant-time identification with a database of size $O(N_T)$.

Assuming that the $\Psi_{i,c}$'s are uniformly distributed, the probability that the truncated version $\Psi^n_{i,c}$ takes a specific value, s, is $\alpha = \Pr(\Psi^n_{i,c} = s) = 2^{-n}$, for any $s \in \{0, 1\}^n$. Let

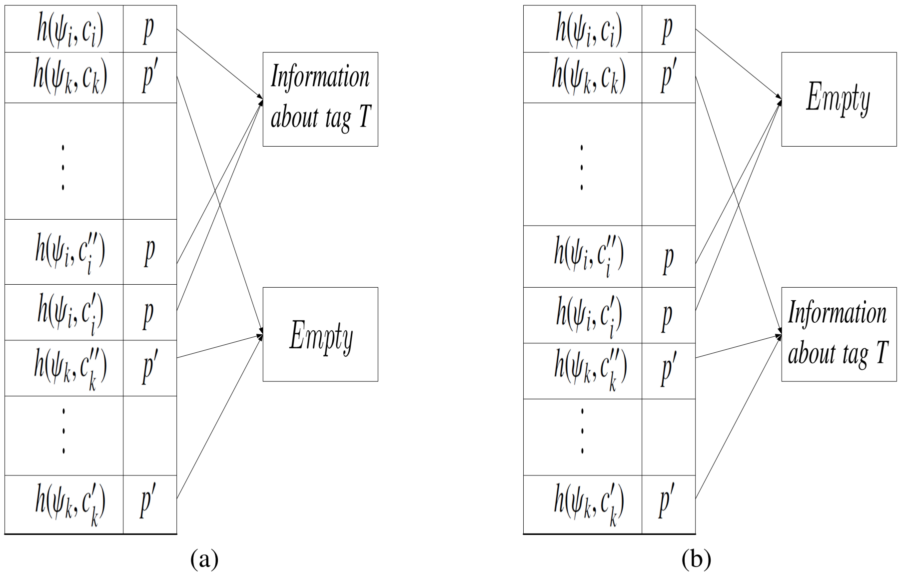

Figure 4. (a) Before (b) After; an illustration of database update. Note that only the tag information is updated, rather than the pointer values. This way, we only have to update two entries instead of $O(C)$ entries.

$M := NC$ and define $m := \log_2 M$, where N is the total number of pseudonyms and C is the maximum counter value. Then, out of the M values of $\Psi_{i,c}$'s, the probability that exactly k of them share the same truncation value (i.e., exactly k of them have the same n most significant bits) is

$$\Pr(\boldsymbol{k} = k) = \binom{M}{k} \alpha^k (1-\alpha)^{M-k}, \tag{2}$$

where $\boldsymbol{k}$ is the random variable representing the number of $\Psi^n_{i,c}$ sharing the same value, s, for any $s \in \{0,1\}^n$. Then, for $k \ll M$,

$$\binom{M}{k} = \frac{M!}{k!(M-k)!} \approx \frac{M^k}{k!}. \tag{3}$$

Using the facts that $\lim_{n\to\infty}(1 - \frac{1}{n})^n = e^{-1}$, $M = 2^m$, and $\alpha = 2^{-n}$ we get:

$$\begin{aligned}(1-\alpha)^{M-k} &\approx (1-\alpha)^M &(4)\\ &= (1-2^{-n})^{2^m} &(5)\\ &= (1-2^{-n})^{2^n \cdot 2^{m-n}} &(6)\\ &\approx e^{-2^{m-n}}. &(7)\end{aligned}$$

Substituting equations (3) and (7) into (2) yields,

$$\Pr(\boldsymbol{k} = k) \approx \frac{M^k}{k!} \cdot \alpha^k \cdot e^{-2^{m-n}} \quad (8)$$

$$= \frac{2^{mk}}{k!} \cdot \frac{1}{2^{nk}} \cdot e^{-2^{m-n}} \quad (9)$$

$$= \frac{1}{k!} \cdot \beta^k \cdot e^{-\beta}, \quad (10)$$

where $\beta = 2^{m-n}$. Choosing $m = n$ yields $\beta = 1$ and equation (10) can be reduced to

$$\Pr(\boldsymbol{k} = k) \approx \frac{1}{k!} \cdot e^{-1} \text{ for } k = 0, 1, \dots . \quad (11)$$

It can be easily verified that $\Pr(\boldsymbol{k} = k)$ in equation (11) is a valid probability mass function by verifying that $\sum_{k=0}^{\infty} \Pr(\boldsymbol{k} = k) = 1$.

