An Almost-Optimal Forward-Private RFID Mutual Authentication Protocol with Tag Control

Paolo D'Arco

Dipartimento di Informatica
Università degli Studi di Salerno
I-84084 Fisciano (SA), Italy.

Abstract. In this paper we propose an efficient forward-private RFID mutual authentication protocol. The protocol is secure under standard assumptions. It builds over a recent work, extends it to achieve mutual authentication, and improves it by introducing a resynchronization mechanism between tag and reader, through which the server-side computation from $O(N\omega)$ is reduced to $O(N + \omega)$, where N is the total number of tags in the system, and ω is the maximum number of authentications each single tag can afford during its lifetime. Moreover, the protocol enables the server to control how many times a tag has been read by legitimate and fake readers.

1 Introduction

Rfid Technology: basics, development and concerns. The Rfid technology enables *automatic object identification* without the need for physical access. Each object is labeled with a tiny integrated circuit equipped with a radio antenna, called *tag*, whose *information content* can be received by another device, called *reader*, at a distance of several meters. Usually the readers are connected to a *back-end server*: they forward to the server the read tag content, and get back the result of the server computation. The interest of the scientific community for the Rfid technology has grown a lot during the last years simultaneously to the wide diffusion of the technology and the deployment of applications which partially deal or embed Rfid components. Indeed, the indubitable advantages come with new challenges: *security* and *privacy*, due to the constrained computational capabilities of the tags, are non trivial properties to achieve. If some applications do not need stringent security and privacy measures, applications which have an impact on the people life style, raise more concerns: in some settings as users (e.g., in access control applications, in anti-theft tools) we would like to be sure that a certain tag cannot be impersonated by an adversary; as well as, there are uses in which tracking features (e.g., postal tracking, pet tracking, airline luggage tracking, waste disposal tracking) are very welcome but others (e.g., when buying tag-equipped goods from a shop) in which we would like to be sure that our privacy is preserved, and no adversary is able to build a preference profile by illegally reading the content of the tags attached to the goods we buy.

State of art. We refer the interested reader to [14,13] for an overview of the applications of the RFID technology and of the main security issues, and to [1] for references to research papers dealing with RFID technology and its challenges[1].

Previous work. Roughly speaking, an Rfid authentication protocol enables tags and readers to be sure they are talking to each other, i.e., to identify and authenticate the other part. It is a key-component for building secure and private Rfid applications. An RFID authentication protocol is *forward-private* if an adversary, who tampers a tag and obtains its keys and state information, is unable to trace the tag, i.e., to associate the tag to previous transcripts of completed protocol executions he has eavesdropped. Obhuko et al. [15] proposed a simple and elegant forward-private scheme, which uses two hash functions. The scheme and its subsequent improvements, however, due to the costs of hash functions, are unsuitable for a real implementation on a tag. Moreover, such schemes are proven secure by using the random oracle methodology which is object of debate and criticism [2,6]. A recent paper [4] introduced a new Obhuko et al. like scheme, called PFP, which is efficient and is secure under standard assumptions, i.e., the existence of pseudorandom number generators and strongly universal hash function families.

Our contribution. In this paper we propose $EFPP$, a new forward-private RFID mutual authentication protocol. It builds over PFP and improves it by introducing a resynchronization mechanism between tag and reader, similar to the one used in [11,12], through which the server-side computation from $O(N\omega)$ is reduced to $O(N+\omega)$, where N is the total number of tags in the system, and ω is the maximum number of authentications each single tag can afford during its lifetime. Since the authors of [10], who focused on the design and the analysis of Rfid protocols based on symmetric-key primitives, showed that, if keys are chosen independently and uniformly at random, $\Omega(N)$ is a *lower bound* on the number of lookup operations the back-end server needs to authenticate a tag, then it follows that our forward-private scheme is *almost optimal*.

Related Work. Apart [4], which is our starting point, and [11,12], from which we borrow the resynchronization technique, other related papers are [17,5]. The $OFRAP$ mutual authentication scheme, proposed in [17], is elegant, efficient, and forward-private. It has been analyzed within the UC framework and proven secure and private under standard assumptions, i.e., the existence of pseudorandom functions. Moreover, it achieves an $O(N)$ overhead in terms of the number of lookup operations the server needs to authenticate a tag. Compared to ours, apart the computational tools, the main difference is that in $OFRAP$ the server has *no way* to control the total number of protocol executions a tag has been subject to, perhaps due to an adversary attack. In our scheme, on the other hand, we *gain* control by paying an additive ω factor within the asymptotic notation, which enables the back-end server to remove from the system a tag once its lifetime is over. On the other hand, the $PEPS$ scheme [5] also

[1] In the full version of the paper [8] are briefly mentioned the most significant efforts in order to provide *precise notions* of security and privacy, and to propose *efficient constructions*.

reduces the $O(N\omega)$ server-side computation of PFP [4] to $O(N)$. It has a design quite similar to $OFRAP$, and it is secure under the same assumptions. The main difference between $PEPS$ and $OFRAP$ is that $PEPS$ requires the tag to generate truly random numbers. We point out that also in $PEPS$ the server has *no way* to control the total number of protocol executions a tag has been subject to. In some applicative settings (e.g., access control, ticketing, automatic tolls ...) such a property is very welcome. The server could estimate the usage of the tag, as well as whether the tag has been target of attacks. $EFPP$ is a *true extension* of PFP, which is efficient and practical, and uses the same computational tools. To our knowledge, $EFPP$ is also the first efficient RFID forward-private authentication protocol enjoying the above tag-control feature.

2 Security Model

Every security model for evaluating Rfid authentication protocols focuses on three aspects: *correctness, security* and *privacy*. Loosely speaking, a protocol is correct if, with overwhelming probability, a legitimate tag and a legitimate reader successfully authenticate each other in an adversary-free protocol execution. Then, it is secure if an adversary has a negligible probability of impersonating a legitimate tag to the reader (vice versa, a reader to tag). Finally, it is private if a tag cannot be traced by analyzing the transcripts of protocol executions, and it is forward-private if the adversary does not succeed even if, at a certain point, gets access to the tag content and tries to trace the tag by using the transcript of previous completed executions.

