Reducing RFID Reader Load with the Meet-in-the-Middle Strategy

Jung Hee Cheon, Jeongdae Hong and Gene Tsudik *

Abstract

In almost any RFID system, a reader needs to identify, and optionally authenticate, a multitude of tags. If each tag has a unique secret, identification and authentication are trivial, however, the reader (or a back-end server) needs to perform a brute-force search for each tag-reader interaction. In this paper, we suggest a simple, efficient and secure technique that reduces reader computation to $O(\sqrt{N} \cdot \log N)$. Our technique is based on the well-known "meet-in-the-middle" strategy used in the past to attack certain symmetric ciphers.

Keywords: RFID Identification, Authentication, Meet-in-the-Middle Strategy, PRF.

1 Introduction

A general Radio Frequency IDentification (RFID) system consists of a back-end database (or simply database), multiple readers, and a multitude of tags. The database maintains all system data. A reader relays data between the database and tags by wireless communication with multiple tags in the same period. A tag is typically attached to an object or a set of objects. A tag usually contains a unique identification number and some object-related data. We skip the technical details of a general RFID architecture and refer to [5] for a good introduction to RFID technology.

In a typical RFID system, the reader and the tags communicate wirelessly. Wireless communication prompts the same set of security threats encountered in a general wireless system. Eavesdropping on tag-reader communication might allow the adversary to track tags. A malicious reader can likewise track tags. A fraudulent (e.g., cloned) tag can convince a reader that it is genuine by supplying fake information. Last but not least, compromised tags and readers need to be promptly revoked. Various solutions to some these threats have been proposed, e.g., in [4] and [6].

*J. Cheon and J. Hong are with Seoul National University, Seoul, Korea. (e-mail: jhcheon@snu.ac.kr, jdhong@theory.snu.ac.kr) G. Tsudik is with University of California, Irvine (e-mail: gts@ics.uci.edu)

In this paper, we focus on tag identification using keyed Pseudo-Random Functions (PRF-s). (See [3] for definitions and a theoretical background on PRF-s.) Usage of PRF-s for identification prompts a key management problem. If a single secret key is used for all tags, compromise of a single tag breaks the entire system. If a unique per-tag secret key is used, privacy concerns preclude the tag from identifying itself explicitly during reader-tag interaction; therefore, the reader has to "guess" the tag's identity and determine it via brute force. Since modern RFID systems can involve upwards of 10^{10} tags and require readers to scan as many tags as possible as fast as possible, a brute force search for each tag-reader interaction is very inefficient. This triggers our motivation for reducing tag identification complexity on the reader side.

Contribution Summary We propose a secure and efficient RFID tag identification technique using PRF-s. The distinctive feature of our technique is that the reader identifies the tag with only $2\sqrt{N}$ PRF-s, where N is the number of tags in the entire RFID system. Whereas, an adversary, who does not know the keys, needs to exhaustively search the entire $O(2^{\ell+1})$ key-space, even if it uses the meet-in-the-middle attack strategy, where 2ℓ is the key bit-size. We also propose some extensions to the basic technique.

2 Efficient Key Searching Protocol

We now describe our technique for efficient tag identification. We assume that the following system participants:

- Server ($\mathcal{SRV}$): trusted entity that sets up the system.
- Reader ($\mathcal{RDR}$): trusted device that interacts with tags.
- Tags ($T_1, ..., T_N$): set of RFID tags.
- Adversary ($\mathcal{ADV}$): malicious entity that aims to subvert the system.

2.1 System Setup

We assume that each tag is equipped with a PRF and has enough memory for a unique secret key shared with the reader. In the setup phase, $\mathcal{SRV}$ selects two global parameters: (1) security parameter ℓ (e.g., $\ell = 80$ for 2^{80} security) and (2) sufficiently large integer N which represents the

maximum number of tags in the system (e.g., $N = 1,000,000$). For notational convenience and without loss of generality, we assume that integer N is a square.[1]