Using the fact that $e = \sum_{k=0}^{\infty} \frac{1}{k!}$, the expected number of truncated $\Psi_{i,c}$'s with the same value is

$$E[\boldsymbol{k}] = \sum_{k=0}^{\infty} k \cdot \Pr(\boldsymbol{k} = k) \quad (12)$$

$$= \sum_{k=1}^{\infty} k \cdot \frac{1}{k!} \cdot e^{-1} \quad (13)$$

$$= 1. \quad (14)$$

Recall that identifiers $\Psi_{i,c}$ with the same truncated value $\Psi_{i,c}^n$ will be in the same table in M-II; and when the reader receives one of these identifiers it will have to search the table to be able to identify the tag. Equation (14), however, implies that the expected size of the tables in M-II is *one*. Therefore, upon receiving a tag identifier $\Psi_{i,c}$, the reader goes to the table entry in M-I at address $\Psi_{i,c}^n$, follows the pointer p_1 stored at that address, searches the table in M-II pointed at by p_1 for the received $\Psi_{i,c}$ (on average there will be only one entry by (14)), and then follows a pointer p_2 to information about the tag. Indeed, the search time is independent of the number of tags in the system (on average).

Since the database consists of three parts, M-I, M-II, and M-III; and since the size of M-I is $O(2^n)$, the size of M-II is $O(NC)$, and the size of M-III is $O(N)$, the only concern is the size of M-I. The above analysis shows that, by choosing $n = \lceil \log_2 NC \rceil$, the system achieves the constant-time identification claim. Therefore, the size of M-I is $O(NC)$ and, consequently, the total size of the database is $O(NC)$. However, C is a constant, independent from the number of tags in the system; and N is $O(N_T)$. Therefore, with the proposed system, the required size of the database for constant-time identification to be achieved is $O(N_T)$.

6. Case Study

Since big O analysis can be impractical by absorbing big constants, we give here a numerical example of the practicality of our system. Assume an enterprise with one billion items to be tagged, i.e., $N_T = 10^9$. Assume further that the total number of pseudonyms is two billions, i.e., $N = 2N_T$ and $C = 1000$. Then, the truncated identifiers are $n = \lceil \log_2 NC \rceil = 41$-bit long. Therefore, M-I can be constructed with a storage smaller than 12 terabyte; a practical storage even for personal usage.[2]

Therefore, an active adversary must interrogate a tag more than 1000 consecutive times, not separated by a protocol run with a valid reader, in order to correlate its responses. Observe that, unlike security models in general computer communications, 1000 consecutive interrogations is an unlikely scenario for RFID systems. A web server, for instance, is always online. In a typical RFID systems, however, adversaries must be in close proximity to tags in order to interrogate them. Observe, moreover, that an adversary who is always in the vicinity of a tag can track it down visually without interrogation. So, in typical designs, the goal is protect tags privacy against adversaries that are not always in close proximity to the RFID tags. Therefore, limiting the number of consecutive tag interrogations is a typical relaxation in RFID models [38].

7. Security Analysis

In this section, we prove that our protocol preserves the integrity of the tag and reader while maintaining user privacy. Before we proceed with the proofs of privacy and integrity, we state some important assumptions about the used hash function that are necessary for our security proofs.

7.1. Cryptographic Hash Functions

We assume the use of a secure cryptographic one-way hash function (the Secure Hash Algorithm, SHA, family is a popular example that is accepted by the National Institute of Standards and Technology, NIST, as a standard [39]). Under practical assumptions about the adversary's computational power, the used hash function satisfies the following properties.

1. Given the output of the hash function, it is computationally difficult to infer the input. That is, given the value of $h(x)$, the probability to predict the correct value of x by computationally bounded adversaries is negligible.
2. Given x and $h(x)$, the probability to predict $h(x + i)$, for any i, without actually evaluating $h(x + i)$ is negligible.

Given the above properties of the used hash function, the following lemma states an important result that will be used for the privacy and integrity proofs.

Lemma 1 *The secret parameters of RFID tags in the proposed protocol cannot be exposed without calling the Reveal oracle.*

Proof: In any interrogation, the tag responds with its current identifier $\Psi_{i,c} = h(\psi_i, c)$, where ψ_i is the tag current pseudonym and c is its internal counter. Given the above properties of the used hash function, the pseudonym cannot be exposed by the observation of $h(\psi_i, c)$ with a non-negligible probability. Furthermore, the new pseudonym is delivered to the tag by transmitting $(h(2, \psi_i, k_i, \tilde{r}) \oplus \psi_{i+1})$, which can be viewed as an encryption of

[2]Western Digital has already released 8-TB hard drives for personal use [37].

ψ_{i+1} with the key $h(2, \psi_i, k_i, \tilde{r})$. Since ψ_i and k_i are unknown to adversaries, $h(2, \psi_i, k_i, \tilde{r})$ will act as a random key and the new pseudonym ψ_{i+1} will be delivered secretly. Moreover, since the outdated and the updated pseudonyms, ψ_i and ψ_{i+1}, are unknown to adversaries, the two identifiers, $h(\psi_i, c)$ and $h(\psi_{i+1}, c)$, cannot be correlated with a non-negligible probability and, similarly, the identifiers $h(\psi_i, c)$ and $h(\psi_i, c + 1)$, cannot be correlated with a non-negligible probability.

Therefore, unless $\mathcal{A}$ calls the *Reveal* oracle, no secret information about RFID tags in the proposed protocol can be revealed. ■

Before we proceed with the formal proofs, we discuss the effect of the *Block* oracle and desynchronization attacks.

