

Time Measurement Threatens Privacy-Friendly RFID Authentication Protocols

Gildas Avoine*, Iwen Coisel** and Tania Martin

Université catholique de Louvain
B-1348 Louvain-La-Neuve, Belgium
{gildas.avoine, iwen.coisel, tania.martin}@uclouvain.be

Abstract. Privacy is one of the most important security concerns in radio frequency identification. The publication of hundred RFID-based authentication protocols during the last decade raised the need of designing a dedicated privacy model. An important step has been done with the model of Vaudenay that combines early models into a unified and powerful one. In particular, this model addresses the case where an adversary is able to know whether or not the protocol execution succeeded. This modelizes the fact that the adversary may get information from a side channel about the termination of the protocol, e.g., she notices that the access is granted to the RFID-tag holder. We go one step forward in this paper and stress that the adversary may also have access to a side channel that leaks the computational time of the reader. This modelizes an adversary who measures how long it takes to grant the access. Although this channel could be seen as an implementation flaw, we consider that it is always risky to require the implementation to solve what the design should deal with. This new channel enables to demonstrate that many key-reference protocols are not as privacy-friendly as they claim to be, e.g., WSRE, OSK, C^2 , O-FRAP, O-FRAKE, . . . We then introduce the TIMEFUL oracle in the model of Vaudenay, which allows to analyze the resistance of the protocols to time-based attacks as soon as the design phase. Finally, we suggest some methods that make RFID-based authentication protocols immune to such attacks.

Keywords. RFID, Authentication, Privacy, Time-Attack

1 Introduction

Radio Frequency IDentification (RFID) is a contactless technology used to identify and/or authenticate remote objects or persons, through a radio frequency channel using RFID *readers* and RFID *tags* (or transponders), the latter embedded into the items. RFID is becoming more and more widespread in daily-life applications, from library management [17] or pet identification [14], to anti

* This work was partially funded by the Walloon Region Marshall plan through the SPW DG06 Project TRASILUX

** This work was partially funded by the Walloon Region Marshall plan through the 816922 Project SEE

counterfeiting [26], access control [23] or even biometric passports [22]. The wide and fast deployment of RFID is mainly due to the diminution of the RFID tags price while their capacities steadily increase.

Moreover, the ubiquity of RFID raises new concerns about privacy. For instance in public transportation, a customer holding an RFID ticket might not want anybody else to be able to track his movings. One option to preempt such a worry is to build secure RFID authentication protocols, in order to ensure privacy for RFID users. Thus, an RFID system should provide *anonymity* (the identity of the tag should be kept secret) and *untraceability* (it should not be possible to link two different tag communications) for a user. Thereby, the design of secure and privacy-preserving RFID protocols requires an attentive and methodical analysis of its characteristics. Such an analysis is carried out with theoretical studies based on privacy models. In 2005, Avoine was the first to present such a framework in [2]. Since then, many attempts [11, 15, 16, 24] have been done to propose a convenient and appropriate privacy model for RFID. But each one suffers from distinct shortcomings: generally, these models do not take into account some important adversary features, such that the information given by the result of a tag authentication (does the tag has been authenticated successfully or not?), or the behavior of a “corrupted” tag (can it still be used in the system?). Given all these proposals for privacy analysis, Vaudenay’s model [25], presented at ASIACRYPT 2007, is known to be one of the most complete. Vaudenay defined eight privacy levels composed of:

- four notions related to the power of an adversary to compromise a tag (WEAK, FORWARD, DESTRUCTIVE and STRONG),
- and the notion related to the possible access by the adversary to the side-channel information given by the result of a tag authentication (NARROW),

and proved some feasible and some impossible privacy results for several well-known authentication protocols. The NARROW notion was introduced by Juels and Weis [15]. They clearly explain that an adversary may have a single bit information from the reader, i.e. whether or not the tag authentication succeeds. An example is an access card opening a building door. Vaudenay was the first to formalize this notion in his model. In the same way, we decide to formalize the time spent by the reader to open this door as an other side information. This crucial privacy notion related to *time* is missing in Vaudenay’s model. To be precise, the *time* that a reader will take to authenticate a tag can also be a hint to recognize a tag from another. Actually, the reader database contains the keys of all the tags involved in the system. These keys are used by the readers to authenticate successfully every tag. Thus, this *time* notion is essentially related to the key infrastructure used by the readers database to organize and retrieve all this information.

The *time* notion in key infrastructures. An RFID scheme can use different key infrastructures. As RFID lives in a constrained environment, it seems that secret-key infrastructures are well-suited for this kind of applications. However,

as some researches improve public-key techniques (*e.g.* efficient implementation of WIPR [20, 21, 28], or of GPS [13]), then an outstanding public-key infrastructure can be really interesting in this domain. It is possible to use a single key, shared between all the readers and tags of the system, or each tag can also hold its own secret key. In the latter situation, every reader has to know all the tags keys and carry on a search procedure during the protocol execution to guess which key it has to use to authenticate a particular tag.

