Reducing Time Complexity in RFID Systems

Gildas Avoine[1], Etienne Dysli[1], and Philippe Oechslin[1,2]

[1] EPFL, Lausanne, Switzerland
[2] Objectif Sécurité, Gland, Switzerland

Abstract. Radio frequency identification systems based on low-cost computing devices is the new plaything that every company would like to adopt. Its goal can be either to improve the productivity or to strengthen the security. Specific identification protocols based on symmetric challenge-response have been developed in order to assure the privacy of the device bearers. Although these protocols fit the devices' constraints, they always suffer from a large time complexity. Existing protocols require $O(n)$ cryptographic operations to identify one device among n.

Molnar and Wagner suggested a method to reduce this complexity to $O(\log n)$. We show that their technique could degrade the privacy if the attacker has the possibility to tamper with at least one device. Because low-cost devices are not tamper-resistant, such an attack could be feasible. We give a detailed analysis of their protocol and evaluate the threat. Next, we extend an approach based on time-memory trade-offs whose goal is to improve Ohkubo, Suzuki, and Kinoshita's protocol. We show that in practice this approach reaches the same performances as Molnar and Wagner's method, without degrading privacy.

Keywords: RFID, time complexity, time-memory trade-off.

1 Introduction

Sometimes presented by the media as the next technological revolution after the Internet, Radio Frequency Identification (RFID) aims to identify objects remotely, with neither physical nor visual contact. They consist of transponders inserted into objects, readers which communicate with the transponders using a radio channel and a database which contains information on the objects.

This technology is not fundamentally new and concerns a whole range of applications. The first RFID application may have been the Royal British Air Force's "Identify Friend or Foe" system, which was used during the Second World War to identify friendly aircrafts. RFID systems have also been used for a few years in commercial applications, for example in contactless smart cards used on public transport. However, the boom that RFID technology enjoys today is chiefly due to the standardization [12, 6] and development of low-cost devices, so-called *tags*. This new generation of RFID tags has opened the door to hitherto unexplored applications. For example in supply chains as suggested by the EPC Global Inc. [6], to locate people in amusement parks [20], to combat the counterfeiting of expensive items [14], to trace livestock [5], to label books in libraries [16], etc.

B. Preneel and S. Tavares (Eds.): SAC 2005, LNCS 3897, pp. 291–306, 2006.

However, these tags also bring with them security and privacy issues. Security issues rely on classic attacks, e.g., denial of service, impersonation of tags or channel eavesdropping. These attacks are rendered more practicable because of the tags' lack of computational and storage capacity. More details on the technical aspects of the tags can be found for example in [14, 7, 8]. Current research deals with these problems but most of them are inherent to the technology itself, and applications have to make do with them. For these reasons, the RFID technology is more suited for bringing functionality (e.g., [6, 20, 5, 16]) rather than security (e.g., [14, 4]).

Nevertheless, whether it has a security or a functionality goal, radio frequency identification raises issues linked to privacy, in particular the problem of traceability of objects and thus indirectly of people [2]. Other technologies also permit the tracking of people, e.g., video surveillance, GSM, Bluetooth, and are extensively used by law enforcement agencies among others. However, RFID tags would permit everybody to track people using only low-cost equipment. This is strengthened by the fact that tags cannot be switched off, they can be easily hidden, their lifespan is not limited, and analyzing the collected data can be efficiently automated. Whether the defenders of RFID minimize the problem of privacy and its detractors amplify it, the fact that some companies have had to renounce this technology after being boycotted by associations [21] which defend individuals' liberty shows that we need to address this problem. Several palliative ways have been explored in order to solve this problem. For example, Juels, Rivest, and Szydlo proposed the "blocker tag" [15] whose goal is to prevent the readers from identifying the tags. With a very different approach, Garfinkel stated the "RFID Bill of Rights" which relates the fundamental rights of the tags' bearers. Today's challenge is to find protocols which allow authorized parties to identify the tags without an adversary being able to track them, thus getting to the root of the privacy problem.

The reason that we cannot use well-known authentication protocols comes from the fact that such protocols do not preserve the privacy of the prover. In other words, the verifier can check whether or not the identity claimed by the prover is true, but he cannot guess it himself: the prover must send his identity in clear which in turn allows an adversary to track him.

Asymmetric cryptography could easily solve this problem: the prover encrypts his identity with the public key of the verifier. Thus, no eavesdropper is able to identify the prover. Unfortunately, asymmetric cryptography is too heavy to be implemented within a tag. Certain classes of tags are simply not able to use cryptography, e.g., Class 0 and Class 1 tags (according to the EPC [6] classification). Several protocols suited to these tags have been proposed (see for example [13,22,11,9] or [1] for a more exhaustive list) but even if they can reduce the capabilities of the attacker – which make them an attractive option – none of them can assure strong privacy. Therefore, we will not consider these tags below; instead, in this paper, we will focus on tags which are capable of embedding a symmetric cryptographic function, either a hash function or a secret-key cipher,

as the suited implementation of AES suggested by Feldhofer, Dominikus, and Wolkerstorfer [7].