7.2. Desynchronization Attacks

Jamming the communication channel, i.e., blocking all messages, is not of an interest to this work, since it does not lead to breaching of tags' privacy nor does it lead to authenticating unauthorized users.

Blocking the first message (from the reader to the tag) will just cause the tag not to respond. Similar to jamming, no information will be leaked by blocking the first message.

Blocking the second message (from the tag to the reader) can be modeled by the *Query* oracle. In fact, intercepting the tag's response is equivalent to a *Query* oracle in which the adversary does not control the value of r transmitted in the first message.

Blocking the last message (from the reader to the tag) has two effects. First, it will cause the tag to increase its internal counter (since the protocol run is incomplete), but this can also be modeled using the *Query* oracle. Second, and more important, it will lead the reader to update the tag's pseudonym while the tag has not,[3] i.e., a desynchronization attack. Fortunately, however, this can be solved by storing both the updated and the outdated pseudonyms in the database (the database must be designed accordingly, as detailed in Section 9).

In what follows, we formally prove the privacy and integrity of the proposed protocol.

7.3. Privacy

In this section, we show that the proposed protocol satisfies the three notions of tag privacy defined in Section 3.3.

Theorem 1 *In the proposed protocol, tags are universally untraceable.*

Proof: Assume the challenger C has chosen two tags, T_0 and T_1, and a reader R for the game. $\mathcal{A}$ starts the game by calling the *Query, Send, Execute* and *Block* oracles on T_0, T_1, and R for a number of times of its choice before deciding to stop. $\mathcal{A}$ records all the outputs of the oracle calls and notifies C.

Now, R carries out protocol runs with T_0 and T_1 causing their pseudonyms and keys to update. C chooses a bit b uniformly at random and sets $T = T_b$. By Lemma 1, $\mathcal{A}$ cannot infer the outdated nor the updated values of the tags' pseudonyms and keys. $\mathcal{A}$ now calls

[3]This is an inherited problem shared by all interactive protocols. The fundamental problem here is that the sender of the last message has no means of confirming that the message has been successfully delivered.

the oracles *Query, Send, Execute* and *Block* and outputs a bit b'. Since $\mathcal{A}$ does not know the outdated or the updated pseudonyms, by the assumptions on the used hash function, the probability $\Pr(b = b')$ will be greater than $1/2$ with a non-negligible probability.

Therefore, the adversary's advantage, as defined in equation 1, will be greater than zero with only a negligible probability. ■

The following theorem concerns forward untraceability in our protocol.

Theorem 2 *In the proposed protocol, tags are forward untraceable.*

Proof: Similar to the proof of universal untraceability, assume the challenger C has chosen two tags, T_0 and T_1, and a reader R for the game. $\mathcal{A}$ starts the game by calling the *Query, Send, Execute* and *Block* oracles on T_0, T_1, and R for a number of times of its choice before deciding to stop. $\mathcal{A}$ records all the outputs of the oracle calls and notifies C.

Now, R carries out protocol runs with T_0 and T_1 causing their pseudonyms and keys to update. C chooses a bit b uniformly at random and sets $T = T_b$ and gives it to $\mathcal{A}$. By Lemma 1, $\mathcal{A}$ cannot infer the outdated nor the updated values of the tags' pseudonyms and keys. $\mathcal{A}$ now calls the *Reveal*(T) oracle, thus getting T's secret parameters, and then outputs a bit b'. Since $\mathcal{A}$ cannot infer the outdated pseudonyms and keys of T_0 and T_1 from the recorded oracle outputs, and since the updated pseudonyms are chosen independently of the outdated ones, $\mathcal{A}$ cannot correlate T's updated pseudonym with its previous responses. Furthermore, since the updated key is a hashed function of the outdated key, by the assumptions on the used hash function, $\mathcal{A}$ cannot infer the value of the outdated key with a non-negligible probability. Hence, the probability $\Pr(b = b')$ will be greater than $1/2$ with only a non-negligible probability.

Therefore, the adversary's advantage, as defined in equation 1, will be greater than zero with only a negligible probability. ■
Finally, the following theorem concerns existential untraceability in our protocol.

Theorem 3 *Without being able to achieve mutual authentication with an authorized reader, a tag interrogated fewer than C number of times by an active adversary is untraceable.*

Proof: Assume that C has given T_0 and T_1 to $\mathcal{A}$. Let ψ_0 and ψ_1 denote the pseudonyms of T_0 and T_1, respectively. Without loss of generality, assume that tags T_0 and T_1 have their internal counters at zero. $\mathcal{A}$ calling the *Query* oracle on T_0 and T_1 for m and n times, respectively, where $m, n < C$ will observe the following sequences

$$\{h(\psi_0, 0), \ldots, h(\psi_0, m-1)\}, \tag{15}$$

$$\{h(\psi_1, 0), \ldots, h(\psi_1, n-1)\}. \tag{16}$$

The challenger C now chooses a bit b at random, sets $T = T_b$, and gives T to $\mathcal{A}$. By interrogating the tag, $\mathcal{A}$ gets an identifier $h(\psi_b, \ell)$, where $b \in \{0, 1\}$ and $\ell \in \{m, n\}$. Again, by Lemma 1, ψ_0 and ψ_1 cannot be recovered by the observation of the sequences in equations (15) and (16). Furthermore, by the assumptions on the hash function, $h(\psi_0, m)$ and $h(\psi_1, n)$ cannot be correlated to the observed values in equations (15) and (16) with a non-negligible probability. Therefore, the probability that $\mathcal{A}$'s guess b' is equal to b can be higher than $1/2$ with only a negligible probability and, hence, $Adv_{\mathcal{A}} = 0$ and tags are existentially untraceable, provided that $m, n < C$. ■