It is quite obvious that the single secret-key infrastructure is not relevant. In fact, if we consider that RFID tags are not tamper-resistant, an adversary can compromise the whole system if she is able to corrupt a single tag.

This is not the case with a single public-key infrastructure as only the reader knows the private secret key and this device is not vulnerable against jeopardy. Some schemes already exist using a single public-key which is generally defined for a public-key encryption scheme (*e.g.* [10, 20, 25]). Although the time needed to authenticate a tag seems to be constant, and thus avoiding time-attacks, the requirements for the encryption scheme (*e.g.* IND-CCA2 in Vaudenay’s model [25], or IND-CPA plus a MAC scheme in [10]) in order to ensure privacy imply a highest computation complexity from the tag’s side. Consequently, these solutions seem still too expensive for RFID context.

In particular infrastructure cases (public or secret), in order to authenticate a tag, the reader has to retrieve the corresponding key(s). As the schemes we study in this paper manage to ensure, at least, the untraceability property, tags should not send a fixed information which allows the reader to immediately retrieve this key. Indeed, a fixed value allows an adversary to trivially link up several authentications of a same tag. As a consequence, this information should be given in an hidden way and the reader will perform a SEARCHID procedure to retrieve the corresponding key. An obvious method to do this is to perform an exhaustive search in the whole reader database. Following this example of exhaustive search, the reader might always scan its database in the same way. So clearly, for a given tag, the reader will always authenticate it at the same moment of its search. Thus the time that the reader will spend to authenticate this tag will be the same at every protocol execution. Consequently, an adversary can deduce which tag has been authenticated by the reader by only observing this time. Clearly, she can have access to this *time* information. In practice, this data is not given by the RFID system itself: the adversary will compute this time herself. Obviously, this is an important issue for tag privacy preservation. But this notion has not been yet included in any existing privacy model for RFID.

Our contributions. In this paper, we first modify Vaudenay’s model in order to add this *time* notion into this model. To do so, we formalize this new privacy level. We call it TIMEFUL. This notion can therefore be combined with all the Vaudenay’s levels. Then, we display the weaknesses of several existing protocols according to this *time* notion: we demonstrate that that OSK [19] is not TIMEFUL-NARROW-WEAK-private, and that neither WSRE [27], nor all the “undesynchronizable” protocols such as C^2 [9], O-FRAP or O-FRAKE [16] are

TIMEFUL-WEAK-private. Finally, we propose various solutions to provide TIMEFUL privacy. They consist in combining an appropriate choice for the reader database structure with a pertinent search procedure: our approaches are based on rainbow tables, hash tables, B-trees, and random search.

Structure of the paper. In Section 2, our modification of Vaudenay’s model is detailed. The privacy analysis of some well-known existing protocols is done in Section 3. Section 4 introduces several solutions and improvements that can be conducted on the search procedure of the protocols to provide tag privacy. We conclude the paper in Section 5.

2 The Modified Vaudenay Privacy Model

In this section, we present a modification of the well-known Vaudenay privacy model [25] for authentication/identification schemes in RFID systems. Then we describe all the possible interactions of an adversary with this system.

A *tag* \mathcal{T} is identified by a unique identifier ID , with limited memory and computational abilities, that can communicate with a *reader* \mathcal{R} up to a limited distance. A reader is composed of (i) a transreceiver which communicates with possibly several tags and (ii) a back-end database containing all identifiers ID of valid tags and additional data such as secret keys. We assume that communications between the transreceiver and the database are secure. In terms of security, one main difference between a tag and a reader is that a tag cannot be considered as a tamper-resistant device (and thus an RFID tag can be corrupted by an adversary against the system). We also assume that the reader is more powerful than a tag.

2.1 Definition of the Procedures

A privacy-preserving RFID authentication scheme, denoted \mathcal{S} , is composed of the following procedures, where λ is a security parameter.

- $\text{SETUPREADER}(1^\lambda)$ is a scheme which generates a private/public key pair (K_S, K_P) for the reader \mathcal{R} , depending on the security parameter λ . It also creates an empty database $DB_{\mathcal{R}}$ which will later contain the identifiers and keys of all tags.
- $\text{SETUPTAG}(ID, K_P)$ is a probabilistic algorithm which returns a tag-dependent key set $\text{tk}[ID]$. $(ID, \text{tk}[ID])$ is added in the reader’s database $DB_{\mathcal{R}}$ when the tag is legitimate.
- IDENT is an interactive protocol between the reader \mathcal{R} taking on inputs $1^\lambda, K_S, K_P$ and $DB_{\mathcal{R}}$, and a tag \mathcal{T} with identifier ID taking on inputs $1^\lambda, \text{tk}[ID], K_P$ and possibly ID . At the end of the protocol, the reader either accepts the tag (if \mathcal{T} is legitimate) and outputs its identifier ID , or rejects it (if not) and outputs \perp .