Despite the properties we would like to get are intuitively clear, providing a suitable security and privacy model is a challenging task. Just to exemplify, the notion of correctness has to take into account a possible desynchronization attack tag and reader can be subject to at a certain point. What do we need to require from an adversary-free protocol execution *after* such an event has occurred? The models in [18,4,11] formalize this requirement in different ways. In this abstract, we do not deal with security model issues: since we basically use the same primitives of [4], for easiness of comparison, we refer to the same model (extended to deal with mutual authentication) which, as stated by the authors of [4], is a simplification and an adaptation of [18,16] to the symmetric setting[2].

The Model. Each tag T has an *internal state*, containing state information and secret keys. Tag secret keys are uncorrelated, chosen independently and uniformly at random. Part of the tag state is shared with the back-end server, which stores tag information in a database DB. Each tag can be used at most ω times. Readers are securely connected to the back-end server. During its lifetime, a tag enters authentication exchanges with the readers, following a *protocol* which specifies which messages have to be computed and exchanged, and how the internal states

[2] An analysis of the protocol in different models is left as future work.

of the tag and the back-end server have to be updated. An authentication exchange between a tag and a reader either results inside the reader (resp. tag) in an authentication success (together with a tag identity for the reader) or in an authentication failure. A tag cannot handle several authentication exchanges simultaneously. We assume that tags are exposed to an adversary during an exposure period, in which the adversary is able to observe and disturb all interactions involving the tag and possibly the reader, without confusing these interactions with exchanges involving other tags of the system. We also assume that no physical characteristics (e.g., radiation pattern, response time, et cetera) allow an adversary to recognize the tag and distinguish it from the other tags of the system, if the adversary observes it again in another exposure period.

Let A be an adversary with running time upper-bounded by T, allowed to trigger, observe, disturb and replace up to $q < \omega$ authentication exchanges involving the tag and the reader, and to access the outcome of the authentication protocol. We say that A is a (q, T)-adversary.

Definition 1. *An Rfid authentication protocol is said to be (q, T, ϵ)-correct iff the probability that a legitimate tag (resp. reader) is not successfully authenticated by a legitimate reader (resp. tag) in an undisturbed exchange at least once in its lifetime is upper-bounded by ϵ, even in the presence of a (q, T)-adversary. The probability is taken over the initial tag's secret values, the random numbers chosen during the protocol executions, and the random numbers chosen by the adversary.*

The definition states that a protocol is correct iff, even in presence of a (q, T)-adversary which tries to desynchronize tag and reader (i.e., so that they reach different states), the probability that in its lifetime there exists an adversary-free execution of the protocol in which the tag (resp. reader) is not authenticated by the reader (resp. tag), is at most ϵ. In other words, the protocol is robust against a (q, T)-adversary and it works almost always well.

Security requires resistance to impersonation attacks, which can be modeled as two-stage processes: during the first stage a (q, T)-adversary interacts both with a legitimate reader and a legitimate tag. During the second stage, the adversary only interacts with the reader (resp. tag) and initiates an authentication exchange to impersonate the tag (resp. the reader). The attack succeeds if the authentication is successful and the adversary is identified as the tag (resp. as the reader).

Definition 2. *An Rfid authentication protocol is said to be (q, T, ϵ)-secure (w.r.t. tag authentication/w.r.t. reader authentication) iff, for any (q, T)-adversary, the probability that an impersonation attack is successful is at most ϵ.*

The privacy requirement can be formalized through the following privacy experiment: during the first stage, a (q, T)-adversary A interacts with any two legitimate tags, T_0 and T_1, and a legitimate reader. At the end of this phase, a bit b (concealed to A) is chosen. Then, during the second stage, A again interacts

with T_b. Then, A is given access to the internal state of T_b. Eventually, A outputs a guess bit b' for the value b, and it succeeds if b' is equal to b.

Definition 3. *An Rfid authentication protocol is said to be (q, T, ϵ)-private iff any (q, T)-adversary A has an advantage at most ϵ in winning the privacy experiment, i.e.,*

$$|Pr[A \text{ succeeds }] - 1/2| \leq \epsilon.$$

Notice that, in the privacy experiment, we assume that A is given access to the internal state of the tag *when a protocol execution is completed*[3]. This precludes tag states following a failed protocol execution. The same approach was followed in [17] and, very recently, in [5], where the authors used the notion of *almost forward private protocol* to refer to the above setting with a restricted corruption capability for the adversary. However, we point out that, by using the same argument used in [16] (see Thm 1, page 294), it is possible to prove that if an adversary has the power to corrupt a tag *during* a protocol execution, then it easily wins the privacy experiment. Unfortunately, this issue *has no protocol solution without extra hardware assumptions* [16,9]. Therefore, the only possible goal is to look for a protocol which *safely locks* previous completed executions[4].

3 Tools

In this section we review some useful tools and properties needed to analyze the strength of the protocol. See [4] (Section 4 and the appendices) for proofs and details.

Let L, n and k be integers such that $L = n + k$, and let $g : \{0,1\}^n \to \{0,1\}^L$ be a binary function, which expands n-bit sequences into L-bit sequences. A distinguisher for g is a probabilistic algorithm A, which on input an L-bit sequence, outputs 0 or 1. The advantage of A in distinguishing g from a perfect random generator is defined as:

$$Adv_g(A) = |Pr[A(g(x)) = 1] - Pr[A(y) = 1]|$$

where the probabilities are taken over $x \in \{0,1\}^n$ (unknown to A) and $y \in \{0,1\}^L$, chosen uniformly at random, and over the random bits chosen by A. The advantage in distinguishing g in time T is:

$$Adv_g(T) = max_A\{Adv_g(A)\}$$

for all distinguishers A running in time at most T.

[3] Using the language of [11], the tag is *clean* at the corruption time, i.e., an undisturbed protocol execution with the reader has been successfully completed and the tag is ready for a new protocol execution.

[4] Notice that similar constraints to get perfect forward-secrecy in the key-exchange setting were shown in [3] (see Remark 7). It is the same problem which appears in two different settings.

