A Hybrid RFID Protocol against Tracking Attacks

Jen-Chun Chang* Hsin-Lung Wu*

Abstract

We study the problem how to construct an RFID mutual authentication protocol between tags and readers. Juels (in Journal on Selected Areas in Communications, 2006) proposed an open question which asks whether there exists a fully privacy-preserving RFID authentication protocol in which the time complexity of a successful authentication phase can be done in sub-linear time $o(n)$ where n is the number of tags participating in the RFID system.

In this work, we answer this question positively in an amortized view. Precisely, we design a fully privacy-preserving protocol in which the amortized cost of a successful authentication phase only requires time $O(n^{3/4})$. In addition, our protocol is a hybrid from the randomized hash lock scheme of Weis et. al., the hash chain scheme of Ohkubo et. al., and the efficient identification scheme of Dimitriou. We combine them delicately to obtain the desired authentication protocol.

1 Introduction

RFID (Radio-Frequency Identification) is a technology in which one can identify objects or people by embedding tags, a small microchip capable of wireless data transmission. By tagging wares in shops, one can speed up the process of registration with wireless scanning. RFID tags have several characteristics. First of all, each tag has an identifier to represent itself. Moreover, such identifiers are long enough so that it has a unique code. In addition, each tiny tag is implanted within an object. Thus, finding the tag means discovering the corresponding object. Second, tag identification via radio frequency allows tagged objects to be read at a distance and in quite large numbers. These described characteristics introduce privacy issues, e.g. the traceability of the tags by illegitimate attackers.

One of important privacy issues is user privacy. Objects embedded with insecure tags may reveal private information as they are queried by readers. Another important privacy issue is location privacy. Objects even embedded with tags that do not reveal any sensitive information may also be tracked by the implanted tags. This is because the tag responses to the requesting readers are possible to help locate the tagged objects by analyzing the data from either legitimate or illegitimate readers in a fixed location. This may cause objects to reveal their private data such as their identifications in the future. The other important privacy issue is the so-called forward privacy defined as follows. At a time t, assume that the adversary is given all communication information between tags and readers and given all information stored in a compromised tag. The forward privacy requires the adversary not to be able to trace any past communication between the compromised tag and readers at a time $t' < t$. For more privacy definitions, we refer readers to the seminal survey of Juels [3].

*Department of Computer Science and Information Engineering, National Taipei University, Taipei County 237, Taiwan. E-mail: {jcchang,hsinlung}@mail.ntpu.edu.tw. This work was supported in part by the National Science Council of Taiwan under contracts NSC-97-2218-E-305-001-MY2 and NSC-97-2218-E-305-001-MY2.

2 Related Work

Many RFID protocol have been proposed to address the privacy issues. Weis et. al. proposed the so-called Hash Lock scheme [7] in which they used the hash function h to hash a random key K and the resulting value is used as the tag's $metaID$, that is, $metaID = h(K)$. When a reader queries a tag, the tag responds its $metaID$ and the reader sends $metaID$ to the back-end database (usually implemented by a hash table) to look up the appropriate key K' such that $h(K') = h(K)$. Once the reader finds the key K', it transmits it to the communicated tag and the tag hashes the key K' and compares it with $h(K)$. If the values match, the tag unlocks itself. Note that this scheme has a short searching time because the database is implemented by a hash table. Nevertheless, the tag implemented in this scheme can be tracked easily by an attacker via the corresponding $metaID$. Moreover, by eavesdropping the communication between the legitimate reader and the targeted tag, the attacker can capture the desired key K since K is sent in this insecure wireless channel.

In [7], Weis et. al. also gave a randomized version of Hash Lock scheme in which a tag hashes its ID with a long random string so that its output looks random and hence not fixed. This prevents the tag from being tracked. Precisely, when queried by the reader, the tag generates a random string R, computes the hash value $h(R||ID)$, and transmits $(R, h(R||ID))$ to the reader. Upon receiving $(R, h(R||ID))$, the reader starts a brute force search of stored IDs in the database until it finds one which matches the tag's challenging string $(R, h(R||ID))$. Although tags in this randomized scheme ensures the full privacy, it is not scalable for a tremendous number of tags since a huge number of hash operations must be performed at the back-end database. In addition, this protocol does not guarantee forward privacy since the stored information in a compromised tag reveals much information of previous communication of the same tag.