Next, $\mathcal{SRV}$ constructs two key-sets:

$$\mathcal{K}1 = \{K_x \mid 1 \leq x \leq \sqrt{N}\}, \qquad \mathcal{K}2 = \{K_y \mid 1 \leq y \leq \sqrt{N}\}$$

such that $\mathcal{K}1 \cap \mathcal{K}2 = \emptyset$. Then, for each tag T_i ($1 \leq i \leq N$), $\mathcal{SRV}$ generates a distinct (two-part) secret key $K^i = \langle K_1^i, K_2^i \rangle$ where:

$$K_1^i = K_x \in \mathcal{K}1, x \in_R [1, \sqrt{N}], \qquad K_2^i = K_y \in \mathcal{K}2, y \in_R [1, \sqrt{N}].$$

In other words, $K^i \in_R \mathcal{K}1 \times \mathcal{K}2$. Note that notation $\in_R$ means "randomly chosen from". Also, selection is done **without replacement**, i.e., for all $i \neq j \in [1, N]$, $K^i \neq K^j$. $\mathcal{SRV}$ then initializes each tag T_i with K^i. K^i, in effect, represents a tag's identity and its unique identifier. The two key-sets $\mathcal{K}1, \mathcal{K}2$ are distributed to each reader.

In the end, each tag only stores a 2ℓ -bit key $K^i = K_1^i \| K_2^i$. Although all tag keys are distinct, $\mathcal{RDR}$ only needs to store set $\mathcal{K}1$ and $\mathcal{K}2$, which together amount to $2\ell\sqrt{N}$ bits.

2.2 Private Identification Protocol

We follow the protocol defined in the ISO/IEC 9798-2 standard. First, $\mathcal{RDR}$ sends a random challenge r to a tag. The tag generates its own random number r', computes:

$$C = \mathrm{PRF}_{K_1^i}(r, r') \oplus \mathrm{PRF}_{K_2^i}(r, r')$$

and sends $\langle C, r' \rangle$ back to $\mathcal{RDR}$.

The reader searches for the tag key in two key-sets $\mathcal{K}1$ and $\mathcal{K}2$ by checking:

$$\{C \oplus \mathrm{PRF}_{K_2^j}(r, r')\} \stackrel{?}{=} \mathrm{PRF}_{K_1^j}(r, r')$$

If there is a match, $\mathcal{RDR}$ identifies the tag.

Note that the protocol requires the tag to generate its own random number r' and include it in all PRF invocations. This is necessary in order to avoid tracking by malicious readers. If the tag computed $C = \mathrm{PRF}_{K_1^i}(r) \oplus \mathrm{PRF}_{K_2^i}(r)$, it would be trivial for a malicious reader to track the tag, since, given the same tag and the same reader challenge r, the tags replies (C-s) would be the same. Moreover, it is difficult to imagine a realistic RFID usage scenario where malicious reader attacks are not applicable.

[1] Otherwise, we would need to replace $\sqrt{N}$ by $\lceil \sqrt{N} \rceil$ – the largest integer not exceeding $\sqrt{N}$.

Figure 1: Key Search Pseudocode

Key Search Algorithm (r, r', C)

for $j = 1$ to $\sqrt{N}$ **do**

Compute $X = \mathrm{PRF}_{K_1^j}(r, r')$

Set $TABLE[j] \leftarrow \langle K_1^j, X \rangle$

end for

SORT($TABLE$)

for $i = 1$ to $\sqrt{N}$ **do**

Compute $Y = C \oplus \mathrm{PRF}_{K_2^i}(r, r')$

Set $Z = \mathrm{SEARCH}(TABLE, Y)$

If $(Z > 0)$ THEN BREAK

end for

If $(Z > 0)$ THEN RETURN$(TABLE[Z], K_2^i)$

2.3 Search Algorithm

As described above, $\mathcal{RDR}$ finds the key-pair (K_1^i, K_2^i) using the key-set $\mathcal{K}1 \times \mathcal{K}2$, which is much smaller than the total $2^{2\ell}$ key-space. Upon receiving C, $\mathcal{RDR}$ computes a $\sqrt{N}$-sized table of $\mathrm{PRF}_{K_1^j}(r, r')$ computed with all possible keys $K_1^j \in \mathcal{K}1$. Each entry in the table contains: $\langle K_1^j, \mathrm{PRF}_{K_1^j}(r, r') \rangle$. Then, $\mathcal{RDR}$ sorts the table. Next, for each $K_2^i \in \mathcal{K}2$, $\mathcal{RDR}$ computes $C \oplus \mathrm{PRF}_{K_2^i}(r, r')$ and searches for it in the table. If the search succeeds, $\mathcal{RDR}$ determines the key pair (K_1^j, K_2^i) which uniquely identifies the tag.