The problem remains that both prover and verifier need to share a common key if a secret-key cipher is used instead of a public-key one. In RFID systems, provers (tags) are not tamper-resistant. Therefore an attacker who tampers with a tag can track its past events if she had access to its previous interactions with readers, e.g., from readers' log files. One could point out that the ease of tampering with a tag is counter-balanced by the difficulty of getting access to it. That is the case with sub-dermal tags for example, or bracelet tags in amusements parks which could be re-initialized when the customer gives back his bracelet [20]. Nevertheless, using a common key for all the tags would be a pity: an attacker who tampers with one tag, e.g., her own tag, would also be able to track all the other tags in the system.

Consequently, another approach consists of using a unique key for each tag such that only the verifier (system) knows all these keys. However, this approach, which is the one taken by several reference papers [16, 19, 23], suffers from an expensive time complexity on the system's side. Indeed, because only symmetric cryptographic functions can be used, the system needs to explore its entire database in order to retrieve the identity of the tag it queries. If n is the number of tags managed by the system, $O(n)$ cryptographic operations are required in order to identify one tag. The advantage of the system over an attacker is that the system knows in which subset of identifiers it needs to search while the attacker has to explore the full range of identifiers.

We will address this problem in the rest of the paper. First of all, we will introduce a real life example in a library. We will use this example to compare the protocols which will be considered in this work. Section 3 will be devoted to the basic secret-key challenge-response protocol, denoted CR and the technique suggested by Molnar and Wagner [16], denoted CR/MW, whose goal is to reduce the complexity of CR. We will prove that this technique degrades the privacy when an attacker is able to tamper with at least one tag. In Section 5, we will deal with the protocol of Ohkubo, Suzuki, and Kinoshita [18], denoted OSK. Relying on a previous abstract [3], we will show in Section 6 how a time-memory trade-off can significantly reduce the identification time of OSK. This variant, called OSK/AO, is as efficient as CR/MW but does not degrade privacy. We will finally summarize our results in Section 7.

2 A Practical Example in a Library

In order to illustrate our comparison between CR, CR/MW, OSK, and OSK/AO, we consider a real life scenario in a library, where tags are used to identify books. Several libraries already use this technology, for example the libraries of Santa Clara (USA), Heiloo (Netherlands), Richmond Hill (Canada), and K.U. Leuven (Belgium). In a library scenario, it is realistic to assume that the tags can contain a secret-key cipher or a hash function because they are not disposable. Thus, a slightly higher cost is conceivable.

In the next sections, we assume that the system relies on a single computer which takes $\theta = 2^{-23}$ seconds to carry out a cryptographic operation, either hashing or encrypting a 128-bit blocks. The library manages 2^{20} tags. As described by Avoine and Oechslin in [2] and also by Molnar and Wagner in [16], we assume that tag singulation and collision avoidance are private and performed at a lower layer. Identification of several tags is therefore sequential. Current implementations allow a single reader to read several hundreds of tags per second, meaning that the system should spend at the most a few milliseconds to identify one tag. In the following sections, t_{P} will denote the average time to identify one tag using a protocol P. Because certain applications (in libraries, in amusement parks, etc.) may use numerous readers, the system should not become a bottleneck in terms of computation. Thus, the system should be capable of identifying the whole set of tags it manages in a few seconds only (e.g., for real-time inventories).

3 Description of Molnar and Wagner's Protocol

Several challenge-response protocols suited to RFID have been suggested during the last years, e.g., [16, 19, 7, 23]. We describe below those suggested by Molnar and Wagner, based on a pseudo-random function.

3.1 Challenge-Response Building Block

The Molnar and Wagner's challenge-response building block (CR), depicted on Figure 1, provides mutual authentication of the reader and the tag in a private way. It shall prevent an attacker from impersonating, tracing or identifying tags.

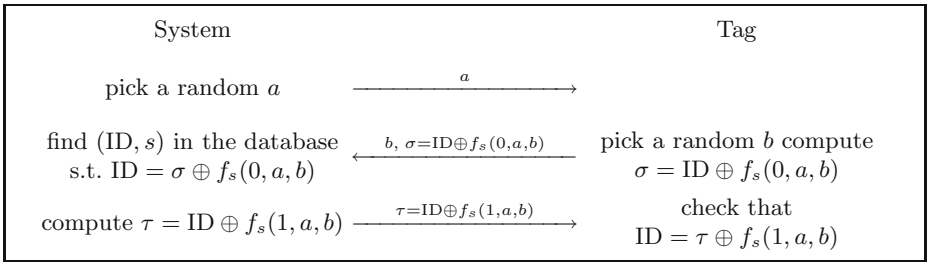

Fig. 1. Challenge-response protocol of Molnar and Wagner

Let ID be the tag's identifier which is stored in both the database of the system and the tag. They also share a secret key s. To initiate the authentication, the reader sends a nonce a to the tag. Next, the tag picks a random b and answers $\sigma := \mathrm{ID} \oplus f_s(0,a,b)$, where f_s is a pseudo-random function. The system retrieves the identity of the tag by finding the pair (ID, s) in its database such that $\mathrm{ID} = \sigma \oplus f_s(0,a,b)$. This completes the authentication of the tag. Now, in order to achieve mutual authentication, the system sends back $\tau := \mathrm{ID} \oplus f_s(1,a,b)$ to the tag. The tag can thus verify the identity of the reader by checking that $\mathrm{ID} = \tau \oplus f_s(1,a,b)$.