2.2 Definition of the Oracles

We now define the adversary \mathcal{A} against such RFID systems. We consider that there is only one valid reader \mathcal{R} in the system. However as we will see below, the adversary may play the role of dishonest readers to interact with a tag. In such a case, we assume that the tag does not know *a priori* if it is interacting with the valid reader \mathcal{R} or the adversary \mathcal{A} . Then, the main features of an adversary are basically given by:

- the actions she is allowed to perform, which are represented by the *oracles* she can query,
- the goal of her attack and the way she will perform it, depicted by an *experiment* (or *game*) containing the rules she has to follow.

At the beginning of each experiment, we assume that the SETUPREADER procedure has already been executed by a challenger denoted \mathcal{C} and thus that the values 1^λ , K_S and K_P already exist. We next assume that 1^λ and K_P are always given to \mathcal{A} , whereas K_S never (since the valid reader cannot be corrupted). At the beginning of one experiment, we consider that there is no tag in the system. We thus give to \mathcal{A} the oracle $\mathcal{O}^{\text{CREATE_TAG}}(\text{ID}, b)$ to introduce new ones. As in the Vaudenay model [25], we consider that the adversary can only interact with tags that are sufficiently close to her without having access to other existing ones. We thus use Vaudenay’s concept of *free* and *drawn* tags. Drawn tags are the ones within “visual contact” to the adversary so that she can communicate while being able to link communications. Free tags are the other tags with which the adversary cannot interact. At the creation of a new tag, that is after the call to $\mathcal{O}^{\text{CREATE_TAG}}(\text{ID}, b)$, the new tag has the status *free* and the adversary cannot interact with it.

- $\mathcal{O}^{\text{CREATE_TAG}}(\text{ID}, b)$: this oracle creates a *free* tag with a unique identifier ID , either legitimate ($b = 1$) or not ($b = 0$). This oracle uses the SETUPTAG algorithm with ID on input to set up the tag with $\text{tk}[\text{ID}]$. For $b = 1$ only, this oracle updates $\text{DB}_{\mathcal{R}}$, adding this new tag. By convention, b is implicitly 1 when omitted.

Next, the adversary can modify the status of the created tag by using the following oracles.

- $\mathcal{O}^{\text{DRAW_TAG}}(\text{distr}, k) \rightarrow (\mathbf{t}_1, b_1, \dots, \mathbf{t}_k, b_k)$: with distribution probability distr , this oracle randomly selects k tags between all the existing (not already drawn) ones. For each chosen tag, the oracle gives it a new pseudonym denoted \mathbf{t}_i and changes its status from *free* to *drawn*. Finally, the oracle outputs all the generated pseudonyms $(\mathbf{t}_1, \dots, \mathbf{t}_k)$ in any order. If there is not enough free tags (*i.e.* less than k), then the oracle outputs \perp . We further assume that this oracle returns bits (b_1, \dots, b_k) telling whether the drawn tags are legitimate or not. All relations $(\mathbf{t}_i, \text{ID})$ are kept in an *a priori* secret table denoted Tab .

- $\mathcal{O}^{\text{FREE}}(\mathbf{t})$: this oracle moves the tag with pseudonym \mathbf{t} from the status *drawn* to the status *free*. This makes \mathbf{t} unavailable from now on (in particular, \mathcal{A} cannot interact with the tag \mathbf{t} anymore).

Then, the adversary is only able to interact with tags by using the pseudonyms and only if the tag has the status *drawn*. To simplify notation, we denote by $\text{tk}[\mathbf{t}]$ the secret key of the tag with pseudonym \mathbf{t} , which is equal to the secret key $\text{tk}[\text{ID}]$ of the underlying identifier ID of this tag. Using a pseudonym, the adversary has now several ways to interact with tags.

First, \mathcal{A} is able to corrupt drawn tags by using the following oracle.

- $\mathcal{O}^{\text{CORRUPT}}(\mathbf{t}) \rightarrow \text{tk}[\mathbf{t}]$: returns the tag-dependent key $\text{tk}[\mathbf{t}]$. The pseudonym \mathbf{t} is now marked as “corrupted”¹. If \mathbf{t} is no longer used by \mathcal{A} after this oracle call, we say that \mathbf{t} is considered as “destroyed”.

Next, the adversary can *passively* witness the whole protocol IDENT between a tag and the valid reader \mathcal{R} by using the following oracle.

- $\mathcal{O}^{\text{EXECUTE}}(\mathbf{t}) \rightarrow (\pi, \text{transcript})$: executes an IDENT protocol between the reader and the tag with pseudonym \mathbf{t} . This oracle outputs the transcript of the protocol instance π , that is the whole list of the successive messages of the protocol.

\mathcal{A} can also *actively* participate in the IDENT protocol by playing the role of either a fake/corrupted tag, or an invalid reader. For this purpose, the following oracles are introduced, and they also allow \mathcal{A} to stop at any step of a “standard” authentication protocol, delete or modify some messages.