The key search algorithm is shown in more detail in Fig. 1. The reader's total computation cost is $O(\sqrt{N} \cdot \log N)$ since it requires $O(\sqrt{N} \cdot \log N)$ for SORT$(\cdot)$ and SEARCH$(\cdot)$, and $2\sqrt{N}$ PRF$(\cdot)$ operations.

2.4 Details and Measurements

Instead of a PRF, we can also use Pseudo-Random Permutation (PRP). Every PRP has an inverse function $PRP_K^{-1}(\cdot)$ which can be easily computed with the knowledge of K. In that case, the tag would compute $C = PRP_{K_2^i}(\mathrm{PRF}_{K_1^i}(r, r'))$. $\mathcal{RDR}$ would search for the tag by checking:

$$\mathrm{PRF}_{K_1^i}(r, r') \stackrel{?}{=} PRP_{K_2^i}^{-1}(C).$$

Any efficient and secure symmetric cipher (e.g., AES) can be used as a PRF. In fact, AES can be used as both PRP and PRF. For example, Feldhofer, et al. [2] proposed an implementation of 128-bit AES for RFID tags which requires only $3,600$ gates. If the cipher requires a 128-bit key, we can add certain padding to ℓ-bit keys. More compact hardware implementations are expected in the near future.

Table 1 shows average run-time for our protocol using AES as the underlying PRF with $N = 1,000,000$. All experiments were conducted on a 1.86-GHz Intel Core-Duo processor with 2GB memory, MS Windows Vista and VS.NET. All timings were obtained from averages over 100 runs with randomly chosen K_1, K_2, r and r'. The search time is comparatively high since, in the worst case, $\mathcal{R}DR$ invokes PRF $|\mathcal{K}2|$ times. Our search algorithm takes about 2.07ms per tag, which is 200 times faster than brute force search (433.52ms).

Table 1: Measurements (in milliseconds) with 128-bit AES

Operation:			
Table Gen.	Table Sort	Search	Total
0.671074	0.572694	0.832777	2.076545

3 Extensions

In the proposed protocol, if $\mathcal{A}DV$ directly compromises c tags, it can effectively impersonate and track $c^2 - c$ other tags. For example, consider two compromised tags T_i and T_j. Given their keys: $\langle K_1^i, K_2^i \rangle, \langle K_1^j, K_2^j \rangle$, $\mathcal{A}DV$ can impersonate and track two other tags (say, T_x and T_y) corresponding to key-pairs: $\langle K_1^i, K_2^j \rangle$ and $\langle K_1^j, K_2^i \rangle$, respectively. We call such tags (i.e., T_x and T_y) *indirectly-compromised*.

We now construct two extensions to mitigate this problem. The first reduces $\mathcal{A}DV$'s probability of tracking and impersonation of indirectly-compromised tags. The second extension completely prevents impersonation (but not tracking) of indirectly-compromised tags via explicit authentication. In addition, we propose a protocol combining with tree-based protocol of Molnar and Wagner [7].

3.1 Extension I: k-resilient Protocol

In the system setup phase, $\mathcal{SRV}$ selects a resiliency parameter k. Then, the maximum number of tags is N_0, where $N = (k\lceil\sqrt{N_0}\rceil)^2$. Since the probability of a key-pair being assigned to a real tag is: $N_0/N \leq 1/k^2$, the expected number of non-compromised tags whose keys are revealed is: $(c^2-c)/k^2 \leq (c/k)^2$. For example, assuming $k = N_0^{1/4}$, the reader's computation is $O(N_0^{3/4} \cdot \log N_0)$ and the expected number of indirectly compromised tags stays below one, until the number of directly compromised tags grows to $N_0^{1/4}$. Considering compromised keys, the reader's search cost is: $O(k\sqrt{N_0} \cdot \log N_0)$.