Definition 4. *The function $g : \{0,1\}^n \to \{0,1\}^L$ is a (T,ϵ)-secure pseudorandom number generator ((T,ϵ)-PRNG, for short) iff $Adv_g(T) \leq \epsilon$.*

By using λ times the function g, we can define an iterated function G_λ. To this aim, let us denote $g(x) = g_1(x)||g_2(x)$, where $g_1(x) \in \{0,1\}^n, g_2(x) \in \{0,1\}^k$, and $||$ represents concatenation. Then, let λ be an integer greater than or equal to 1. The iterated function $G_\lambda : \{0,1\}^n \to \{0,1\}^{n+\lambda k}$ is defined by:

$$x \to (g_2(x), g_2(g_1(x)), \ldots, g_2(g_1^{\lambda-1}(x)), g_1^\lambda(x)).$$

Assuming that T_g is the time to compute g, it holds that:

Theorem 1. *If $g : \{0,1\}^n \to \{0,1\}^{n+k}$ is a (T,ϵ_g)-PRNG then, for any $\lambda \geq 1$, the associated iterated function G_λ is a $(T-(\lambda+1)T_g, \lambda\epsilon_g)$-PRNG.*

Similarly, a duplicated function $G^N : \{0,1\}^{nN} \to \{0,1\}^{LN}$ is simply defined by $(x_1, \ldots, x_N) \to (G(x_1), \ldots, G(x_N))$. It holds that:

Lemma 1. *If G is a (T',ϵ_G)-PRNG, then G^N is a $(T', N\epsilon_G)$-PRNG.*

Finally, a duplicated iterated function $G_\lambda^N : \{0,1\}^{nN} \to \{0,1\}^{(n+\lambda k)N}$ is defined by $(x_1, \ldots, x_N) \to (G_\lambda(x_1), \ldots, G_\lambda(x_N))$.

Theorem 1 and Lemma 1 were proven in [4] by using standard hybrid arguments. From them, it follows that:

Theorem 2. *For any (T,ϵ_g)-PRNG $g : \{0,1\}^n \to \{0,1\}^{n+k}$, any $\lambda \geq 1$, and any $N \geq 1$, the associated duplicated function $G_\lambda^N : \{0,1\}^{nN} \to \{0,1\}^{(n+\lambda k)N}$ is a $(T-(\lambda+1)T_g, N\lambda\epsilon_g)$-PRNG.*

The second key-tool we need in our construction are function families with special uniformity properties, referred to as universal classes of hash functions [7]. The idea of a universal class of hash functions is to define a collection $\mathcal{H}$ of hash functions in such a way that a random choice of a function $h \in \mathcal{H}$ yields a low probability that any two distinct inputs x and y will collide when their hashed values are computed using the function h. A more structured function family is defined as follows:

Definition 5. *A family $\mathcal{H} = \{h_s : \{0,1\}^\ell \to \{0,1\}^m\}$ of hash functions is called ϵ-almost strongly universal if and only if: $\forall a \in \{0,1\}^\ell, \forall b \in \{0,1\}^m$, it holds that $Pr_{s\in S}[h_s(a) = b] = 2^{-m}$, and $\forall a_1 \neq a_2 \in \{0,1\}^\ell, \forall b_1, b_2 \in \{0,1\}^m$, it holds that $Pr_{s\in S}[h_s(a_2) = b_2 | h_s(a_1) = b_1] \leq \epsilon$.*

Notice that, the first condition states that any input a is mapped to any hashed value b with probability $\frac{1}{2^m}$. The second states that, given that a_1 is mapped to b_1, the conditional probability that a_2 is mapped to b_2, for any $a_2 \neq a_1$, is at most ϵ. A 2^{-m}-almost strongly universal hash function family $\mathcal{H}$ is called a *strongly universal* hash function family. Further details and applications can be found in [20,19].

The following lemma was proven in [4]. It states that an adversary who knows a pair $(a_0, h_s(a_0))$ and a bunch of pairs (a_j, b_j), with $b_j \neq h_s(a_j)$, has a small probability of guessing the correct value of the function $h_s(a)$ on a new randomly chosen value a.

Lemma 2. *Let $\mathcal{H} = \{h_s : \{0,1\}^\ell \to \{0,1\}^m\}$ be an ϵ-almost strongly universal hash function family, let s^* be a (secret) value randomly chosen in S, and let A be a computationally unbounded adversary who tries to predict the value of h_{s^*} on a randomly chosen input value a. Suppose that A is given at most one pair (a_0, b_0) and at most $p \leq \frac{1}{2\epsilon}$ pairs (a_j, b_j) such that $h_{s^*}(a_0) = b_0$ and, for $0 < j \leq p$, it holds that $h_{s^*}(a_j) \neq b_j$. Then,*

$$Pr_{a \in \{0,1\}^\ell, s^*}[A(a) = h_{s^*}(a)] = 2^{-\ell} + \epsilon(1 + 2p\epsilon).$$

Lemma 3. *If s, s' are chosen independently, then $Pr_{s,s' \in S}[h_s(a) = h_{s'}(a)] = 2^{-m}$.*

In the following, we will denote with T_h the time to compute the function h_s.

4 Protocol Description

In this section we introduce our protocol. Let us briefly describe PFP, the forward-private protocol proposed in [4] we start from. Let $g : \{0,1\}^n \to \{0,1\}^{n+k}$ be a PRNG, and let $\mathcal{H} = \{h_s : \{0,1\}^\ell \to \{0,1\}^m\}$ be a strongly universal hash function family. Moreover, let $g(x) = g_1(x)||g_2(x)$, where $g_1(x) \in \{0,1\}^n$ and $g_2(x) \in \{0,1\}^k$. Each tag can be used at most ω times. It stores the description of g and $\mathcal{H}$, and a state variable σ. The back-end server stores the same information for all tags in its database. The protocol works as follows:

1. The reader chooses uniformly at random a challenge $c \in \{0,1\}^n$ and sends c to the tag.
2. The tag, receiving c, *updates* its state σ and *chooses* a random function h_s from $\mathcal{H}$ by computing $(\sigma, s) = (g_1(\sigma), g_2(\sigma))$. Then, it computes $r = h_s(c)$, and sends r to the reader.
3. The reader, for each tag T, fetches into the database DB the last known state for tag T, say σ_j^T, and checks whether there exists an index $i \geq 0$ such that $j + i < \omega$ and $h_{g_2(g_1^i(\sigma_j^T))}(c) = r$. If such an index is found, then the tag is authenticated. Otherwise, it is refused.