An extended version of randomized Hash Lock scheme that ensures forward privacy is proposed by Ohkubo et. al. [6]. To guarantee the forward privacy, their basic idea is to refresh the tag identifier each time the tag is queried by a reader. The scheme can be obtained via a low-cost hash chain mechanism. However, this scheme is also not scalable as it still requires exhaustive search. To reduce the searching complexity, Avoine and Oechslin [1] used the time-space memory tradeoff technique to save the search time. Although they obtained a sub-linear search time in terms of the number of tags, their proposed scheme is vulnerable in privacy as mentioned in [3, 4] and requires an extra counter to keep synchronization between an individual tag and the back-end database.

Dimitriou [2] used a mutual authentication scheme between tags and readers to settle the issue of desynchronization. His proposed scheme is a three-round protocol as follows. While a reader starts a request to the tag, it also appends a random string K_R. Once receiving K_R, the tag computes the hash of its temporary identifier ID_i, i.e. $h(ID_i)$, generates a new random string K_T, and computes the hash value of the concatenated string (ID_i, K_R, K_T), i.e. $h(ID_i||K_R||K_T)$. Then the tag transmits $h(ID_i)$, K_T, and $h(ID_i||K_R||K_T)$ back to the reader and uses a hash-based device to update its identifier as ID_{i+1}. A legitimate reader knows the tag's temporary identifier ID_i by a quick search for the hash value $h(ID_i)$ stored in the database. After computing ID_i, the database can also update the tag's identifier as ID_{i+1} following the same mechanism of the tag. Now, the legitimate reader can pass the tag's challenge. It just computes the hash of the concatenated string of ID_{i+1}, K_R, and K_T back to the tag. Finally, the tag can check the validity by computing $h(ID_{i+1}||K_R||K_T)$ and comparing it with the message sent from the reader. This scheme allows both readers and tags always in perfect synchronization, and hence guarantees the forward privacy. Nevertheless, as mentioned in [3, 5], if no authorized reader refreshes the tag's temporary identifier, this identifer remains constant and thus may be tracked again. Even between identification session with a legitimate reader, the output of the tag remains static. Tags may be tracked during this period [3].

From the above discussion, one may wonder whether it is possible to have a fully privacy-preserving hash-based RFID authentication scheme with an efficient identification process which only requires $o(n)$ computation of the back-end database where n is the number of tags. In fact, it has remained an open question since it was proposed by Juels [3].

3 Our contribution and technique

In this work, we give a fully privacy-preserving hash-based protocol in which the time complexity is sub-linear in an amortized sense. This partially answers the open problem proposed by Juels [3]. Here, we briefly give our construction idea. The intuition behind our protocol is to delicately combine the randomize hash lock scheme [7] and Dimitriou's mutual authentication scheme [2]. As mentioned in the previous section, Dimitriou's protocol is almost fully privacy-preserving except for the location tracking by unauthorized parties. The main drawback is that the output $h(ID_i)$ of the target tag is always constant (hence it is easy to be tracked) when illegitimate readers query it ceaseless. To overcome it, we apply the randomized hash lock technique with a short random string. However, this results in a brute force search in the back-end database as shown in [7]. To solve this dilemma, we observe the following. Since the random string we apply is short enough, for each tag, we may store all possible hash values of the the tag's identifier associated with all possible short random strings in the database. It requires not too much space to implement the database. Thus the time complexity of the back-end database search is still more efficient than the fully randomized hash lock scheme in [7]. On the other hand, the proposed scheme seems to bear being tracked by illegitimate readers' endless query. However, a birthday attack may render the tag being tracked again regardless of authorized or unauthorized queries. To bypass it, in each tag we use a counter to record the number of illegitimate queries. Once the number of illegitimate queries is greater than some fixed threshold value, the protocol proceeds to execute the original fully randomized hash lock scheme to update the tag's identifier. As you can see, our scheme obtains lower amortized cost in the authentication process while preserving the full privacy.