- $\mathcal{O}^{\text{LAUNCH}}() \rightarrow \pi$: makes the legitimate reader \mathcal{R} launch a new IDENT protocol instance, that is the first request to an unknown tag so as to authenticate and identify it. It outputs the identifier π for this protocol instance.
- $\mathcal{O}^{\text{SENDREADER}}(m, \pi) \rightarrow r$: sends a message m to the reader \mathcal{R} in the protocol instance π . It outputs the response r from the reader.
- $\mathcal{O}^{\text{SENDTAG}}(m, \mathbf{t}) \rightarrow r$: sends a message m to the tag with pseudonym \mathbf{t} . It outputs the response r from the tag.
- $\mathcal{O}^{\text{RETURN}}(\pi) \rightarrow x$: when π is completed, it outputs $x = 0$ if the output of the reader during the IDENT protocol instance π is \perp , and $x = 1$ otherwise.

Finally, the adversary is allowed to ask the time spent by the reader to compute all the operations and to perform its SEARCHID procedure, in order to authenticate the tag linked to a particular protocol instance.

- $\mathcal{O}^{\text{TIMER}}(\pi) \rightarrow \delta$: it outputs the time δ taken by the reader for its overall computations during the protocol instance π .

¹ Note that the underlying tag with identifier ID is also corrupted. However, a new pseudonym of this tag can also be corrupted. Thus, a pseudonym \mathbf{t} can only be corrupted once while a tag ID may be corrupted several times.

2.3 The Security of an Authentication Scheme

We remind here the notions of *completeness*, *availability* and *soundness* that are the basis of the well functioning and the security level of an authentication protocol. First, it is necessary to prove that a valid tag is always authenticated successfully by a valid reader.

Definition 1 (Completeness). *For every legitimate tag \mathcal{T} of an RFID system, the probability that the reader \mathcal{R} returns the tag identifier ID at the end of the IDENT protocol is overwhelming.*

Because an adversary \mathcal{A} is able to interact with a tag for an attack, it is also important to prove that a valid tag \mathcal{T} is still authenticated successfully by a valid reader, even after that an adversary conducted an attack on \mathcal{T} .

Definition 2 (Availability, Strong Completeness). *For every legitimate tag \mathcal{T} of an RFID system that could have been subjected to an attack, the probability that the reader \mathcal{R} returns the tag identifier ID at the end of the IDENT protocol is overwhelming.*

The previous notions ensure the authentication success of a legitimate tag \mathcal{T} . But the security of a scheme is also based on the fact that an adversary must not be able to impersonate \mathcal{T} . To prove it, \mathcal{A} can use every oracle, except $\mathcal{O}^{\text{CORRUPT}}$ since this oracle makes the impersonation trivial.

Definition 3 (Soundness). *A scheme is said sound if the probability that an adversary impersonates a legitimate tag is negligible.*

2.4 Definition of the Adversary

We now define the different classes of adversaries who will play security experiments. We here give the classification given by Vaudenay in [25] with our modification which introduces the notion of *time*.

Definition 4 (Adversary Class). *An adversary \mathcal{A} against the RFID system who has no access to the $\mathcal{O}^{\text{TIMER}}$ oracle is said to be:*

- **STRONG** if \mathcal{A} has no limit on all the others oracles;
- **DESTRUCTIVE** if \mathcal{A} cannot use anymore a “corrupted” tag (i.e. the tag has been destroyed);
- **FORWARD** if \mathcal{A} is committed to only use the $\mathcal{O}^{\text{CORRUPT}}$ oracle after her first call to the $\mathcal{O}^{\text{CORRUPT}}$ oracle;
- **WEAK** if \mathcal{A} has no access to the $\mathcal{O}^{\text{CORRUPT}}$ oracle.

\mathcal{A} is said **NARROW** if she has no access to $\mathcal{O}^{\text{RETURN}}$.

\mathcal{A} is moreover said **TIMEFUL** if she has access to the $\mathcal{O}^{\text{TIMER}}$ oracle.

According to the Vaudenay privacy model, a “blinded” adversary is defined as an entity which interacts with a simulated system, controlled by a simulator who does not know anything about secret values. Then, a scheme ensures the

privacy property if, for a given experiment (see below), the success probability of an adversary which interacts with the system through oracles (as defined in section 2.2) is undistinguishable of a “blinded” adversary.

More formally, we define the following experiment with \mathcal{A}_P being the adversary with power $P \in \{\emptyset, \text{TIMEFUL}\} \cup \{\emptyset, \text{NARROW}\} \cup \{\text{WEAK}, \text{FORWARD}, \text{DESTRUCTIVE}, \text{STRONG}\}$:

Experiment $Exp_{\mathcal{S}, \mathcal{A}_P}^{\text{Vaud-priv}}$

1. The challenger \mathcal{C} initializes the system and sends 1^λ , and K_P to \mathcal{A}_P .
2. \mathcal{A}_P interacts with the whole system, limited by her class P .
3. \mathcal{A}_P submits an hypothesis about the system and receives the hidden table Tab of the $\mathcal{O}^{\text{DRAWTAG}}$ oracle.
4. \mathcal{A}_P returns 1 if her hypothesis is correct and 0 otherwise.