In other words, at each protocol execution, the reader checks in DB along chains of at most ω elements if a match is found. The protocol is correct, secure and forward-private and, in a system with N tags, it has complexity $O(N\omega)$. In the following we show how to improve the scheme in order to get *mutual authentication* and to *reduce the complexity* from $O(N\omega)$ to $O(N + \omega)$.

Let us start by describing the information held by tags and readers in the new protocol.

Common public information: two d-bit values pad_1, pad_2, used for padding, and the descriptions of a pseudorandom number generator g (PRNG) and of a strongly universal hash function family H (SUHF, for short). The PRNG g is used for *identification* purposes, for *updating* tag information, and within the *authentication* process. The SUHF H is used within the *authentication* process.

As before, we split the values of $g : \{0,1\}^n \rightarrow \{0,1\}^{n+k}$ in two parts, i.e., $g(x) = g_1(x)||g_2(x)$.

Tag information: a (n-d)-bit randomizer CR, an identification key k, a state variable σ, the two d-bit values pad_1 and pad_2, and the descriptions of g and H

Reader information: a DB which stores the description of g and H, the two values pad_1 and pad_2, and, for each tag, the tag identifier ID, a counter CNT_{ID}, and two tuples: $< I_{old}, \sigma_{old}, k_{old}, CR_{DB}^{old} >$ and $< I, \sigma, k, CR_{DB} >$. Let us denote by $DB_{ID}[i]$, for $i = 0, 1$, the memory locations for the two tuples for tag ID. At the beginning, when DB is initialized, all counters and old tuples are set to zero, i.e., $CNT_{ID} = 0$ and $DB_{ID}[0] =< 0, 0, 0, 0 >$ for all N tags. The DB automatically removes tag ID when $CNT_{ID} = \omega$. Moreover, let us denote by $\sigma_{|n-d}$ the first $n - d$ bits of σ, and by rnd a value chosen uniformly at random.

Three-round authentication protocol overview:

1. The reader chooses uniformly at random a challenge $c \in \{0,1\}^n$ and sends c to the tag.
2. The tag updates its state, and computes and sends to the reader a *triple*, $(I, v_T, auth)$. The first two entries are used for identification and (if the synchronization is lost) to resynchronize tag and reader. More precisely, the value I can be seen as a sort of *pseudonym*, which changes at each invocation, while the value v_T contains information about the *current randomizer* CR of the tag. Finally, the value $auth$ is the *authenticator*, used to authenticate the tag to the reader.
3. The reader, once received $(I, v_T, auth)$, looks in DB for a tuple starting with pseudonym I. If a tuple is found, the reader checks the received values are computed correctly from the tag, overwrites the old tuple for the tag with the current tuple, updates the current tuple $< I, \sigma, k, CR_{DB} >$, and sends to the tag a value which acknowledges the received triple and authenticates the reader to the tag. Otherwise, it first tries to resynchronize with the tag and, then, it does the same check and update. If fails then it sends a random value to the tag.
4. The tag checks whether the received value is equal to the value it is expecting to receive and, accordingly, updates its key and outputs 1 or outputs 0.

A complete description of the protocol, referred to as $EFPP$, is given in Figure 1. The subroutines tag and reader invoke are described below.

Compute(c)	Verify($ID, c, I, v_T, auth$)
$I = g_1((CR\|\|pad_1) \oplus k)$	$s = g_2(\sigma)$
$(v_0, v_1) = g((c \oplus I) \oplus k)$	if $auth \neq h_s(c \oplus I \oplus v_T)$ then return $(rnd, 0)$
$v_T = (CR\|\|pad_2) \oplus v_0$	$(v_0, v_1) = g((c \oplus I) \oplus k)$
$CR = CR + \sigma_{\|n-d}$	if $(CR_{DB}\|\|pad_2) \neq v_T \oplus v_0$ then return $(rnd, 0)$
$(\sigma, s) = (g_1(\sigma), g_2(\sigma))$	return $(v_1, 1)$
$auth = h_s(c \oplus I \oplus v_T)$	
return $(v_1, I, v_T, auth)$	

Protocol steps:

1. Reader: chooses $c \in \{0,1\}^n$ uniformly at random, and sends c to the Tag.
2. Tag: sets $(v_1, I, v_T, auth) = \mathsf{Compute}(c)$ and sends $(I, v_T, auth)$ to the Reader.
3. Reader: if there exists a tuple $tp = (I, \sigma, k, CR_{DB})$ in DB for tag ID
   (a) computes $(v_1, b) = \mathsf{Verify}(ID, c, I, v_T, auth)$
   (b) if $(b == 1)$ then invokes $\mathsf{Update}(ID, tp)$ and outputs 1; else outputs 0
   (c) sets $v_R = v_1$ and sends v_R
   else, if no such a tuple exists, then
   (a) sets $(CR_N, ID) = \mathsf{Lookupkey}(I, c, v_T)$ and $d = \mathsf{Resynch}(CR_N, ID)$
   (b) if $(d == 0)$ then sets $v_1 = rnd$;
   else $(v_1, b) = \mathsf{Verify}(ID, c, I, v_T, auth)$
   if $(b == 1)$ then $\mathsf{Update}(ID, ID[1])$ and outputs 1; else outputs 0
   (c) sets $v_R = v_1$ and sends v_R
4. Tag: if $v_R == v_1$, then sets $k = g_1(k)$ and outputs 1; else outputs 0

Fig. 1. EFPP: Efficient Forward-Private Protocol.