7.4. Mutual Authentication

We shift our attention now to the other security requirement, authenticity.

Theorem 4 *The proposed protocol performs secure mutual authentication.*

Proof: Assume that C has given $\mathcal{A}$ a tag T and a reader R. Assume further that $\mathcal{A}$ has called the *Query, Send, Execute* and *Block* oracles for a number of times of its choice and recorded the oracle outputs.

The first condition of Definition 4 of secure mutual authentication is satisfied by Lemma 1.

Assume now that $\mathcal{A}$ attempts to impersonate the tag T. $\mathcal{A}$ must answer the reader's challenge r with a response $s = \big(h(\psi, c), \tilde{r} = h(0, \psi, c, k, r)\big)$, where ψ is the tag's current pseudonym and k is its key. Since ψ and k remain secret, by Lemma 1, $\mathcal{A}$ can be successful with only a negligible probability. Observe further that, even if $\mathcal{A}$ attempts to impersonate an arbitrary tag in the system (the one with pseudonym ψ), $\mathcal{A}$ must know the value of k corresponding to the tag with pseudonym ψ in order to be authenticated with a non-negligible probability. Therefore, the probability of impersonating a tag in the system is negligible.

On the other hand, assume that $\mathcal{A}$ attempts to impersonate the reader R. $\mathcal{A}$ sends r to the tag and receives $h(\psi, c)$ and $\tilde{r} = h(0, \psi, c, k, r)$, where ψ is the tag's pseudonym, k is its secret key, and c is its internal counter. Since, by the assumption on the hash function, $\mathcal{A}$ cannot infer the secret parameters, the probability of coming up with a response that will be equal to $h(1, \psi, k, \tilde{r})$ is negligible. Consequently, the probability of impersonating an authorized reader in the system is negligible.

Therefore, the probability of mutual authentication when the protocol is not honest is negligible and, hence, the second condition of Definition 4 of secure mutual authentication is satisfied.

As shown above, the adversary's probability of causing a desynchronization between the tag and the reader by authenticating herself to either one of them is negligible. Causing a desynchronization by blocking the last message of the protocol can be solved by making the reader store both the updated and the outdated values (as will be discussed in Section 9). Therefore, if the protocol run is honest, mutual authentication will be achieved with probability one and, consequently, the third condition of Definition 4 of secure mutual authentication is satisfied.

Hence, all conditions of Definition 4 of secure mutual authentication are satisfied and the proposed protocol is shown to provide secure mutual authentication. ■

8. Tag Compromise Analysis

In this section we describe a vulnerability to tag compromise, modify the adversarial model to capture this vulnerability, analyze our system using the modified model, and propose countermeasures to mitigate tag compromise attacks.

8.1. The Compromise attack

Each tag in the proposed protocol has two pieces of secret information, its pseudonym and its key. Since tags' pseudonyms and keys are designed to be statistically independent

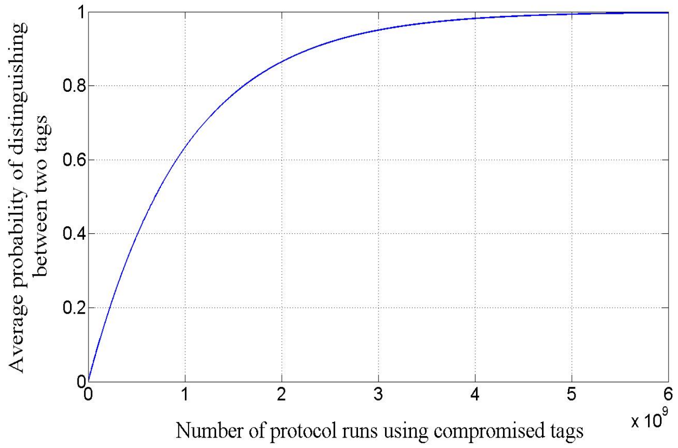

Figure 5. The adversary's average probability of distinguishing between two tags vs. the number of protocol runs using a compromised tag, in a system with 2×10^9 pseudonyms.

for different tags, compromising some tags in the system does not affect the security of other, uncompromised tags. An adversary, however, can compromise a tag in the system and attempt to harvest as many pseudonyms as possible by performing multiple protocol runs with a valid reader.