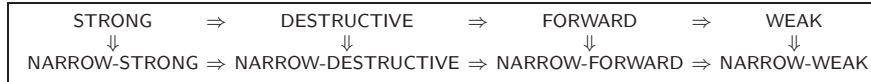
The adversary wins if she returns 1.

Definition 5 (Trivial Adversary). *An adversary \mathcal{A} is said trivial if it is possible to define a simulator Sim who perfectly simulates the system, without knowing any secrets, for a “blinded” adversary denoted \mathcal{A}^{Sim} , such that $|\text{Pr}[\mathcal{A} \text{ wins}] - \text{Pr}[\mathcal{A}^{\text{Sim}} \text{ wins}]|$ is negligible.*

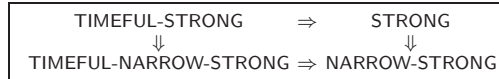
If those success probabilities are undistinguishable, it means that there is no privacy loss through the communication channel. In other words, the adversary makes no effective use of the messages as their simulation (without using the secret values) leads to the same probability of success. Thus the RFID authentication scheme \mathcal{S} can be considered private.

Definition 6 (Privacy). *A scheme is said P -private if all the adversaries who belong to class P are trivial.*

For our modification of the Vaudenay model, it is essential to understand that the TIMEFUL adversary class formalizes the notion of *time* we want to introduce. Concretely, if \mathcal{A} has access to the $\mathcal{O}^{\text{TIMER}}$ oracle, she knows the time that the reader has taken to authenticate a tag. With this information, if \mathcal{A} cannot deduce anything about the tag identity, we will say that the protocol is TIMEFUL-private. We remind the following implications between Vaudenay privacy properties which are obvious.



With the introduction of the TIMEFUL adversary class, we now have new connections at each level (STRONG, DESTRUCTIVE, FORWARD and WEAK) of the previous diagram. For better clearness and understanding, we only give the new links for the STRONG level:



3 Existing Protocols

In this section, we analyze the authentication time of some existing schemes. We only get interested in protocols based on symmetric-key cryptography, since they are more suitable for lightweight RFID. In Vaudenay’s model, all of the following schemes ensure at least the NARROW-WEAK privacy. Using our TIMEFUL adversary, we will prove that none of them reaches the TIMEFUL-WEAK privacy. Note that in the following, the only time difference appears in the SEARCHID procedure. We thus only focus on this part for our study. This is not always the case in practice, consequently, we define a more general model where the whole protocol execution is taken into account. For an example, the reader can look at the protocol O-FRAPv2 introduced by Burmester, de Medeiros and Motta in [6, 7] and described in Appendix A.

3.1 A Trivial Example: WSRE

We first study the scheme introduced by Weis, Sarma, Rivest and Engels in [27]. This scheme is a simple challenge/response protocol. The reader sends an authentication request and the tag ID answers by $(f(\text{tk}[\text{ID}]||N_T), N_T)$, where N_T is a nonce and f a one-way function. To authenticate the tag, the reader performs a SEARCHID procedure where it computes for each possible key stored in the database the output of f using the received nonce. When there is a match, the reader outputs the associated identifier.

This scheme does not ensure the FORWARD privacy. Assume that an adversary has stored some transcripts of past authentications. Then when she corrupts a tag, she obtains its key, thereby she can recompute the output of f using this key and the nonce of a transcript to compare it with the sent output. If there is a match, that means the adversary has linked the identity of the tag with a previous authentication transcript.

In fact, we can also prove that a TIMEFUL adversary can trace a tag without corrupting it. Note that for the rest of the paper, we assume that the f function, but also hash functions, have the same execution time, whatever the input is.

Theorem 1. *The WSRE protocol does not ensure the TIMEFUL-WEAK privacy.*

Proof. To prove this result, we have to exhibit an adversary which has a success probability different than the one of whichever blinded adversary. This adversary, denoted \mathcal{A} , can be described as follows.

- \mathcal{A} creates two legitimate tags using twice $\mathcal{O}^{\text{CREATE TAG}}(\text{ID}, b)$ and affects them by a call to $\mathcal{O}^{\text{DRAW TAG}}(1/2, 2)$. \mathcal{A} receives two pseudonyms t_1 and t_2 .
- \mathcal{A} calls $\mathcal{O}^{\text{EXECUTE}}(t_1)$ and $\mathcal{O}^{\text{EXECUTE}}(t_2)$. She receives $(\pi_1, \text{transcript}_1)$ and $(\pi_2, \text{transcript}_2)$. Then she asks the time for each of these authentication, thus she requests $\mathcal{O}^{\text{TIMER}}(\pi_1)$ and $\mathcal{O}^{\text{TIMER}}(\pi_2)$ to obtain δ_1 and δ_2 .
- \mathcal{A} frees both tags with the requests $\mathcal{O}^{\text{FREE}}(t_1)$ and $\mathcal{O}^{\text{FREE}}(t_2)$, and re-affects only one of them with $\mathcal{O}^{\text{DRAW TAG}}(1/2, 1)$. She obtains a new pseudonym t_3 .