The remaining part of the paper is organized as follows. In Section 4, we give our mutual authentication scheme and its security analysis. Finally we conclude in Section 5.

4 Our Protocol

In this section, we give our authentication scheme. For convenience, we use the following notation in the rest of the paper.

Notation	Corresponding Meaning
n	Number of tags participating in the RFID system
Tag_q	The q-th tag
Reader	The legitimate reader
RNG	The random number generator
$A_{q,i}$	The temporary identifier of Tag_q during the i-th legitimate query.
$M_1\|\|M_2$	Concatenation of messages M_1 and M_2.
h	The hash function used in the protocol.
K_R	The random string generated by the reader.
K_T	The random string generated by the tag.
r	The short random string generated by the tag.
C_q	The counter which counts the number of illegitimate queries.

Now we are going to describe our protocol in the following subsections.

4.1 Initialization

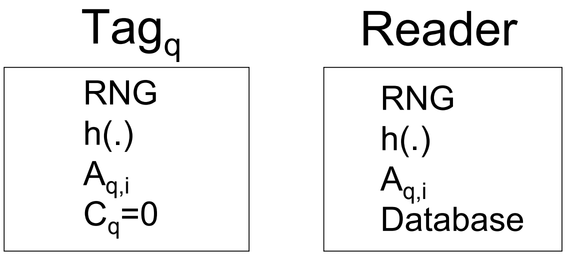

Figure 1: Initialization Step for tags and readers.

First of all, let us describe the setup of tags and readers. Each Tag and each reader have random number generators and a hash function h. As mentioned in [2], our protocol relies on the existence of a random secret shared both in a particular tag and the reader. For the q-th tag, let $A_{q,i}$ denote its shared secret which is also stored in the database. Moreover, to guarantee forward privacy, we allow both the q-th tag and the reader use an irreversible computation to refresh the identifier as $A_{q,i+1}$ in a synchronized way once the authentication process successes. Finally, in order to prevent a birthday attack, we set a counter C_q to count the number of illegitimate queries. Note that we assume that the tag is loaded with an initial identifier ID_0 which is chosen randomly. To be more illustrative, the protocol setup is shown in Fig. 1.

Next, let us look closely at the implementation of the back-end database. Note that we combine a short random string r and the temporary identifier $A_{q,i}$ ($metaID$) with a hash function h. Let $r_1, \ldots, r_m$ be the enumerate of all possible random strings r of length $\log m$. To be efficient to find the $metaID$, we store the index $h(r_k||A_{q,i})$ for all $k \in \{1, \ldots, m\}$ as shown in Fig. 2.

4.2 Authentication Process

Now we proceed to show how the authentication process goes as follows. The implemented version is illustrated in Fig. 3.

Back-end Database	
Index	metaID
$h(r_1\|\|A_{q,i})$	
....	$A_{q,i}$
$h(r_m\|\|A_{q,i})$	
	...
$h(r_1\|\|A_{p,i})$	
....	$A_{p,i}$
$h(r_m\|\|A_{p,i})$	

Figure 2: The structure of the back-end database.

1. The reader requests the tag and sends a long random string K_R to it.

2. The tag generates a short random string r and a long random string K_T. Then it computes two hash values $h(r||A_{q,i})$ and $h(A_{q,i}||K_T||K_R)$. Next the tag sends them as well as r and K_T to the reader.

3. The reader's first goal is to identify the tag. It uses the value $h(r||A_{q,i})$ to quickly search for the index in the database and find out the tag's $metaID = A_{q,i}$. It then computes the hash value $h(A_{q,i}||K_T||K_R)$ by itself and check if this value equals the one the tag sent. If it matches, the reader computes the hash value $h(A_{q,i}||K_T)$, sends the tag this value, and refreshes $A_{q,i}$ to $A_{q,i+1}$. While the reader refreshes the identifier, it also updates all indices related to the used identifer $A_{q,i}$ in the hash table.