3.2 Efficiency

In order to identify a tag, the system must carry out an exhaustive search on the n secrets stored in its database. Therefore the system's workload is linear in the number of tags. More precisely, the average number of cryptographic operations required to identify one tag is $n/2$ and therefore we have $t_{\mathsf{CR}} = \frac{n\theta}{2}$. With the parameters given in Section 2, we have $t_{\mathsf{CR}} \approx 62$ ms which is too high in practice. Since CR does not scale well in a system with many tags, we next examine the three-based technique of Molnar and Wagner [16], whose main strength is the reduction of the system's workload from $O(n)$ to $O(\log n)$.

3.3 Tree-Based Technique

The technique suggested by Molnar and Wagner [16], namely CR/MW, relies on a tree structure in order to reduce the identification complexity. Instead of searching a flat space of secrets, let's arrange them in a balanced tree with branching factor δ. The tags are the leaves of this tree and each edge is associated with a value. Each tag has to store the values along the path from the root of the tree to itself. This sequence makes up its *secret*, and each value is called a *block* of secret. On the other side, the reader knows all secrets. We describe the protocol below.

Setup. Let n be the number of tags managed by the system and $\ell := \lceil \log_\delta n \rceil$ be the depth of the tree with a branching factor δ. Each edge in the tree is valuated with a randomly chosen secret $r_{i,j}$ where i is the level in the tree and j is the number of the branch. Figure 2 represents such a tree with parameters $n = 9$ and $\delta = 3$. The secret of a given tag is the list of the values $r_{i,j}$ from the root to the leaf. For example, the secret of T_5 on Figure 2 is $[r_{1,1}, r_{2,5}]$.

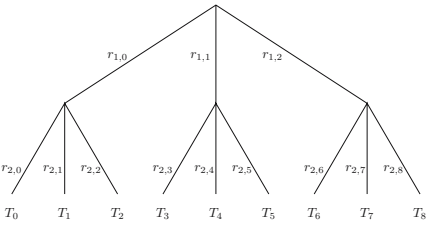

Fig. 2. Tree of tags' secrets

Interrogation. The tag is queried level by level from the root to the leaves. At each level i, CR/MW runs CR for each secret of the explored subtree. That is the reader tries every edge in turn in order to know on which one the tag is. If CR fails for all current level's secrets, the tag rejects the reader and the protocol stops. If the reader has been successfully authenticated at each level the protocol succeeds. Note that CR inevitably does not need to be performed δ times per level in practice. One run is enough if the reader checks the tag's answer with all current level's secrets, as described below.

Identification. At each level i, the system has to search in a set of δ secrets for the one matching the tag's secret. Given that $[s_1, \ldots, s_\ell]$ denotes a secret, the system has thus to compute $\delta/2$ times $f_{s_i}(0, a, b)$ on average at level i, meaning that $\frac{\delta}{2}\ell$ operations are required in order to identify one tag. Thus we have

$$t_{\mathsf{CR/MW}} = \frac{\delta\theta}{2} \log_\delta n.$$

The identification of one tag is far below the threshold of a few milliseconds. Identifying the whole system takes more than 2 minutes when $\delta = 2^{10}$ and decreases to 2 seconds when $\delta = 2$. However, we will see in Section 4 that having a small branching factor enables to trace the tags.

4 Privacy-Weakening Attacks

4.1 Tampering with Only One Tag

We examine in this section how the tree technique suggested by Molnar and Wagner allows tracing a tag when the attacker is able to tamper with some tag. The attack consists of three phases:

1. The attacker has one tag T_0 (e.g., her own) she can tamper with and thus obtain its complete secret. For the sake of calculation simplicity, we assume that T_0 is put back into circulation. When the number of tags in the system is large, this does not significantly affect the results.
2. She then chooses a target tag T. She can query it as much as she wants but she cannot tamper with it.
3. Given two tags T_1 and T_2 such that $T \in \{T_1, T_2\}$, we say that the attacker succeeds if she definitely knows which of T_1 and T_2 is T. We define the probability to trace T as being the probability that the attacker succeeds. To do that, the attacker can query T_1 and T_2 as many times as she wants but, obviously, cannot tamper with them.

We assume that the underlying challenge-response protocol assures privacy when all the blocks of secrets are chosen according to a uniform distribution. We consequently assume that the attacker cannot carry out an exhaustive search over the secret space. Hence, the only way for an attacker to guess a block of secret of a given tag is to query it with the blocks of secret she obtained by tampering with some tag. When she tampers with only one tag, she obtains only one block of secret per level in the tree. Thus, she queries T, and then T_1, and T_2 with this block. If either T_1 or T_2 (but not both) has the same block as T_0, she is able to determine which of them is T. If neither T_1 nor T_2 has the same block as T_0, she cannot answer. Finally, if both T_1 and T_2 have the same block as T_0, she cannot answer, but she can move on the next level of the tree because the authentication of the reader succeeded. We formalize the analysis below. We denote the secrets of T, T_0, T_1, and T_2 by $[s_1, \cdots, s_\ell]$, $[s_1^0, \cdots, s_\ell^0]$, $[s_1^1, \cdots, s_\ell^1]$, and $[s_1^2, \cdots, s_\ell^2]$ respectively. We consider a given level i where s_i^1 and s_i^2 are in the same subtree. Four cases must be considered:

- $C_i^1 = ((s_i^0 = s_i^1) \wedge (s_i^0 \neq s_i^2))$ then the attack succeeds,
- $C_i^2 = ((s_i^0 \neq s_i^1) \wedge (s_i^0 = s_i^2))$ then the attack succeeds,
- $C_i^3 = ((s_i^0 \neq s_i^1) \wedge (s_i^0 \neq s_i^2))$ then the attacks definitively fails,
- $C_i^4 = (s_i^0 = s_i^1 = s_i^2)$ then the attacks fails at level i but can move onto level $i+1$.