- Again \mathcal{A} executes an instance protocol by requesting $\mathcal{O}^{\text{EXECUTE}}(\mathbf{t}_3)$ and asks for the time δ_3 of this authentication instance with $\mathcal{O}^{\text{TIMER}}(\pi_3)$.
- If $\delta_3 = \delta_1$, \mathcal{A} claims that $\mathbf{t}_1 = \mathbf{t}_3$, else she claims that $\mathbf{t}_2 = \mathbf{t}_3$.

It is obvious that the success probability of this adversary is 1. Now we have to prove that any blinded adversary can have this probability. The simulator does not have any clue on which of the two tags has been drawn during the second call to the $\mathcal{O}^{\text{DRAWTAG}}$ oracle, but he can simulate perfectly the answer of the tags (as f is assumed to be pseudo-random). On the other hand, he has to make a choice between the two times δ_1 and δ_2 . His only solution is to perform a random choice. In this way, the simulation stays perfect. But, the blinded adversary will only have a success probability of $1/2$ as her success is based on the correctness of the simulator's choice.

Consequently, the protocol is not TIMEFUL-WEAK private.

3.2 OSK Scheme

In order to ensure the FORWARD-privacy, Ohkubo, Suzuki, and Kinoshita have introduced the well-known OSK scheme in [19]. In the latter, they use a key-update mechanism in order to modify the internal state of a tag after each protocol execution (successful or not) in a one-way manner. The OSK scheme is presented in Figure 1, where H_1 and H_2 are cryptographic secure hash functions.

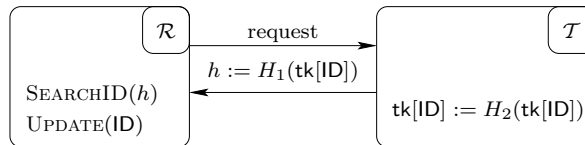


Fig. 1. OSK protocol

By updating the secret key of a tag after each authentication protocol, an adversary will not be able to recompute a previous answer of a tag after she had corrupted it (as the update is one-way). In the SEARCHID procedure, the reader computes for each tag the hash (using H_1) of its key. At any moment, if there is a match with the received value, the reader stops the procedure and outputs the corresponding identifier and updates the key, using H_2 . After testing every key, if the reader did not find a match, it tries again with the updated key (computed on the fly), and so on.

Even so, it has been shown in [9, 15] that the protocol is weakened by desynchronization attacks. Thus, this protocol does not ensure the WEAK privacy as it does not ensure the availability property. But it provides NARROW-WEAK privacy. See the original publications [9, 15] for more details on this proof.

On the other hand, the time attack presented in the previous subsection is possible. Moreover, for any pair of tags, the difference of authentication times

can be increased. Indeed, as the adversary can desynchronize a tag, she increases the authentication time of a tag at every desynchronization. Consequently, she can distinguish one tag among two easily, proving that the OSK scheme is not TIMEFUL-NARROW-WEAK private.

Remark 1. In the original article [19], the SEARCHID procedure was not described as presented here. In fact, the reader first performs all computations before comparing all these values with the one received. Thus, our time attack does not work with this procedure when tags are synchronized. Nevertheless, when a tag is desynchronized the authentication time will still be longer as the one of a synchronized tag. We present this SEARCHID procedure instead of the original one as it is generally how it is presented in many contributions.

3.3 Undesynchronizable Schemes

Some attempts have been done in order to define undesynchronizable schemes using key-update mechanisms. The objective of these schemes is to ensure the availability property and to reach the FORWARD privacy. To do so, mutual authentication schemes have been defined, where the tags update their key *if and only if* they authenticate the reader. Based on this fact, these schemes ensure that the key inside the tag and the one inside the reader database can be desynchronized at most one time. Thus to ensure the availability, it is sufficient to store in the database two keys per tag. As an example, we describe the C^2 scheme in Figure 2, introduced in [9].

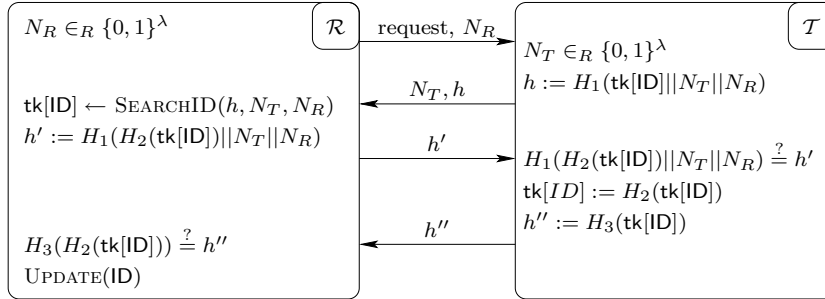


Fig. 2. C^2 protocol

As said previously, the database contains two keys per tag: the current one and the “next” one. As a consequence, the scheme is no longer vulnerable to traceability using desynchronization attacks. This technique was also present in the Dimitriou scheme [12], even if the WEAK privacy is not ensured by this scheme as proved in [9]. However, this scheme still does not reach the FORWARD privacy as the following attack is doable: if the tag is corrupted by an adversary just after the latter has blocked the last authentication of this tag after it sent

N_T, h , then she is able to recompute h and thus to trace it. Consequently, this scheme only ensure the WEAK privacy property.