The adversarial model of Section 3 can be modified to capture the tag compromise attack. Let an adversary calling the *Reveal* (T) oracle, thus capturing the tag T, have the ability to perform multiple protocol runs with the system. Let q be the number of protocol runs an adversary has performed with the system using compromised tags. The number of interest here is how many distinct pseudonyms the adversary has collected, after q protocol runs. This is known in the literature of probability theory as the "coupon collecting problem" [40]. Given there are N distinct pseudonyms and the adversary has performed q protocol runs, assuming each pseudonym is equally likely to be selected, the expected number of distinct pseudonyms collected by the adversary is [40]:

$$N\Big(1 - (\frac{N-1}{N})^q\Big). \tag{17}$$

Assume an adversary has built a system, similar to our construction, with the collected pseudonyms. The adversary's advantage of distinguishing between two tags, given by equation (1), will be greater than zero if at least one of the two tags' pseudonyms is in the constructed table. Thus, given the adversary has performed q protocol runs with a system of N pseudonyms, the probability of distinguishing between two tags is:

$$1 - \Big(\frac{N-1}{N}\Big)^{2q}. \tag{18}$$

Consider the numbers given in Section 6, i.e., $N = 2\times10^9$. To have a 0.001 probability of distinguishing between two tags, an adversary needs to compromise a tag and complete more than a million protocol runs with the system. Figure 5 shows the adversary's probability of having an advantage greater than zero as a function of the number of protocol runs performed with the system using compromised tags.

8.2. Countermeasures

Remember, however, that the database is a powerful device. Therefore, designing the database to record timing information about the tag's past protocol runs can mitigate this threat. For example, the database can store information about the tag's last five protocol runs (this can be stored as part of the tag's information, i.e., in M-III). If the adversary attempts to harvest different pseudonyms by performing multiple protocol runs with the system, the tag will be detected. Therefore, to harvest enough pseudonyms, the adversary will need to compromise more than one tag, depending on the system's parameters and the required probability of success.

Furthermore, the database can periodically update the system by replacing vacant pseudonyms with new pseudonyms (recall that the number of pseudonyms in the database, N, is only a small fraction of the number of all possible pseudonyms, 2^{ℓ}). This pseudonym update procedure is performed offline by the database, thus, not affecting identification time. Moreover, as a result of the independence of secret parameters amongst tags, the updating procedure is independent of tags.

With the periodic update described earlier, the space of possible pseudonyms will increase to all possible ℓ-bit long strings, as opposed to the predefined *smaller* number N. Therefore, for a bounded adversary, any polynomial number of collected pseudonyms is negligible in the security parameter ℓ. (Recall that the size of the actual database is still proportional to N; only from the adversary's point of view the size is proportional to 2^{ℓ}.) Consequently, the adversary's probability of breaking the privacy of the system is negligible in ℓ, provided the periodic update of the database.

9. Preventing Desynchronization Attacks

Recall that if the tag does not accept the reader's response, the database will update the tag's pseudonym while the tag has not. Consequently, the reader will not be able to identify the tag in future protocol runs. As mentioned in Section 7.2, however, the database can be designed to overcome such attacks by storing both the updated and outdated pseudonyms; details are as follows.

9.1. Redesigning the Update Procedure

Consider the update procedure described in Section 4.6. Let each entry of M-III consists of a linked list data structure, as opposed to a single entry as in the basic description. For illustration purposes, assume the linked list consists of four fields containing the following data. The first field contains information about a tag T_i with pseudonym ψ_i, where ψ_i is the T_i's "updated" pseudonym. The second field will contain a pointer to the entry of M-III corresponding to T_i's outdated pseudonym, if it existed (i.e., if the tag has been interrogated previously). The third field contains information about a tag T_k with pseudonym ψ_k, where ψ_k is the T_k's "outdated" pseudonym. The fourth field will contain a pointer to the entry of M-III corresponding the T_k's updated pseudonym, if it existed. The construction is best illustrated through the following example.

Consider Figure 6 for updating the database. Assume the reader has authenticated the tag T_1 with a current pseudonym ψ_i. Assume further that the database returns a new