3.2 Extension II: Authentication Protocol

At setup time, for each tag, we introduce an additional ℓ -bit key unique $K_3^i \in_r \mathcal{K}3$, i.e., each T_i has a three-part key: $K^i = K_1^i \| K_2^i \| K_3^i$. For $0 \leq \alpha \leq 1/2$, we set $|\mathcal{K}1| = |\mathcal{K}2| = N^\alpha$ and $|\mathcal{K}3| = N$ so that $N^{1-2\alpha}$ tags have the same key pair $\langle K_1^i, K_2^i \rangle$.

For each interaction with $\mathcal{RDR}$, in addition to C as described above, the tag also computes $C' = \mathrm{PRF}_{K_3^i}(r, r')$ and sends $\langle C, C' \rangle$ to $\mathcal{RDR}$. Next, $\mathcal{RDR}$ performs key search using C as before to determine $\langle K_1^i, K_2^i \rangle$. Then, $\mathcal{RDR}$ checks $N^{1-2\alpha}$ candidates of K_3^i to identify the correct K_3^i using C'. This last step represents *tag authentication*, as opposed to plain key-search which corresponds to (private) *tag identification*.

In this extension it might seem necessary for $\mathcal{RDR}$ to additionally store the entire N-element set $\mathcal{K}3$ as well as a way to associate each key in $\mathcal{K}3$ with a tuple drawn from $\mathcal{K}1 \times \mathcal{K}2$. This would represent a heavy storage burden over the plain protocol where $\mathcal{RDR}$ stored only $2 \cdot \sqrt{N}$ keys. However, one simple optimization which requires **no extra storage** is to generate each K_3^i as a one-way trapdoor function of $\langle K_1^i, K_2^i \rangle$, some master secret key KK known only to $\mathcal{SRV}$ and $\mathcal{RDR}$. For example, we can set $K_3^i = AES\_CBC(KK, K_1^i, K_2^i, b)$, where $(1 \leq b \leq N^{1-2\alpha})$. Note that the probability of randomly generated N 128-bit keys having a match is about $N^2/2^{128}$, which is very small for a reasonably large N. Besides KK, this involves no storage overhead and the computation overhead is clearly minimal.

It is easy to see that the use of K_3^i prevents impersonation of indirectly-compromised tags. However, it does not prevent their tracking completely since C is still computed as a function of K_1^i and K_2^i. However, a smaller value of α can further inhibit tracking, i.e. $\langle K_1^i, K_2^i \rangle$ pairs can be used to identify a tag with probability $1/N^{1-2\alpha}$.

3.3 Combination with Tree-based Protocol

Our basic protocol can be used with Molnar and Wagner's tree-based protocol (MW protocol) [7]. The protocol considers tags as leaves in a tree, then associate each edge in the tree with a secret (level key). Each tag should store the secrets corresponding to the path from the root to the tag. $\mathcal{RDR}$ is required to store all secrets of which number is determined by branching factors of the tree. In a d-ary tree, for identifying a tag, $\mathcal{RDR}$ needs $O(\log_d N)$ rounds of interactions with the tag and $d \cdot \log_d N$ PRF computations. If our protocol is used as a subprotocol for searching level keys, then the tag's computation is doubled up but the $\mathcal{RDR}$'s computation is reduced to $2\sqrt{d} \cdot \log_d N$ PRF-s with the same communication cost.

In the example using two-level tree for $N = 2^{20}$ tags, presented in [7], the branching factor is $d = 1024$. In this case, $\mathcal{RDR}$ needs to compute $2^{10} \cdot \log_{2^{10}} 2^{20} = 2048$ PRF-s for MW protocol, but it is reduced to $2 \cdot 2^5 \cdot \log_{2^{10}} 2^{20} = 128$ PRF-s for our protocol.

4 Analysis

We now briefly discuss efficiency and security properties of the proposed protocol.