Although these schemes ensure the availability, an adversary is still capable to perform the previous time attack as the reader perform an exhaustive search. This implies at least one computation per tag. In the C^2 scheme, a tag can still be desynchronized one time which allows the adversary to distinguish any tag among two, as in the OSK scheme. As a result, C^2 is not TIMEFUL-WEAK private.

3.4 Overview

We have presented several schemes where a TIMEFUL adversary is able to break the privacy while some of them were assumed to ensure at least the WEAK privacy property. The different protocols that we have presented do not define an exhaustive list of those weakened by this new attack. For instance, the protocols O-FRAP and O-FRAKE, introduced by Le, Burmester and de Medeiros in [16], are also weakened by this attack as the SEARCHID procedure also implies a linear exhaustive search. This attack also works when the underlying architecture of the database is different. Namely, the tree-based protocol introduced by Molnar and Wagner in [17] suffers of a time-flaw when the SEARCHID procedure is defined as an exhaustive search at each tree level (see [17] for more details).

Moreover, one can think that this attack only affects protocols using symmetric cryptography architecture as we only present this kind of protocol. We have made this choice since most of them are affected by this attack. Nevertheless, some authentication schemes using public-key technique can be attacked by a TIMEFUL adversary. For example, this is the case of the protocol introduced by Bringer, Chabanne and Icart in [5].

4 Solutions and Improvements

Our attacks presented in the previous section always work in theory, but they require in practice a tight time measurement. For instance, we saw previously that WSRE involves a SEARCHID procedure done with an exhaustive search. Thus, if we consider two tags which data are very close in the reader database (*e.g.* one following the other), the time of the SEARCHID procedure will be almost the same for both: the time difference between the computation of one function f or two can be very small. Then in such a case, if the time measurement of the adversary is not precise enough, she will not be able to differentiate one of these two tags from the other. In our study, we consider that this adversarial issue mainly depends on the implementation of the function f , and thus that it is a programmer concern.

Here we want to propose theoretical solutions to this time problem that can be applied to these protocols. The most obvious one is to compute the worst case's time and that whatever happens, the reader waits² until it reaches this

² During this time, it can also compute unused hash functions if the adversary has a look-up of its power consumption.

time to output the result. This has been mentioned by Burmester, Le, and de Medeiros in [8]. This solution clearly repairs all the previous protocols against our time attack. However, the protocol efficiency can be highly decreased depending of the protocol. Our goal is to optimize the average authentication time while the protocol resists against a TIMEFUL adversary.

In our first study, we focus on protocols where the reader cannot predict the tag outputs (*i.e.* the tag inserts a nonce in its answer). Then we propose three solutions to improve OSK. Finally, we present a protocol which is claimed to be based on constant time.

4.1 The Random Search

In the case of C^2 or O-FRAP/O-FRAKE, the only solution for a reader to authenticate a tag is to compute for each key stored in its database the theoretical output of the corresponding tag, and to compare it with the received value. We have shown in the previous section (see Section 3.3) that, if this SEARCHID procedure is done linearly, then a TIMEFUL adversary can trace a tag with a non-negligible probability.

A simple solution is to randomize this search. The objective is to avoid the adversary to predict the time spent for a given tag authentication. To do so, the easiest way is to modify each time the “start-point” of the linear search in the database. If it is chosen uniformly in $[1, n]$, where n is the total number of tags, then the authentication time of a tag cannot be guessed in advance.

However, we have to take into account that a tag can be desynchronized once. In our solution, the reader first computes all the theoretical outputs using the current key. Then it only computes these outputs with the old key if it did not find a match before. Thus a “synchronized tag” will be authenticated in $\delta_h \cdot n/2$ time in average, whereas a desynchronized tag is authenticated in $3\delta_h \cdot n/2$ time in average, where δ_h denotes the execution time of the h function. So, by desynchronizing a tag, a TIMEFUL adversary is still able to trace a tag. Consequently, to ensure the TIMEFUL-WEAK privacy, the randomized SEARCHID procedure should indifferently test the current or the old key of tags. Thus, this procedure should consider a set of $2n$ keys and randomly tests one key after the other without considering if it is a current key or a future one.

Using this SEARCHID procedure, it is obvious that these kinds of protocols reach the TIMEFUL-WEAK privacy property. Unfortunately, the price to pay is that the system’s efficiency is decreased. Indeed, the average time to authenticate a tag under a normal behavior (when the adversary does not desynchronize a tag) is $\delta_h \cdot n$ instead of $\delta_h \cdot n/2$. Yet we still improve the “wait” solution as this one requires a time of $2\delta_h \cdot n$.