When the number of tags in the system is large, we can assume that

$$\Pr\left(C_i^1\right) = \Pr\left((s_i^0 = s_i^1)\right) \times \Pr\left((s_i^0 \neq s_i^2)\right).$$

The same assumption also applies to C_i^2, C_i^3, and C_i^4. Thus we have

$$\Pr\left(C_i^1 \vee C_i^2\right) = \frac{2(\delta-1)}{\delta^2} \quad (1 \leq i \leq \ell) \quad \text{and} \quad \Pr\left(C_i^4\right) = \frac{1}{\delta^2}.$$

The overall probability P that the whole attack succeeds is therefore

$$\begin{aligned} P &= \Pr\left(C_1^1 \vee C_1^2\right) + \sum_{i=2}^{\ell} \left(\Pr\left(C_i^1 \vee C_i^2\right) \times \prod_{j=1}^{i-1} \Pr\left(C_j^4\right) \right) \\ &= \frac{2(\delta-1)}{\delta^2} + \sum_{i=2}^{\ell} \left(\frac{2(\delta-1)}{\delta^2} \left(\frac{1}{\delta^2}\right)^{i-1} \right) = 2(\delta-1)\frac{1-\left(\frac{1}{\delta^2}\right)^{\ell}}{1-\frac{1}{\delta^2}}\frac{1}{\delta^2}. \end{aligned}$$

Remembering that $\delta^\ell = n$ yields $P = \dfrac{2}{\delta+1}\left(1 - \dfrac{1}{n^2}\right)$. The curve of P when $n = 2^{20}$ is the curve plotted on Figure 3 with $k_0 = 1$.

4.2 Tampering with Several Tags

We now consider the case where the attacker can tamper with more tags, e.g., she borrows several books in the library in order to tamper with their tags. We examine the influence of the number of opened tags on the probability of tracing the target tag. As before each opened tag is put back into circulation to simplify calculations. When n is large, this does not affect the results. As in the previous section, we denote the secrets of T, T_1, and T_2 by $[s_1, \cdots, s_\ell]$, $[s_1^1, \cdots, s_\ell^1]$, and $[s_1^2, \cdots, s_\ell^2]$ respectively. We consider a given level i where s_i^1 and s_i^2 are in the same (one-level) subtree. Let $\mathcal{K}_i$ denote the set of blocks of this (one-level) subtree which are known by the attacker and let $\mathcal{U}_i$ denote the set of those which are unknown by the attacker. k_i denotes the number of blocks in $\mathcal{K}_i$. Five cases must be considered:

- $C_i^1 = ((s_i^1 \in \mathcal{K}_i) \wedge (s_i^2 \in \mathcal{U}_i))$ then the attack succeeds,
- $C_i^2 = ((s_i^1 \in \mathcal{U}_i) \wedge (s_i^2 \in \mathcal{K}_i))$ then the attack succeeds,
- $C_i^3 = ((s_i^1 \in \mathcal{K}_i) \wedge (s_i^2 \in \mathcal{K}_i) \wedge (s_i^1 \neq s_i^2))$ then the attack succeeds,
- $C_i^4 = ((s_i^1 \in \mathcal{U}_i) \wedge (s_i^2 \in \mathcal{U}_i))$ then the attacks definitively fails,
- $C_i^5 = ((s_i^1 \in \mathcal{K}_i) \wedge (s_i^2 \in \mathcal{K}_i) \wedge (s_i^1 = s_i^2))$ then the attacks at level i fails but can move onto level $i+1$.

Thus, we have for all i such that $1 \leq i \leq \ell$:

$$\Pr(C_i^1 \vee C_i^2 \vee C_i^3) = \frac{2k_i}{\delta}\left(1 - \frac{k_i}{\delta}\right) + \left(\frac{k_i}{\delta}\right)^2 \left(1 - \frac{1}{k_i}\right)$$
$$= \frac{k_i}{\delta^2}(2\delta - k_i - 1),$$

and $$\Pr(C_i^5) = \frac{k_i}{\delta^2}.$$

The overall probability P that the attack succeeds is therefore

$$P = \Pr(C_1^1 \vee C_1^2 \vee C_1^3) + \sum_{i=2}^{\ell} \left(\Pr\left(C_i^1 \vee C_i^2 \vee C_i^3\right) \times \prod_{j=1}^{i-1} \Pr\left(C_j^5\right) \right)$$
$$= \frac{k_1}{\delta^2}\left(2\delta - k_1 - 1\right) + \sum_{i=2}^{\ell} \left(\frac{k_i}{\delta^2}(2\delta - k_i - 1) \prod_{j=1}^{i-1} \frac{k_j}{\delta^2} \right).$$