$\mathsf{Update}(ID, tp)$
$CNT_{ID} = CNT_{ID} + 1$
$DB_{ID}[0] = tp$
$CR_{DB} = CR_{DB} + \sigma_{|n-d}$
$\sigma = g_1(\sigma)$
$k = g_1(k)$
$I = g_1((CR_{DB}||pad_1) \oplus k)$

$\mathsf{Lookupkey}(I, c, v_T)$	$\mathsf{Resynch}(CR_N, ID)$
look in DB for a key k for which	if $(0,0)$ then return 0
$(v_0, v_1) = g((c \oplus I) \oplus k)$ are such that	while $(CR_{DB} \neq CR_N$ and $CNT_{ID} < \omega)\{$
$v_T \oplus v_0 = (CR_N||pad_2)$	$CR_{DB} = CR_{DB} + \sigma_{|n-d}$
and $g_1((CR_N||pad_1) \oplus k) = I$	$\sigma = g_1(\sigma)$
if no k exists, then return$(0,0)$	$CNT_{ID} = CNT_{ID} + 1$
else $DB_{ID}[1] =< I, \sigma, k, CR_{DB} >$	$\}$
return(CR_N, ID)	if $CNT_{ID} == \omega$ then return 0
	else return 1

5 Properties

The protocol enjoys several properties. Before going through formal proofs, and in order to get the ideas underlying the design, we provide some observations.

– Desynchronization attacks. An adversary might attack the system by sending multiple challenges c to the tag. In such a way, the tag updates CR and σ, which become different from CR_{DB} and σ stored in DB. However, notice

that the identification key k, *stays the same*: it is updated *only* after a successful execution with the reader. In such a way, tag and reader, at the first adversary-free execution, recover the same value of CR and resynchronize their states. Similarly, an adversary might discard the last message from the reader to the tag. Again, the attack fails since the server stores new and old tag records in DB.

- Computational efficiency. PFP, to authenticate the tag to the reader, requires $O(N\omega)$ iterations of the PRNG g. Our scheme requires $O(N+\omega)$ iterations of g. The small extra-amount of tag-side computation at each protocol execution consistently reduces the server-side computation.
- Forward-privacy. If an adversary corrupts the tag *after* a successful execution with the reader has occurred, he gets the current state and identification key. The assumptions on g (i.e., it is a PRNG) guarantee the property[5].

6 Security Reductions

In this section we show that the protocol is private, secure and correct.

Privacy. We prove that the privacy property holds by showing that if there exists an adversary A_p which wins the privacy experiment for our protocol, then there exists an adversary B_p which wins the privacy experiment for PFP with the same advantage. Since PFP is private, we conclude that our protocol is private, too. Essentially B_p uses *the context of his privacy experiment*, to simulate *the context of the privacy experiment for adversary* A_p, which works against our protocol. Hence, we need to show *how* B_p simulates the context for A_p and *why* A_p does not distinguish the simulated context from the real one. Adversary B_p works as follows:

- B_p starts the simulation of 2 'augmented' tags, T'_0 and T'_1, for the adversary A_p, by using the real tags T_0 and T_1 of the privacy experiment for PFP he is interacting too.
- B_p runs A_p. B_p answers correctly all A_p's queries relaying modified queries to the tags and extending tags replies in phase 1. The modified queries and replies are constructed as follows: if A_p asks the reader to start a new execution, then B_p chooses a uniformly at random c and sends it to A_p. If A_p sends c to the tag, then B_p *chooses uniformly at random* the values I and v_T, and computes $c' = c \oplus I \oplus v_T$. Then sends c' to the tag and gets back $h_s(c')$. Hence, constructs the triple $(I, v_T, h_s(c'))$, stores it in a database of simulated transcripts $STDB$, and sends it to A_p. If A_p sends the triple to the reader, then B_p checks if the triple is in $STDB$, simulates acceptance of the reader and sends (and stores in $STDB$) v_1, *chosen uniformly at random*, to

[5] It is easy to see that, if an Adv corrupts the tag, for example, after sending it a challenge, then he gets the identification key k (which stays the same as long as an adversary-free execution tag/reader does not occur) and can trace the tag by re-computing the pseudonym I. As we have stressed before, such corruptions *during or after failed executions* are precluded by the model.

A_p. If A_p sends v_1 to the tag, B_p checks in $STDB$ and simulates acceptance by the tag. Therefore, B_p (using the real tags) is able to provide a *partially simulated* transcript to A_p.
- Let b be the bit chosen by the privacy experiment runner R for PFP, and let T_b be the target tag. Since B_p is simulating the privacy experiment for A_p, implicitly the choice of R holds for B_p. (Let us assume that R just removes from the scene $T_{b\oplus 1}$.) Then, B_p keeps going with the simulation described before and, when A_p asks to corrupt T'_b, then B_p corrupts T_b, and forwards to A_p the state σ (read in T_b memory) and values *chosen uniformly at random* for CR and k to complete the amount of information which would be stored in a real tag T'_b. Finally, B_p outputs the same bit b' that A_p outputs.

B_p defeats the privacy of PFP exactly with the same probability with which A_p defeats the privacy of $EFPP$. What is left in the proof is to show that A_p *does not distinguish* the simulated transcript from a real transcript.

To this aim, notice that the transcript of a protocol execution is given by tuples of the form $(c, I, v_T, auth, v_1)$, where c is chosen uniformly at random, I, v_T and v_1 are computed through the PRNG g, and $auth$ is computed through the strongly universal hash function h_s, plus CR and k, obtained by corrupting the tag *after* a successful protocol execution. Let H_0, H_1, H_2 and H_3 (hybrid) distributions of tuples defined as follows:

- H_0 contains tuples $(c, I, v_T, auth, v_1)$ computed like in the real protocol
- H_1 contains tuples $(c, I, v_T, auth, v_1)$ where I is chosen uniformly at random
- H_2 contains tuples $(c, I, v_T, auth, v_1)$ where I and v_T are chosen uniformly at random
- H_3 contains tuples $(c, I, v_T, auth, v_1)$ where I, v_T and v_1 are chosen uniformly at random

Notice that H_3 is the distribution of sequences produced in our simulation by B_p. By showing that, for $i = 0, 1, 2$, it holds that H_i is indistinguishable from H_{i+1}, we infer that H_0 is indistinguishable from H_3. To show that, for $i = 0, 1, 2$, it holds that H_i is indistinguishable from H_{i+1}, we use the same technique: if there exists a distinguisher D_H between the two hybrids, then there exists a distinguisher D_g which distinguishes the outputs of the PRNG from truly random values. Let us report the proof for the first case.