4. After receiving the value $h(A_{q,i}||K_T)$, the tag checks whether it is a true value by computing the corresponding hash value. If yes, then it allows the reader to use its all functionality and updates $A_{q,i}$ to $A_{q,i+1}$. If not, the tag rejects and records this illegitimate query by adding the counter C_q.

5. Once the number recorded in the counter C_q is greater than some threshold value (say $m^{1/3}$), the tag starts to run the fully randomized hash lock protocol [7] with readers. We depict the flow chart in Fig. 4.

4.3 Complexity and Security Analysis

Here we analyze the computational complexity of our protocol. Suppose the tag does not enter the fully randomized hash lock scheme. The identification process can be done in time $O(\log n)$. Once the mutual authentication successes, refreshing the indices related to the used identifier requires

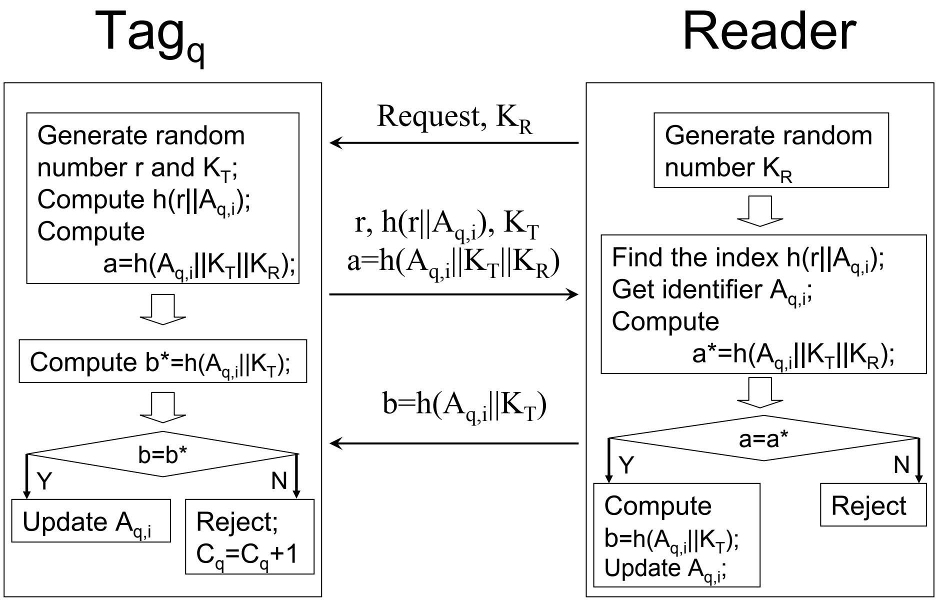

Figure 3: A tag-reader authentication protocol.

time m where $\log m$ is the length of the short random string generated by the tag. Thus the computation of one successful authentication phase requires time $m + O(\log n)$.

On the other hand, if the tag starts the fully randomized hash lock scheme, this requires time $O(n)$. So in the worst-case case, we do not improve the scheme in [7]. Nevertheless, in an amortized viewpoint, our protocol has running time much less than all previous results. Since we set a counter C_q for the q-th tag, the tag only starts the fully randomized hash lock scheme when C_q is greater than a threshold value T. Thus the amortized cost of each successful authentication execution is at most $m + O(\log n) + O(n/T)$. If we let $m = n^{3/4}$ and $T = m^{1/3}$, then the amortized cost is at most $O(n^{3/4})$ which is significantly less than n. The last step is to guarantee the security under such parameter settings.

Similar to the security analysis of Dimitriou's protocol [2], our protocol can prevent adversaries from not only all attacks mentioned in [2] but also the following attacks.

- *Attack on location privacy.* Unlike Dimitriou's protocol, our protocol ensures the location privacy not only for legitimate reader queries but also for illegitimate reader queries. Note that, as mentioned in the introduction, Dimitriou's protocol cannot prevent the tag's location privacy from illegal requests since the queried tag always outputs the same value and hence it can be tracked. On the contrary, our protocol is able to prevent tags from such attacks since the tag outputs the hash value of the tag's temporary identifier and a short random string r. This results in a phase-variant output which looks random to adversaries.