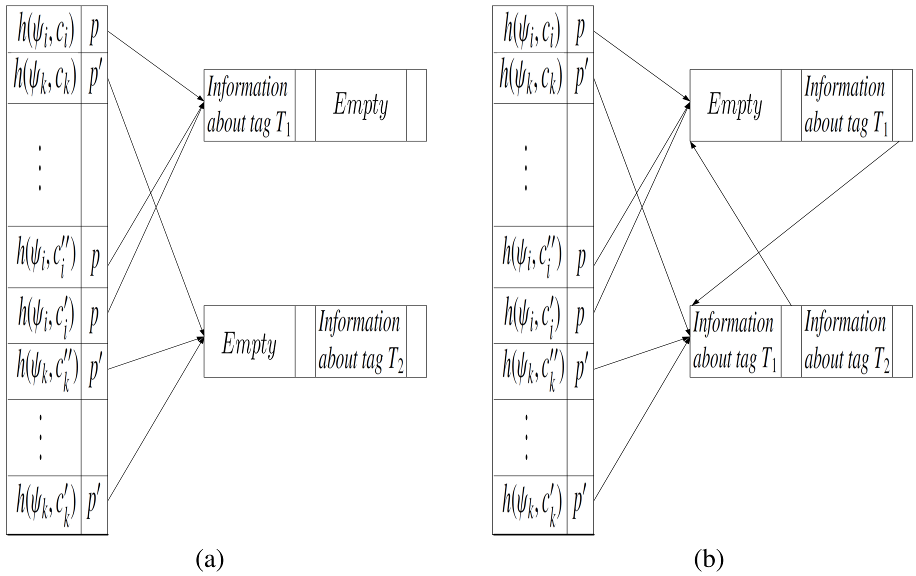

Figure 6. (a) Before (b) After; an illustration of database update. Note that only the tag information is updated, rather than the pointer values. This way, we only have to update two entries instead of $O(C)$ entries.

pseudonym ψ_k as the updated pseudonym for the tag T_1. Just like the update procedure described in Section 4.6, the information about tag T_1 in M-III will be copied into the data entry pointed at by the pointers in M-II corresponding to the updated pseudonym ψ_k (i.e., pointer p' in the example of Figure 6). However, instead of deleting the information about tag T_1 in the entry pointed at by the pointer corresponding to the outdated pseudonym ψ_i (i.e., pointer p in the example of Figure 6), the information remains there.

Observe, however, that by continuing in this fashion, information about the tag will have multiple copies in the database, one for each identification run. To prevent this problem, we use the pointer field in M-III. That is, the use of the new pointer fields in M-III will allow preventing the desynchronization attack with only two copies of tag information in M-III, one corresponding the updated pseudonym and one corresponding to the outdated pseudonym. Observe, in Figure 6-b, that the information about tag T_1 corresponding to the outdated pseudonym ψ_i is followed by a pointer field that stores a pointer to the information about T_1 corresponding to the updated pseudonym ψ_k. Similarly, that the information about tag T_1 corresponding to the updated pseudonym ψ_k is followed by a pointer field that stores a pointer to the information about T_1 corresponding to the outdated pseudonym ψ_i.

Assume now that the tag T_1 has received the updated identifier ψ_k successfully and, hence, no desynchronization attack has been attempted. Upon interrogation, the tag will respond with its identifier $\Psi_{k,c}$, which will enable the reader to identify the tag. Once the entry in M-III with information about the tag has been found (the bottom box of M-III in the example of Figure 6-b), the pointer in the field after the tag's information is followed to empty the data entry with information corresponding to the tags outdated pseudonym ψ_i (in the top box of M-III in the example of Figure 6-b). The database then draws an unused pseudonym ψ_j to update the tag, mark the information corresponding to ψ_k as outdated, and copies the tag's information to the entry corresponding to ψ_j. Therefore, only two copies of the tag's information need to be stored in M-III.

On the other hand, assume that there has been a desynchronization attempt during the last protocol run and, thus, the tag has not updated its pseudonym to ψ_k. Therefore, upon the next interrogation, the tag will respond with its identifier $\Psi_{i,c}$. Since both the updated and the outdated pseudonyms are stored, the database can still identify the tag via its outdated pseudonym (in the top box of M-III in the example of Figure 6-b). Once the tag's information has been found, the pointer is followed to delete the tag's information corresponding to the undelivered pseudonym ψ_k (in the bottom box of M-III in the example of Figure 6-b). Just like the previous case, the database then draws an unused pseudonym ψ_j to update the tag, mark the information corresponding to ψ_i as outdated, and copies the tag's information to the entry corresponding to ψ_j. Therefore, whether a desynchronization attack has been attempted or not, only two copies of the tag's information need to be stored in M-III.

As can be observed in the example of Figure 6-a, the tag T_2 has ψ_k as its outdated pseudonym. This does not prevent the database from choosing ψ_k as the new pseudonym to update tag T_1. If the existence of a tag with an outdated pseudonym prevents the database from using this pseudonym to update other tags, then each tag in the system will occupy two pseudonyms. As this might not cause a problem when the number of tags in the system is not too large, it can be problematic if the number of tags in the system is very large (a billion tags, for instance). Therefore, we allow the database to update tags with any pseudonym as long as there is no other tag in the system with this pseudonym as its "updated" pseudonym, even if other tags have this pseudonym as their "outdated" pseudonym. In the example of Figure 6, ψ_k is chosen to update the tag T_1 even though the tag T_2 has the same pseudonym as its outdated pseudonym.

Assume now that the tag T_1 in the example of Figure 6 has received ψ_k successfully. Since in the next interrogation, T_1 will respond with $\Psi_{k,c}$, the pseudonym ψ_k will be marked now as the tag's outdated pseudonym. Therefore, there might be more than one tag with the same outdated pseudonym and, hence, upon receiving an identifier corresponding to such pseudonym, the database will search linearly (amongst tag stored in the same entry of M-III) until it finds the match. We show next that this does not violate the constant-time identification claim by showing that the expected number of tags in the same entry of M-III is independent of the total number of tags in the system.