Let H_i and H_j be two distributions over sequences of m tuples. We say that H_i and H_j are (T, ϵ)-indistinguishable, iff $Adv_{D_{H_i,H_j}}(T) \leq \epsilon$, for any distinguisher D_{H_i,H_j} running in time at most T.

Lemma 4. *If g is a (T, ϵ_g)-secure PRNG, then H_0 and H_1 are (T_i, ϵ_i)-indistinguishable where $T_i = T - 3(m-1)T_g - (m-1)T_h$ and $\epsilon_i = \epsilon_g m^2$.*

Proof. Let $H_{0,0}$ be a sequence of m tuples, generated according to distribution H_0, and let $H_{0,m}$ be a sequence of m tuples, generated according to distribution H_1. Moreover, for $i = 1, \ldots, m-1$, let $H_{0,i}$ be a sequence of m tuples, generated by choosing in the first i tuples the value I uniformly at random, and

the remaining $m - i$ as in H_0. If there exists a (T_H, ϵ_H) distinguisher D_{H_0,H_1} for the distributions H_0 and H_1, then there exists an index i such that D_{H_0,H_1} distinguishes between $H_{0,i}$ and $H_{0,i+1}$ with advantage $\geq \epsilon_H/m$. We construct a distinguisher D_g, by using D_{H_0,H_1} as a subroutine, as follows:

- Let V be the challenge-value D_g has to decide (PRNG output or Random).
- Chooses uniformly at random an index $i \in \{1, \ldots, m-1\}$.
- Constructs a sequence of tuples H_c by following the distribution $H_{0,i}$.
- Substitutes in the i-th tuple the value of I with the challenge-value V and provides H_c to D_{H_0,H_1}.
- If D_{H_0,H_1} outputs 0, then D_g outputs 0. Else D_g outputs 1.

Notice that, when $V = g_1((CR||pad_1) \oplus k)$ then $H_c = H_{0,i}$. On the other hand, when V is a random value, then $H_c = H_{0,i+1}$. It follows that D_g distinguishes the output of the PRNG from a random value with advantage $\geq \epsilon_H/m^2$. Moreover, D_g has running time upper bounded by $T_H + 3(m-1)T_g + (m-1)T_h$. Since by assumption g is (T, ϵ_g)-secure, if $T_H + 3(m-1)T_g + (m-1)T_h < T$, we get that $\epsilon_H/m^2 \leq \epsilon_g$, from which it follows that H_i and H_j are (T_i, ϵ_i)-indistinguishable, where $T_i = T - 3(m-1)T_g - (m-1)T_h$ and $\epsilon_i \leq \epsilon_g m^2$. $\triangle$

Similarly, we can show that if A_p distinguishes CR and k of the simulated transcript from CR and k obtained by opening a tag after a real successful protocol execution, then we can construct a distinguisher for the PRNG g. It follows that A_p does not distinguish a simulated transcript from a real one and we conclude that:

Theorem 3. *If PFP is (q, T, ϵ_p)-private, then EFPP is (q, T, ϵ_p)-private.*

Secure tag authentication. Notice that, restricting the attention to the first two rounds, our protocol generalises PFP. The first round is the same. In the second, the tag sends a triple (containing a value of h_s) instead of a single value h_s. Along the same line of the proof of [4], we show that, if there exists an efficient adversary A, which is able to impersonate a tag to the reader, then there exists an efficient distinguisher B capable of distinguishing outputs of the PRNG G_ω from random values in $\{0,1\}^{\omega k}$. By suitably choosing the PRNG g, we show that the probability with whom B (and hence A) succeeds is small. More precisely, we construct B as follows:

- B receives in input the sequence of values $z_1, \ldots, z_\omega$ it has to decide from which source it comes from.
- Then, it uses the above values $z_1, \ldots, z_\omega$ to simulate the computations of the tag T (of unknown state) and the reader with whom A is supposed to interact. More precisely, B answers all A's queries in phase 1 as follows: if A asks the reader to start a new execution, then B chooses a uniformly at random c and sends it to A. If A sends c to the tag then, assuming it is the i-th execution, B chooses uniformly at random the values I, v_1 and v_T, sets $s = z_i$, and computes and sends to A the triple $(I, v_T, h_s(c \oplus I \oplus v_T))$. It also stores the triple and v_1 in $STDB$. If A sends the triple to the reader, then

B checks in $STDB$ whether there exists an entry which matches the triple, simulates acceptance of the reader and sends v_1 to A. If A sends v_1 to the tag, B checks in $STDB$ and simulates acceptance of the tag.
- Let c be the challenge on which A, in phase 2, tries to impersonate T. B gets back from A the triple $(I, v_T, h_s(c \oplus I \oplus v_T))$. B checks whether $h_s(c \oplus I \oplus v_T) = h_{z_{q+1}}(c \oplus I \oplus v_T)$ (A has interacted $q < \omega$ times with tag and reader in phase 1) and, if the check is satisfied then accepts and outputs 1; otherwise, it outputs 0.