- *Birthday attack on the hash function.* Since the tag hashes a random string r of quite short length concatenated by the tag's temporary identifier and then sends it to the reader. A huge number of unauthorized queries may cause the tag reveal the privacy information due to a birthday attack. Thus, to prevent this, we have a counter to record the number of

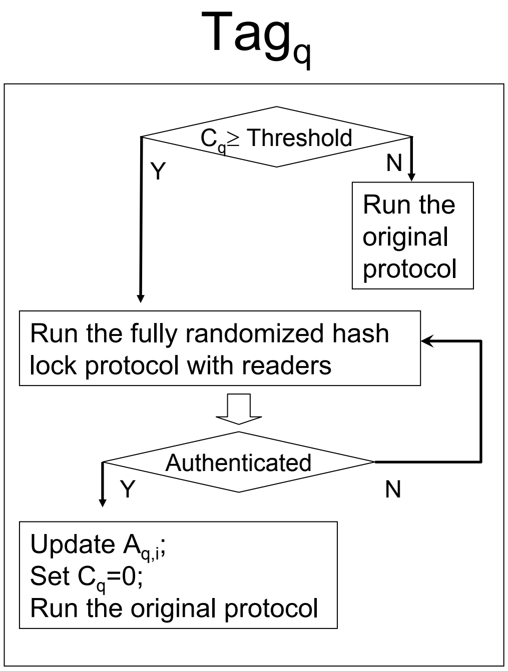

Figure 4: Defence for the huge number of illegitimate queries.

unauthorized queries. Once the number in the counter is greater than $m^{1/3} = n^{1/4}$, then it proceeds to run the fully randomized hash lock protocol with readers. Since the length of short random strings is $3 \log n/4$, the probability that a birthday attack with $n^{1/4}$ queries successes is negligible. Note that, after entering the state to proceed the fully randomized scheme, such a scheme still preserves location privacy against the attack by a huge number of illegitimate queries.

5 Conclusion

In this paper, we give a fully privacy-preserving authentication protocol between RFID tags and readers. The time complexity of a successful mutual authentication between a tag and a reader requires $O(n^{3/4})$ time in an amortized analysis where n is the number of tags participating in the RFID system. This partially answers an open problem proposed in [3] which asks whether there exists a fully privacy-preserving RFID protocol in which the time complexity is $o(n)$ in the worst case.

In addition, our protocol improves the protocol proposed by Dimitriou [2] in which it cannot prevent the object embedding an RFID tag from being tracked by unauthorized parties. The main contribution of our protocol is to prevent this kind of attack while keeping the amortized time complexity within $o(n)$ where n is the number of tags involving in the RFID system. Finally, the major open problem proposed by Juels [3] is still to determine whether one can obtain a fully privacy-preserving protocol whose (not amortized) time complexity is within sub-linear time.

Acknowledgment

The authors would like to thank the National Science Council of Taiwan for supporting this research in part under contracts NSC-96-2221-E-305-006-MY3 and NSC-97-2218-E-305-001-MY2.

References

[1] Gildas Avoine and Philippe Oechslin. A scalable and provably secure hash based RFID protocol. In *International Workshop on Pervasive Computing and Communication Security – PerSec 2005*, pages 110–114, Kauai Island, Hawaii, USA, March 2005. IEEE, IEEE Computer Society Press.

[2] Tassos Dimitriou. A lightweight RFID protocol to protect against traceability and cloning attacks. *Security and Privacy for Emerging Areas in Communications Networks, International Conference on*, 0:59–66, 2005.

[3] Ari Juels. RFID security and privacy: a research survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, 2006.

[4] Ari Juels and Stephen A. Weis. Defining strong privacy for RFID. In *PerCom Workshops*, pages 342–347. IEEE Computer Society, 2007.

[5] Marc Langheinrich. A survey of RFID privacy approaches. *Personal and Ubiquitous Computing*, October 2008. Online First Edition, available from http://www.springerlink.com/content/p71246k75029v715/.

[6] Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic approach to privacy-friendly tags. In *In RFID Privacy Workshop*, 2003.

[7] Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In *SPC*, pages 201–212, 2003.