9.2. Identification Complexity

We seek to find the number of tags with the same outdated pseudonym, thus, falling in the same entry of M-III, causing the database to search linearly amongst them. Recall that pseudonyms are drawn uniformly at random to update tags. That is, the tag's information can fall into any entry of M-III with equal probability. This problem is equivalent to a well-studied problem in probability theory called the "balls in bins" problem [41]. In a classic variant of the balls in bins problem, m balls are thrown at n bins and the probability of any ball falling in a certain bin is the same for all balls and all bins.

Instead of m balls and n bins, we are interested in throwing N_T RFID tags into N possible pseudonyms (recall that each entry in M-III corresponds to one pseudonym). Therefore, the probability that a certain tag will fall into a particular entry in M-III is $1/N$. Consequently, the expected number of tags that will fall in a particular entry of M-III is

$\sum_{i=1}^{N_T} \frac{1}{N} = \frac{N_T}{N}$. Since $N > N_T$ by design, the expected number of outdated information in a single entry of M-III is less than one. Therefore, given the redesigned updating procedure described in Section 9.1 to prevent desynchronization attacks, the identification complexity of the proposed protocol is constant.

10. Conclusion

In this paper, we addressed the problem of individual tag identification in large-scale RFID systems. We proposed a protocol that enables the private identification of tags in the system with constant-time complexity. By utilizing the existence of a large storage device in the system, the constant-time identification is achieved by performing the necessary time consuming computations offline (independent of the reader-tag interactions). As opposed to tree based protocols, the proposed protocol does not further complicate the already challenging problems in RFID systems, namely, collision avoidance and medium access control. Furthermore, tag compromise threats can be mitigated by periodically updating the database which, due to independence of secret parameters amongst tags, can be performed independent of any tag-reader interaction. To the best of our knowledge, this is the first symmetric-key, constant-time identification protocol in the literature of RFID that allows for secure mutual authentication and private identification.

References

[1] S. Garfinkel, A. Juels, and R. Pappu, "RFID Privacy: An Overview of Problems and Proposed Solutions," *IEEE SECURITY & PRIVACY*, 2005.

[2] A. Juels and R. Pappu, "Squealing euros: Privacy protection in RFID-enabled banknotes," in *Financial Cryptography – FC'03*, ser. Lecture Notes in Computer Science, R. N. Wright, Ed., vol. 2742, IFCA. Le Gosier, Guadeloupe, French West Indies: Springer-Verlag, January 2003, pp. 103–121.

[3] G. Avoine, "Privacy issues in RFID banknote protection schemes," in *International Conference on Smart Card Research and Advanced Applications-CARDIS'04*. IFIP, 2004.

[4] M. Burmester, B. De Medeiros, and R. Motta, "Anonymous RFID authentication supporting constant-cost key-lookup against active adversaries," *International Journal of Applied Cryptography*, vol. 1, no. 2, pp. 79–90, 2008.

[5] B. Liang, Y. Li, C. Ma, T. Li, and R. Deng, "On the Untraceability of Anonymous RFID Authentication Protocol with Constant Key-Lookup," *5th International Conference on Information Systems Security-ICISS'09*, pp. 71–85, 2009.

[6] 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*, 2003.

[7] M. Ohkubo, K. Suzuki, and S. Kinoshita, "Cryptographic Approach to "Privacy-Friendly" Tags," in *RFID Privacy Workshop*, 2003.

[8] G. Avoine, E. Dysli, and P. Oechslin, "Reducing time complexity in RFID systems," *Selected Areas in Cryptography–SAC*, vol. 3897, pp. 291–306, 2005.

[9] H.-Y. Chien, "SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity," *IEEE Transactions on Dependable and Secure Computing*, 2007.

[10] B. Song and C. J. Mitchell, "RFID Authentication Protocol for Low-cost Tags," in *ACM Conference on Wireless Network Security, WiSec'08*, 2008.

[11] D. Molnar and D. Wagner, "Privacy and security in library RFID: issues, practices, and architectures," *11th ACM conference on Computer and communications security*, 2004.

[12] L. Lu, J. Han, L. Hu, Y. Liu, and L. Ni, "Dynamic Key-Updating: Privacy-Preserving Authentication for RFID Systems," *Fifth IEEE International Conference on Pervasive Computing and Communications*, 2007.

[13] W. Wang, Y. Li, L. Hu, and L. Lu, "Storage-Awareness: RFID Private Authentication based on Sparse Tree," in *Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, SECPerU 2007.*, 2007.

[14] L. Lu, J. Han, R. Xiao, and Y. Liu, "ACTION: Breaking the Privacy Barrier for RFID Systems," *INFOCOM 2009. The 28th IEEE Conference on Computer Communications.*, 2009.

[15] G. Avoine and P. Oechslin, "RFID Traceability: A Multilayer Problem," in *Financial Cryptography – FC'05*, ser. Lecture Notes in Computer Science, A. Patrick and M. Yung, Eds., vol. 3570, IFCA. Roseau, The Commonwealth Of Dominica: Springer-Verlag, February–March 2005, pp. 125–140.