Notice that, if $z_1, \ldots, z_\omega$ is pseudorandom (let us denote it as Z_{G_ω}), then B outputs 1 with probability p_A. Indeed, it is possible to show that the transcript of the simulated executions is *indistinguishable* from the transcript of real executions and, by assumption, on real transcripts, A impersonates the tag with probability p_A. The indistinguishability can be shown by using standard arguments: if A distinguishes between the transcripts, then can be constructed an efficient distinguisher for g. On the other hand, following the reasoning of [4] and applying Lemma 2, simple computations show that, if $z_1, \ldots, z_\omega$ are truly random values (let us denote them as Z_U), the probability that B outputs 1 is less than $\omega(2^{-\ell} + \epsilon(1 + 2q\epsilon))$. It follows that:

$$|Pr[B(Z_{G_\omega}) = 1] - Pr[B(Z_U) = 1]| \geq p_A - \omega(2^{-\ell} + \epsilon(1 + 2q\epsilon)).$$

However, if g is a (T, ϵ_g)-secure PRNG, applying Theorem 1, we get that the advantage $|Pr[B(Z_{G_\omega}) = 1] - Pr[B(Z_U) = 1]| \leq \omega\epsilon_g$. The last two equalities show that $p_A \leq \omega(\epsilon_g + 2^{-\ell} + \epsilon(1 + 2q\epsilon)$. Moreover, B's running time is equal to A's running time plus q computations of h_s for the tag simulation. Therefore, we can conclude that:

Theorem 4. *If $\mathcal{H}$ is an ϵ-almost strongly universal hash function family, g is a (T, ϵ_g)-secure PRNG, and $q \leq 1/2\epsilon$, then EFPP is (q, T', ϵ_s)-secure (w.r.t tag authentication) with $T' = T - (\omega + 1)T_g - qT_h$ and $\epsilon_s = \omega(\epsilon_g + 2^{-\ell} + \epsilon(1 + 2q\epsilon))$.*

Secure reader authentication. An adversary A, to be authenticated as reader from the tag T, has to send in the third round of the protocol the right value v_1 to T. By using the same argument and simulation we have used before we show that, if there is an efficient A who guesses v_1 with probability p_A, then there exists a distinguisher B which distinguishes outputs of the PRNG G_ω from random values in $\{0,1\}^{\omega k}$, and then, by suitably choosing the PRNG g, we show that the probability p_A is small. The distinguisher B uses A as a subroutine and simulates A's interaction with tag and reader. Eventually, if A impersonates the reader, then B outputs 1. Otherwise, if A fails, then B outputs 0. When the sequence $z_1, \ldots, z_\omega$ is chosen uniformly at random, then B outputs 1 with probability at least $\frac{1}{2^k}$. On the other hand, if $z_1, \ldots, z_\omega$ is pseudorandom, then, B outputs 1 with probability p_A. Indeed, as argued before, the simulated values received by A from B are indistinguishable from the values of real executions with the tag T and, by assumption, on real transcripts, A impersonates the reader with probability p_A. It follows that:

$$Adv_{G_\omega}(B) = |Pr[B(Z_{G_\omega}) = 1] - Pr[B(Z_U) = 1]| \geq p_A - 1/2^k$$

Due to Theorem 1, if g is a (T, ϵ_g)-secure PRNG, then it follows that $Adv_{G_\omega}(B) \leq \omega\epsilon_g$. Hence, it holds that $p_A \leq 1/2^k + \omega\epsilon_g$. In conclusion:

Theorem 5. *If g is a (T, ϵ_g)-secure PRNG, then EFPP is $(q, T', \omega\epsilon_g)$-secure (w.r.t. reader authentication) where $T' = T - (\omega + 1)T_g - qT_h$.*

Correctness. In an adversary-free execution, a tag T (resp. reader) is not authenticated by the reader (resp. tag), only if the reader *updated twice* the tuple associated to the tag in the database DB and the tag did not or the tag updated its secret key k and the reader did not. Such events happen only if

1. The adversary is able to impersonate the tag T (resp. the reader).
2. Collisions of g and h occur.

Due to the security of the scheme, as we have seen before, the first possibility happens with small probability. Hence, we do not need to care about it. Regarding the second, collisions of g and h, we need to consider two separate cases: during an execution of the protocol, at a certain point, there exists a tuple in DB associated to *another* tag either with the same I and matching equations or with a different I' but a secret key k by means of which we get a collision on c, I, v_T, and $auth$. More precisely, in the first case there exists in DB a tuple $(I, k', \sigma', CR'_{DB})$, associated to tag $ID' \neq ID$, for which $(v_0, v_1) = g(c \oplus I \oplus k')$ are such that:

$$v_T \oplus v_0 = CR'_{DB}||pad_2 \bigwedge g_1((CR'_{DB}||pad_1) \oplus k') = I \bigwedge h_{s'}(c \oplus I \oplus v_T) = auth$$

while, in the second, there exists a tuple $(I', k', \sigma', CR'_{DB})$ for which $(v_0, v_1) = g(c \oplus I \oplus k')$ are such that:

$$v_T \oplus v_0 = CR_N||pad_2 \bigwedge g_1((CR_N||pad_1) \oplus k') = I \bigwedge h_{s'}(c \oplus I \oplus v_T) = auth$$

Notice that $g_1((CR'_{DB}||pad_1) \oplus k') = I$ implies that $g_1((CR_{DB}||pad_1) \oplus k) = g_1((CR'_{DB}||pad_1) \oplus k') = I$ i.e., g produces a collision. If g is a (T, ϵ_g)-secure PRNG, then it produces collisions with probability less than ϵ_g. Otherwise, it would possible to construct a simple distinguisher for g which distinguishes pseudorandom values from truly random values with probability higher than ϵ_g. Moreover, due to Lemma 3, the equality $h_s(c \oplus I \oplus v_T) = h_{s'}(c \oplus I \oplus v_T) = auth$, for $s \neq s'$, occurs with probability $1/2^m$. A similar analysis applies to the second case. In conclusion, it holds that:

Lemma 5. *If $\mathcal{H}$ is an ϵ-almost strongly universal hash function family and g is a (T, ϵ_g)-secure PRNG, a collision during an execution of the protocol occurs with probability $< 2 \cdot \epsilon_g/2^m = \epsilon_g/2^{m-1}$.*

We need to consider the probability of collisions within the lifetime of the protocol. If the system has N tags, since each tag can be used at most ω times, the protocol is useful for at most $N\omega$ authentications. By using the above result, we get that the probability of a collision within the system is $p_c \leq (N-1)\omega^2\epsilon_g/2^{m-1}$.