We now compute k_1, i.e., the number of different blocks known by the attacker at level 1, given that k_0 is the number of tags tampered with by the attacker. We have

$$k_1 = \delta\left(1 - (1 - \frac{1}{\delta})^{k_0}\right)$$

and then $$k_i = \delta\left(1 - (1 - \frac{1}{\delta})^{g(k_i)}\right) \quad (2 \leq i \leq \ell),$$

where $$g(k_i) = k_0 \prod_{j=1}^{i-1} \frac{1}{k_j}.$$

Results are plotted on Figure 3. We would like to highlight the surprising behavior of P when the branching factor is small. This is due to the fact that neither $\Pr(C_i^1 \vee C_i^2 \vee C_i^3)$ nor $\Pr(C_i^5)$ are monotonous and they reach their optimum at different values of δ. Table 1 supplies a few values in order to illustrate our attack.

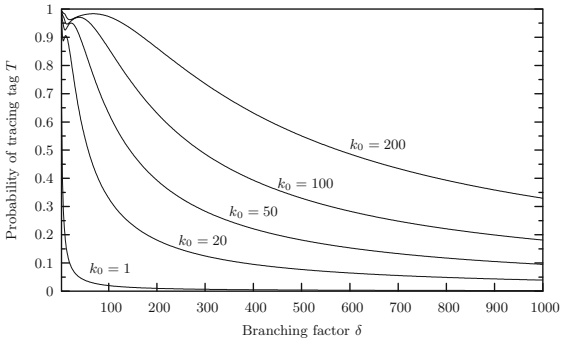

Fig. 3. Probability of tracing a tag when the attacker tampered with k_0 tags

Table 1. Probability that the attack succeeds according to the branching factor δ, given that k_0 tags have been opened and the system contains 2^{20} tags

k_0 \ δ	2	20	100	500	1000
1	66.6%	9.5%	1.9%	0.3%	0.1%
20	95.5%	83.9%	32.9%	7.6%	3.9%
50	98.2%	94.9%	63.0%	18.1%	9.5%
100	99.1%	95.4%	85.0%	32.9%	18.1%
200	99.5%	96.2%	97.3%	55.0%	32.9%

4.3 Notes on the Original Tree-Based Technique

In the original tree-based scheme proposed by Molnar and Wagner in [16], the blocks of secret of the tags were not chosen according to a uniform distribution. Instead, subtrees of a given level have the same set of blocks of secrets. This seems to be due to a typo in the setup algorithm of [16]. The attack is obviously more efficient on this original scheme because, the k_is are larger (for a same value of k_0).

5 Ohkubo, Suzuki, and Kinoshita's Protocol

5.1 Description

The protocol proposed by Ohkubo, Suzuki, and Kinoshita [18] relies on hash chains. When a tag is requested by a reader, it sends a hash of its current identifier and then renews it using a second hash function. Obviously, only the system is able to link all the values sent by the tag while an attacker cannot. More precisely, the scheme works as follows.

Setup. The personalization of a tag T_i consists of storing in its memory a random identifier s_i^1, which is also recorded in the system's database. Thus, the database initially contains the set of random values $\{s_i^1 \mid 1 \leq i \leq n\}$. Two hash functions G and H are chosen. One hash function is enough if a one-bit parameter is added to the function.

Interrogation. When the system queries T_i, it sends an identification request to the tag and receives back $r_i^k := G(s_i^k)$ where s_i^k is the current identifier of T_i. While T_i is powered, it replaces s_i^k by $s_i^{k+1} := H(s_i^k)$. The exchanges between the system and the tag are depicted on Figure 4.

Identification. From r_i^k, the system has to identify the corresponding tag. In order to do this, it constructs the hash chains from each n initial value s_i^1 until it finds the expected r_i^k or until it reaches a given maximum limit m on the chain length. The lifetime of the tag is *a priori* limited to m identifications. However, when a tag is scanned by a reader (in the library), its field in the database can

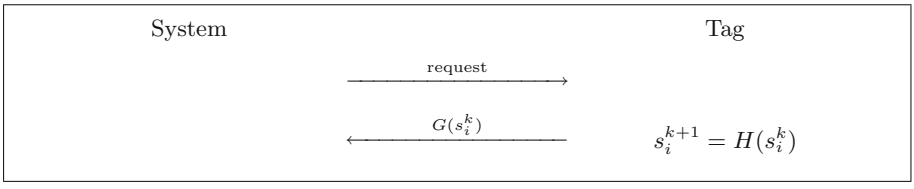

Fig. 4. Protocol of Ohkubo, Suzuki, and Kinoshita

be refreshed. The threshold m is therefore the number of read operations on a single tag between two updates of the database. A suited size for m could be 128, meaning that a tag can be queried 128 times before updating its entry in the database.

5.2 Replay Attack Avoidance and Reader Authentication

Like CR, OSK assures privacy because the information sent by the tag is indistinguishable from a random value, in the random oracle model. The main advantage of OSK compared with CR is that it also assures forward privacy, meaning that if an attacker can tamper with a tag, she is not able to track its past events. However, OSK does not prevent replay attacks. Common techniques to avoid replay attacks are usually incremental sequence number, clock synchronization, or a fresh challenge sent by the verifier. This latter option is the most suited to RFID tags. We propose therefore to modify OSK as depicted on Figure 5. Note that OSK does not provide authentication of the reader. However, this feature can be obtained if the system sends a third message containing $G(s_i^{k+1} \oplus w)$ where w is a fixed and public non-zero binary string.