[16] J. Myung and W. Lee, "An adaptive memoryless tag anti-collision protocol for RFID networks," in *IEEE INFOCOM*, 2005.

[17] M. Kodialam and T. Nandagopal, "Fast and reliable estimation schemes in RFID systems," in *Proceedings of the 12th annual international conference on Mobile computing and networking, MobiCom*, 2006.

[18] S. Jain and S. Das, "Collision avoidance in a dense RFID network," in *Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization*. ACM, 2006, p. 56.

[19] G. Khandelwal, K. Lee, A. Yener, and S. Serbetli, "ASAP: a MAC protocol for dense and time-constrained RFID systems," *EURASIP Journal on Wireless Communications and Networking*, 2007.

[20] 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.

[21] P. Peris-Lopez, J. C. Hernandez-Castro, J. Estevez-Tapiador, and A. Ribagorda, "LMAP: A real lightweight mutual authentication protocol for low-cost RFID tags," Printed handout of Workshop on RFID Security – RFIDSec 06, Ecrypt, Graz, Austria, July 2006.

[22] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "EMAP: An efficient mutual authentication protocol for low-cost RFID tags." in *OTM Federated Conferences and Workshop-IS'06*, 2006.

[23] Q. Yao, Y. Qi, J. Han, J. Zhao, X. Li, and Y. Liu, "Randomizing RFID private authentication," in *IEEE International Conference on Pervasive Computing and Communications-PerCom'09*. IEEE Computer Society, 2009, pp. 1–10.

[24] B. Alomair, L. Lazos, and R. Poovendran, "Securing Low-cost RFID Systems: an Unconditionally Secure Approach," *The 2010 Workshop on RFID Security-RFIDsec'10 Asia available at http://www.ee.washington.edu/research/nsl/faculty/radha/ypublicationsmain.html*, 2010.

[25] A. Juels, R. Rivest, and M. Szydlo, "The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy," in *Conference on Computer and Communications Security – ACM CCS*, V. Atluri, Ed., ACM. Washington, DC, USA: ACM Press, October 2003, pp. 103–111.

[26] C. Floerkemeier, R. Schneider, and M. Langheinrich, "Scanning with a Purpose-Supporting the Fair Information Principles in RFID Protocols," *2nd International Symposium on Ubiquitous Computing Systems, Tokyo, Japan*, 2004.

[27] M. Rieback, B. Crispo, and A. Tanenbaum, "RFID Guardian: A battery-powered mobile device for RFID privacy management," *Proc. 10th Australasian Conf. on Information Security and Privacy (ACISP 2005*, vol. 3574, pp. 184–194.

[28] A. Juels, P. Syverson, and D. Bailey, "High-power proxies for enhancing RFID privacy and utility," *Proc. of the 5th Workshop on Privacy Enhancing Technologies*, 2005.

[29] A. Juels and S. Weis, "Defining Strong Privacy for RFID," in *International Conference on Pervasive Computing and Communications – PerCom 2007*, IEEE. New York City, New York, USA: IEEE Computer Society Press, March 2007, pp. 342–347.

[30] T. Lim, T. Li, and Y. Li, "A Security and Performance Evaluation of Hash-Based RFID Protocols," *4th International Conferences on Information Security and Cryptology-Inscrypt'08*, pp. 406–424, 2008.

[31] G. Tsudik, "YA-TRAP: Yet Another Trivial RFID Authentication Protocol," in *Proc. IEEE Intern. Conf. on Pervasive Computing and Communications (PerCom 2006)*. IEEE, 2006.

[32] C. Chatmon, T. van Le, and M. Burmester, "Secure anonymous RFID authentication protocols," *Florida State University, Department of Computer Science, Tech. Rep*, 2006.

[33] M. Conti, R. D. Pietro, L. V. Mancini, and A. Spognardi, "RIPP-FS: an RFID Identification, Privacy Preserving Protocol with Forward Secrecy," in *International Workshop on Pervasive Computing and Communication Security – PerSec 2007*, IEEE. New York City, New York, USA: IEEE Computer Society Press, March 2007, pp. 229–234.

[34] T. Dimitriou, "A secure and efficient RFID protocol that could make big brother (partially) obsolete," in *Fourth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom 2006.*, 2006.

[35] G. Avoine, "Adversary model for radio frequency identification," Technical Report, 2005.

[36] C. Ma, Y. Li, R. Deng, and T. Li, "RFID privacy: Relation Between Two Notions, Minimal Condition, and Efficient Construction," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 54–65.

[37] http://www.westerndigital.com/en/.

[38] A. Juels, "RFID security and privacy: A research survey," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 381–394, 2006.

[39] National Institute of Standards and Technology (NIST), "Secure Hash Standard," *FIPS PUB 180-3 Federal Information Processing Standards Publication http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf*, October 2008.

[40] S. Ross, *A First Course in Probability*. Prentice Hall, 2002.

[41] W. Feller, *An introduction to probability theory and its applications, vol 2*. Wiley India Pvt. Ltd., 2008.