Let g be a (T, ϵ_g)-secure PRNG, where $T \geq (N-1)\omega^2 T_h + (\omega+1)T_g$. Theorem 2 shows that the PRNG G_ω^N, constructed from g (which models tag state updates and the generation of seeds for h_s,) is an $((N-1)\omega^2 T_h, N\omega\epsilon_g)$-secure PRNG. Applying the same steps of [4], we can conclude that the probability of failure of the protocol is $p < (N-1)\omega^2\epsilon_g/2^{m-1} + N\omega\epsilon_g + \epsilon_s$, where ϵ_s is the probability of impersonation. In conclusion:

Theorem 6. *Let g be a (T, ϵ_g)-secure PRNG where $T \geq (N-1)\omega^2 T_h + (\omega+1)T_g$, let $\mathcal{H}$ be an ϵ-almost strongly universal hash function family, and let $q \leq 1/2\epsilon$. The EFPP authentication protocol is (q, T', ϵ_c)-correct, with $T' = T - (\omega + 1)(3T_g + qT_h)$ and $\epsilon_c = (N-1)\omega^2\epsilon_g/2^{m-1} + N\omega\epsilon_g + \epsilon_s$.*

7 Acknowledgment

The work described in this paper has been supported in part by the European Commission through the ICT program under contract 216676 ECRYPT II, in part by the Italian Ministry of University and Research - Project PRIN 2008 PEPPER: Privacy and Protection of Personal Data (prot. 2008SY2PH4), and in part by Project MTM2010-15167 from the Spanish Ministry of Science and Technology.

8 Conclusions

We have proposed an efficient forward-private RFID mutual authentication protocol, secure under the assumption that exist secure pseudorandom number generators and strongly universal hash function families. At each authentication, compared to PFP where the tag computes one time the PRNG g and one time the hash function h_s, the tag has to apply 3 times g and one time h_s. On the other hand, the server, to authenticate a tag, in the worst case, instead of $O(N\omega)$ evaluations of g and h_s as in PFP, needs only $O(N+\omega)$ evaluations, where N is the total number of tags in the system, and ω is the maximum number of authentications each single tag can afford during its lifetime. The server has full control over the number of protocol executions a tag has been subject to. The full version of this paper [8] reports an experimental comparison of PFP vs $EFPP$, obtained by implementing the protocols.

References

1. G. Avoine, *RFID Security and Privacy Lounge*, http://www.avoine.net/rfid/
2. M. Bellare and P. Rogaway., *Random oracles are practical: A paradigm for designing efficient protocols*, Proceedings of the First Annual Conference on Computer and Communications Security, ACM, 1993.
3. M. Bellare, D. Pointcheval, and P. Rogaway, *Authenticated key exchange secure against dictionary attacks*, Proceedings of Eurocrypt 2000, Lecture Notes in Computer Science, Vol. 1807, pp. 139-155, 2000.

4. C. Berbain, O. Billet, J. Etrong, and H. Gilbert, *An Efficient Forward Private RFID Protocol*, 16th ACM Conference on Computer and Communications Security (CCS 09), pp. 43 - 53, 2009.
5. O. Billet, J. Etrog, and H. Gilbert, *Lightweight Privacy Preserving Authentication for RFID Using a Stream Cipher*, Proceedings of Fast Software Encryption (FSE 2010), Lecture Notes in Computer Science, Vol. 6147, pp. 55-74, 2010.
6. R. Canetti, O. Goldreich and S. Halevi. *The Random Oracle Methodology, Revisited*, Proceedings of ACM STOC, 1998.
7. J. L. Carter and M. N. Wegman, *Universal classes of hash functions*, J. Computer and System Sci., Vol. 18, pp. 143-154, 1979.
8. P. D'Arco, *An Almost-Optimal Forward-Private RFID Mutual Authentication Protocol with Tag Control*, available at http://www.dia.unisa.it/∼paodar.
9. P. D'Arco, A. Scafuro and I. Visconti, *Revisiting DoS Attacks and Privacy in RFID-Enabled Networks*, 5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS '09), Lecture Notes in Computer Science, Vol.5304, pp. 76–87, 2009.
10. I. Damgård and M.Østergaard, *RFID Security: Tradeoffs between Security and Efficiency*, Proceedings of the RSA Conference, Cryptographers' Track, Vol. , pp. 318 – 332, 2008.
11. R. Deng, Y. Li, A. Yao, M. Yung and Y. Zhao, *A New Framework for RFID Privacy*, eprint archive, report no. 2010/059, available at http://eprint.iacr.org/
12. R. Deng, Y. Li, M. Yung and Y. Zhao, *A New Framework for RFID Privacy*, Proceedings of Esorics 2010, Lecture Notes in Computer Science, 2010, Vol. 6345, pp. 1-18, 2010.
13. A. Juels, *The Vision of Secure RFID*, Proceedings of the IEEE, Vol. 95, No. 8, pp. 1507-1508, August 2007.
14. A. Juels, R. Pappu, and S. Garfinkel, *RFID Privacy: An Overview of Problems and Proposed Solutions*, IEEE Security and Privacy, Vol. 3, No. 3, pp. 34-43, May-June 2005.
15. M. Ohkubo, K. Suzuki, and S. Kinoshita, *Efficient hash-chain based RFID privacy protection scheme*, Proc. of the International Conference on Ubiquitous Computing Ubicomp Workshop Privacy: Current Status and Future Directions, Nottingham, England (September 2004).
16. R. Paise and S. Vaudenay, *Mutual Authentication in RFID: Security and Privacy*, Proceedings of Aisaccs08, Lecture Notes in Computer Science, Vol. , pp. 292–299, 2008.
17. T. Van Le, M. Burmester, and B. de Medeiros, *Universally Composable and Forward-Secure Rfid Authentication and Authenticated Key Exchange*, Proc. of ASIACCS'07, pp. 242-252, 2007.
18. S. Vaudenay, *On Privacy Models for RFID*, Proceedings of Asiacrypt 2007, Lecture Notes in Computer Science, Vol. , pp. 68–87, 2007.
19. D. R. Stinson, *Universal hashing and authentication codes*, Designs, Codes and Cryptography, No. 4, 369-380, 1994.
20. M. N. Wegman, *New hash functions and their use in authentication and set equality*, Journal of Computer and System Sciences, Vol. 22, No. 3, pp. 265-279, 1981.