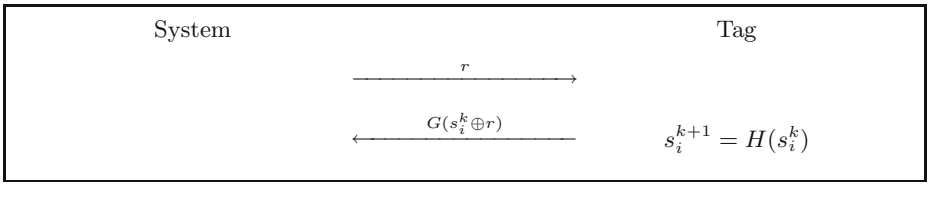

Fig. 5. Modified protocol of Ohkubo, Suzuki, and Kinoshita

5.3 Efficiency of OSK

Outside the library, tags can be queried by foreign readers. This avoids maintaining synchronization between the tag and the system. Therefore the complexity in terms of hash operations in order to identify one tag is $t_{\mathsf{OSK}} = mn\theta$ on average (2 hash operations are carried out $mn/2$ times). With the parameters given in Section 2 and chains of length 128, we have $t_{\mathsf{OSK}} \approx 16$ seconds. Note that if we had considered that readers of the library may read foreign tags (hold by people in the library), then the complexity would tend towards to $2mn$ because the

system would have to explore the whole database to determine whether or not a tag is owned by the system. Note that even if tags and readers were able to stay synchronized, the complexity of OSK cannot be better than CR if no additional memory is used.

6 Using a Time-Memory Trade-Off to Improve OSK

We recall and detail in this section our previous work [3] on OSK. Thus we will be able to compare CR/MW and OSK/AO.

6.1 Time-Memory Trade-Off

To reduce the complexity of OSK, we propose to improve how the data is managed by the system, without modifying the exchanges between tags and readers; so, the privacy of OSK remains. For that, we suggest to use a specific time-memory trade-off based on Hellman's original work [10] and Oechslin's optimizations [17]. This type of trade-off reduces the amount of work T needed to invert any given value in a set of N outputs of a one-way function F with help of M units of memory. The efficiency follows the rule $T = N^2\gamma/M^2$ where γ is a small factor depending on the probability of success and the particular type of trade-off being used. Compared to a brute-force attack, the trade-off can typically reduce the amount of work from N to $N^{2/3}$ using $N^{2/3}$ units of memory.

The basic idea of time-memory trade-off techniques consists in chaining (almost) all the possible outputs of F using a *reduction function* R which generates an arbitrary input of F from one of its outputs. By alternating F and R on a chosen initial value, a chain of inputs and outputs of F can be built. If enough chains of a given length are generated, most outputs of F will appear at least once in any chain. The trade-off comes from the fact that only the first and the last element of each chain is stored. Thus, a substantial memory space is saved, but computations will be required on-the-fly to invert a given element. Given one output r of F that should be inverted, a chain starting at r is generated. If r was part of any stored chain, a last element of a chain in the table will eventually be reached. Looking up the corresponding start of the chain, we can regenerate the complete chain and find the input of F that yields the given output r. To assure a high success rate, several tables have to be generated with different reduction functions. The exact way of doing this is what differentiates existing trade-off schemes.

In what follows, we will use the *perfect rainbow* tables [17] which have been shown to have better performances than other types of tables. The characteristic of the rainbow tables is that each column of a table has a different reduction function. So, when two chains collide, they do not merge (except if they collide at the same position in the chain). When the residual merged chains are removed during the precomputation step, the tables are said to be perfect. With such tables and a probability of success of 99.9%, we have $\gamma = 8$.

6.2 Adapting the Trade-Off to Our Case

The time-memory trade-off technique described above cannot be directly applied to our case. Indeed, the input of F must cover all the identifiers but no more.

Otherwise, the system would have no advantage over the attacker. Consequently, it is important to choose F such that its input space is as small as possible. We define the function F as follows:

$$F : (i, k) \mapsto r_i^k = G(H^{k-1}(s_i^1))$$

where $1 \leq i \leq n$ and $1 \leq k \leq m$. Thus, given the number of the tag and the number of the identification, F outputs the value which will be sent by the tag. We also need to define an arbitrary reduction function R such that

$$R : r_i^k \mapsto (i', k')$$

where $1 \leq i' \leq n, 1 \leq k' \leq m$. For example, we take

$$R(r) = (1 + (r \bmod n), 1 + (\left\lfloor \frac{r}{n} \right\rfloor \bmod m)).$$

There are still two important points that distinguish classical time-memory trade-offs from ours.

Firstly, the brute force method of OSK needs $n|s|$ units of memory to store the n values s_i^1 while usual brute-force methods do not require any memory. Thus, it makes sense to measure the amount of memory needed by the trade-off in multiples of $n|s|$. We call c the ratio between the memory used by the trade-off and the memory used by the brute-force. The memory used to store the tables is a multiple of the size of a chain while it is a multiple of s in the case of the brute-force. A stored chain is represented by its start and end points which can be either the output of F or its input. In our case the input is smaller, we therefore choose to store two pairs of (i, k), thus requiring $2(|n| + |m|)$ bits of memory. The conversion factor from units of brute-force to units of trade-off is $\mu = |s|/(2|n| + 2|m|)$. In the scenarios we are interested in, μ is typically between 2 and 4.