4.1 Efficiency

In our (non-extended) protocol, the reader requires $O(|\mathcal{K}1| + |\mathcal{K}2|)$ operations for computing $\mathrm{PRF}_{K_1}(r, r')$ and $\mathrm{PRF}_{K_2}(r, r')$ for $K_1 \in \mathcal{K}1$ and $K_2 \in \mathcal{K}2$ and $O((|\mathcal{K}1| + |\mathcal{K}2|) \cdot \log|\mathcal{K}1|)$ for sorting ciphertexts and searching collisions. Therefore, when we take $|\mathcal{K}1| = |\mathcal{K}2| = N^{1/2}$, a reader requires only $O(\sqrt{N} \cdot \log N)$ computation cost, where N is the maximum number of tag keys. For an adversary who does not have an information on the sets $\mathcal{K}1$ and $\mathcal{K}2$, searching the key pair requires exhaustive search over all $2^{\ell+1}$ keys even though "meet-in-the-middle" attack strategy is used with a pair of plaintext and ciphertext. Our identification protocol requires only $O(\ell)$ bits per message (random number).

In the authentication protocol, when $|\mathcal{K}1| = |\mathcal{K}2| = N^{\alpha}$, the verification time is about $(2N^{\alpha} + N^{1-2\alpha})$ PRF computations. We find that $\alpha = \frac{1}{3}$ is optimal with the time complexity $O(N^{1/3} \cdot \log N)$.

4.2 Security

If no tag is compromised, it is easy to see that the proposed protocols are resistant to both tracking and impersonation. The former is because $\mathcal{ADV}$ can not link two different tag responses, and the latter – because $\mathcal{ADV}$ can not impersonate a tag without knowing that tag's key.

For example, consider the situation where c tags are compromised. In our basic protocol, this translates into at most $c^2 - c$ indirectly compromised tags, i.e., for which (K_1, K_2) are revealed. Aside from these tags, $\mathcal{ADV}$ can not tell if a tag has K_1 or K_2 as a part of his key. Therefore, tags with partially compromised key are still secure in the sense that the exhaustive search for the other key requires 2^ℓ computations.

In our k-resilient protocol, the number of indirectly-compromised tags can be made small by increasing the computation cost. In our authentication protocol, even the indirectly compromised tags can not be impersonated since their K_3 values remain secret (since (K_1, K_2) is only used to narrow down the candidates for K_3). However, $c^2 N^{1-2\alpha}$ indirectly compromised tags can be *weakly tracked* in the sense that it can be determined only whether it is one of the $N^{1-2\alpha}$ tags with the same (K_1, K_2).

In summary, if the number of compromised tags is very small, both basic and k-resilient protocols are appropriate; otherwise, our authentication protocol extension is preferable.

5 Conclusion

We proposed efficient and secure RFID tag identification and authentication technique using the well-known meet-in-the-middle strategy. In it, the reader's computation is reduced to $O(\sqrt{N} \cdot \log N)$ where N is the maximal number of tags. For authentication, we need to add one more message but the reader's computation is reduced to $O(N^{1/3} \cdot \log N)$.

References

[1] W. Diffie and M. Hellman, *Exhaustive Cryptanalysis of the NBS Data Encryption Standard*, IEEE Computer, Vol. 10, No. 6, pp. 74-84, June 1977.

[2] M. Feldhofer, S. Dominikus and J. Wolkerstorfer, *Strong Authentication for RFID Systems using the AES Algorithm*, Workshop on Cryptographic Hardware and Embedded Systems (CHES'04), LNCS 3156, pp. 357-370, Springer, 2004.

[3] O. Goldreich, *Foundations of Cryptography*, Volume 1, Chapter 3. Cambridge University Press, 2004.

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

[5] T. Karygiannis, B. Eydt, E. Barber, L. Bunn, and T. Phillips, *Guidelines for Securing Radio Frequency Identification (RFID) Systems*, NIST Special Publication 800-98, April. 2007.

[6] M. Lehtonen, T. Staake, F. Michahelles and E. Fleisch, *From Identification to Authentication – a Review of RFID Product Authentication Techniques*, Workshop on RFID Security (RFIDSEC'06), 2006.

[7] D. Molnar and D. Wagner, *Privacy and Security in Library RFID: Issues, Practices, and Architectures*, ACM Conference on Computer and Communications Security (CCS'04), November 2004.

[8] International Organization for Standardization. *ISO/IEC 9782-2: Information Technology – Security Techniques – Entity Authentication Mechanisms Part 2: Entity Authentication using Symmetric Techniques*, 1993.