Secondly, when used in the trade-off, F is more complex than when used in the brute-force. Indeed, in the brute-force, the hash chains are calculated sequentially, thus needing just one H and one G calculation at each step. In the trade-off, i and k are arbitrary results from R and have no incremental relation with previous calculations. Thus, on average, each step computes $(m-1)/2+1$ times the function F and G once. We can now rewrite the trade-off relation:

$$T = \frac{N^2}{M^2}\gamma = \frac{n^2m^2}{(c-1)^2\mu^2n^2}(\frac{m-1}{2} + 1)\gamma \approx \frac{m^3\gamma}{2(c-1)^2\mu^2}.$$

We now show how this issue can be mitigated. So far, among the c shares of memory, $(c-1)$ shares are used to store the chains, and 1 share is used to store the n values s_i^1. If we not only store the first element of the chains, but also the element at the middle of the chain, we sacrifice even more memory but we reduce the average complexity of F. We will have only $(c-2)$ shares of the memory available for the tables, but F will have a complexity of $\frac{m-1}{4} + 1$ (we need to generate only a quarter of a chain on average). We have therefore a

trade-off between the memory sacrificed to store the intermediary points and the complexity of F. In general, if we store x values per chain, sacrificing x shares of memory, the complexity of the trade-off becomes:

$$T = \frac{n^2m^2}{(c-x)^2\mu^2n^2}(\frac{m}{2x}+1)\gamma \approx \frac{m^3\gamma}{2x(c-x)^2\mu^2}.$$

The optimal complexity is achieved when $x = \frac{c}{3}$. So we have

$$T \approx \frac{3^3}{2^3}\frac{m^3\gamma}{c^3\mu^2}.$$

Since a pair of (i,k) is 27 bits large (20 bits for i and 7 bits for k) we need at most 54 bits to store one chain. We can thus store more than two chains in the same amount of memory it takes to store one s ($\mu \geq 2$). Assuming that all calculations are carried out on a single back-end equipped with $\frac{c(n|s|)}{8} = 2^{24}c$ bytes of memory and that we choose a success rate of 99.9% ($\gamma = 8$) the time to read a tag with our method is

$$t_{\mathsf{OSK/AO}} \approx \frac{6^9\theta}{c^3} \text{ seconds.}$$

For example, with 1 GB of RAM (i.e., c=64), we have $t_{\mathsf{OSK/AO}} \approx 0.004$ milliseconds. Precomputation takes $nm^2\theta/2$ seconds, that is to say about 17 minutes. The technique used for that can be found in [3].

Note that the time-memory trade-off cannot be applied directly to the modified OSK suggested in Section 5.2. This is due to the randomization of the tag's answer. In order to apply our time-memory technique on the modified version of OSK, the tag must answer with both $G(s_i^k)$ and $G(s_i^k \oplus r)$. The former value enables the reader to identify the tag and the latter one allows detecting replay attacks.

7 Final Comparison and Conclusion

First of all, we consider the storage aspect. On the tag side, the storage of the identifiers becomes a real problem with CR/MW when δ is small. Having a large δ is therefore preferable. Storage is the main drawback of OSK/AO because precomputation and storage of tables is required. In the example given in Section 6, 1 GB of RAM is used. Today, such a memory is available on Joe Blow's computer.

Next, we address the complexity question. Both CR/MW and OSK/AO are parameterizable. CR/MW depends on δ which can be chosen between 2 and n. Obviously, the case $\delta = n$ leads to CR. Having $\delta > \sqrt{n}$ is possible but in this case the tree is no longer complete. Actually, a typical value could be $\delta = \sqrt{n}$. On the other hand, OSK/AO depends on the available memory. Table 2 gives a numerical comparison of CR, CR/MW, OSK, and OSK/AO.

We now consider the privacy issue. While CR is secure, CR/MW degrades the privacy because, when an attacker is able to tamper with at least one tag (e.g., her own tag), she is also able to trace other tags in a probabilistic way. We

Table 2. Time to identify one tag

Scheme (parameter)	Time (millisecond)
CR	62.500
CR/MW ($\delta = 2^{10}$)	0.122
CR/MW ($\delta = 2$)	0.002
OSK	16'000.000
OSK/AO (342 MB)	0.122
OSK/AO (1.25 GB)	0.002

have shown that the probability to trace tags decreases when the computation complexity grows. Thus, CR/MW can be seen as a trade-off between privacy and complexity. We proved that the probability to trace tags is far from being negligible. For example, when the branching factor is $\delta = 2^{10}$, the probability to trace a tag is about 0.1% when only one tag has been opened, but it is about 32.9% when 200 tags have been tampered with (see Table 1). OSK/AO inherits from the security proofs of OSK, in particular the fact that OSK is forward private, because it modifies neither the information exchanged, nor the content of the tag. It only improves the way the system manages and stores the data.

Thus, we can say that the main advantage of CR/MW rests on the fact that it does not require precomputation. Moreover the number of tag readings with OSK/AO is limited by the chain length while it is not with CR/MW (however overpassing this threshold does not threaten the privacy). Hence, when CR/MW is used, we recommend using a large branching factor in order to limit the privacy threat.

Finally, one may think that trade-off techniques could be used to improve CR/MW. Unfortunately, this seems difficult and cannot be done using the same approach because the answers of the tags in CR/MW are randomized. This implies carrying out a time-memory trade-off on a larger space.

Acknowledgment

The authors would like to thank Matthieu Finiasz and Serge Vaudenay for their helpful comments on this work, as well as David Molnar and Andrea Soppera. Gildas Avoine is supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

1. Gildas Avoine. Security and privacy in RFID systems. Online bibliography available at `http://lasecwww.epfl.ch/~gavoine/rfid/`.
2. Gildas Avoine and Philippe Oechslin. RFID traceability: A multilayer problem. In Andrew Patrick and Moti Yung, editors, *Financial Cryptography – FC'05*, volume 3570 of *Lecture Notes in Computer Science*, pages 125–140, Roseau, The Commonwealth Of Dominica, February – March 2005. IFCA, Springer-Verlag.

3. 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.
4. Steve Bono, Matthew Green, Adam Stubblefield, Ari Juels, Avi Rubin, and Michael Szydlo. Security analysis of a cryptographically-enabled RFID device. In *14th USENIX Security Symposium*, pages 1–16, Baltimore, Maryland, USA, July-August 2005. USENIX.
5. Susy d'Hont (Editor). International news letter of the TI RFID group. Electronic Newsletter, Issue 20, November 2000.
6. Electronic Product Code Global Inc. `http://www.epcglobalinc.org`.
7. Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In Marc Joye and Jean-Jacques Quisquater, editors, *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370, Boston, Massachusetts, USA, August 2004. IACR, Springer-Verlag.
8. Klaus Finkenzeller. *RFID Handbook*. Wiley, England, second edition, 2003.
9. Philippe Golle, Markus Jakobsson, Ari Juels, and Paul Syverson. Universal reencryption for mixnets. In Tatsuaki Okamoto, editor, *The Cryptographers' Track at the RSA Conference – CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 163–178, San Francisco, California, USA, February 2004. Springer-Verlag.
10. Martin Hellman. A cryptanalytic time-memory trade off. *IEEE Transactions on Information Theory*, IT-26(4):401–406, July 1980.
11. Dirk Henrici and Paul Müller. Tackling security and privacy issues in radio frequency identification devices. In Alois Ferscha and Friedemann Mattern, editors, *Pervasive Computing*, volume 3001 of *Lecture Notes in Computer Science*, pages 219–224, Vienna, Austria, April 2004. Springer-Verlag.
12. International Organization for Standardization. `http://www.iso.org`.
13. Ari Juels. Minimalist cryptography for low-cost RFID tags. In Carlo Blundo and Stelvio Cimato, editors, *The Fourth International Conference on Security in Communication Networks – SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 149–164, Amalfi, Italia, September 2004. Springer-Verlag.
14. Ari Juels and Ravikanth Pappu. Squealing euros: Privacy protection in RFID-enabled banknotes. In Rebecca Wright, editor, *Financial Cryptography – FC'03*, volume 2742 of *Lecture Notes in Computer Science*, pages 103–121, Le Gosier, Guadeloupe, French West Indies, January 2003. IFCA, Springer-Verlag.
15. Ari Juels, Ronald Rivest, and Michael Szydlo. The blocker tag: Selective blocking of RFID tags for consumer privacy. In Vijay Atluri, editor, *Conference on Computer and Communications Security – CCS'03*, pages 103–111, Washington, DC, USA, October 2003. ACM, ACM Press.
16. David Molnar and David Wagner. Privacy and security in library RFID: Issues, practices, and architectures. In Birgit Pfitzmann and Peng Liu, editors, *Conference on Computer and Communications Security – CCS'04*, pages 210–219, Washington, DC, USA, October 2004. ACM, ACM Press.
17. Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630, Santa Barbara, California, USA, August 2003. IACR, Springer-Verlag.

18. Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *RFID Privacy Workshop*, MIT, Massachusetts, USA, November 2003.
19. Keunwoo Rhee, Jin Kwak, Seungjoo Kim, and Dongho Won. Challenge-response based RFID authentication protocol for distributed database environment. In Dieter Hutter and Markus Ullmann, editors, *International Conference on Security in Pervasive Computing – SPC 2005*, volume 3450 of *Lecture Notes in Computer Science*, pages 70–84, Boppard, Germany, April 2005. Springer-Verlag.
20. SafeTzone. `http://www.safetzone.com`.
21. Stop RFID. `http://www.stoprfid.org`.
22. István Vajda and Levente Buttyán. Lightweight authentication protocols for low-cost RFID tags. In *Second Workshop on Security in Ubiquitous Computing – Ubicomp 2003*, Seattle, Washington, USA, October 2003.
23. Stephen Weis, Sanjay Sarma, Ronald Rivest, and Daniel Engels. Security and privacy aspects of low-cost radio frequency identification systems. In Dieter Hutter, Günter Müller, Werner Stephan, and Markus Ullmann, editors, *International Conference on Security in Pervasive Computing – SPC 2003*, volume 2802 of *Lecture Notes in Computer Science*, pages 454–469, Boppard, Germany, March 2003. Springer-Verlag.