Remark 2. Considering the WSRE protocol (presented section 3.1) which does not use a key-update mechanism, this solution does not modify the average authentication time while it ensures the TIMEFUL-WEAK privacy property.

4.2 The OSK Scheme

Another situation is when it is possible to precompute the whole set of possible answers (or a part of it). Then it is possible to use some time/memory trade-off to enhance the complexity of the SEARCHID procedure and, in some cases, to obtain a constant look-up in the database. For example, in the OSK scheme (see Figure. 1) the tag answer does not include any nonce and thus the database can store all the tags answers. As said previously, this scheme is extremely desynchronizable. Consequently, instead of storing one answer per tag, the database should contain m successive answers for each tag. This highly increases the size of the database ($O(n.m)$ instead of $O(n)$), but depending of the database infrastructure, the SEARCHID procedure can be really efficient. We here present three infrastructure possibilities which ensure the TIMEFUL-NARROW-WEAK privacy property.

First, we shortly present an optimization called OSK-AO introduced by Avoine, Dysli and Oechslin in [3,4] based on the well-known rainbow tables, introduced by Oechslin in [18]. In a nutshell, all of the $n.m$ possible answers (*i.e.* the hash values of the m successive keys for each of the n tags) are distributed uniformly in a table (the table's size defines the time-memory trade-off). Each row of the database contains a succession of hash values. One value is obtained from the previous one by applying an arbitrary reduction function composed with the hash function. This reduction function takes in input a hash value and outputs an identifier in $[1, n]$ and the "update value" in $[1, m]$. Using these two values, the next hash value can be computed using the hash function. The database has to store the first and the final column of this table. Upon receiving a hash value, the reader will compose a chain of values (as done in the construction of the table) until a match is found with the last column of the table. When it happens, the reader reconstructs the corresponding row until it found the previous value, and thus obtains the identifier and the "update value". If the reduction function maps all the possible values in a uniformly manner in the database, an adversary will not be able to predict the authentication time for a given tag, and thus to trace it. Moreover, contrary to the next following solutions, this solution does not store the $n.m$ answers of tags. However, this structure is not dynamic and cannot be modified. Consequently, to introduce tags updates, the whole table must be recomputed.

Another solution is to compose the database as a hash table, where the entries are indexed by the hash values. In this kind of database, the SEARCHID procedure is quite instantaneous ($O(1)$ in average). However, to avoid collisions, the hash index used should be as long as the output of the hash function. This is quite impracticable when $n.m$ is large. Moreover, this solution is not adapted for dynamic system where the number of tags can increase during the life system. Indeed, as the number of inputs increases, so does the probability of a collision in the hash index. If this happens, then SEARCHID takes up to linear time (in $O(n)$).

Remark 3. Moreover, the database should keep in another table the current key (from the database point of view) of each tag. Indeed, if the tag is desynchro-

nized, the use of the hash table allows the reader to authenticate the tag and to obtain the used key, but it will not be able to recompute the previous ones (as the hash function is one-way). Consequently, to delete the previous entries of the database (to keep it as small as possible), the reader should use this new table to recompute all the previous theoretical answers to delete them.

Finally, another possibility is to use a balanced binary search tree (called B-tree). This technique ensures a complexity in $O(\log n)$ for the SEARCHID procedure. The advantage of this structure is its dynamism. Contrary to the hash table, new entries can be added indefinitely in this structure without compromising its functioning. Moreover, in the worst case, the B-tree ensures a better complexity rather than the hash table where the complexity reach is in $O(n)$ in the worst case (*i.e.* when collisions happen on the hash indexes).

These three practical solutions avoid time attack for the OSK protocol, and thus ensure the TIMEFUL-NARROW-WEAK privacy. But, except for the OSK-AO solution, the database's size is $O(n.m)$ which becomes quickly infeasible, especially if the availability is highly recommended (and thus m must be large enough).

4.3 Constant-Time Identification

Not necessarily to solve the problem of time attack, but rather to reduce the complexity of the SEARCHID procedure, some protocols with a constant-time identification have been proposed (*e.g.* [1, 6, 7]). In this section, we present in detail the protocol proposed by Alomair, Clark, Cuellar and Poovendran in [1]. We give in appendix a description of the protocol of Burmester, de Medeiros and Motta [6, 7] which unfortunately has some security flaws.

The protocol of Alomair, Clark, Cuellar and Poovendran is detailed in Figure 3. We give here a short description of it. For more details, see the original publication [1]. The important step in the tag authentication is the sent value $h(\Psi||m)$. Ψ represents a pseudonym associated to this tag and m is a counter value which is incremented after each (successful or not) tag authentication. In the database, all the possible hash values for all the possible pseudonyms and all the counter values are precomputed and stored. Based on a special infrastructure detailed later, the reader is able to retrieve quite instantaneously all the associated values to the corresponding pseudonym, *i.e.* the secret key of the tag and its identifier. The reader is then able to compute the last message of the protocol which is composed of three parts. The first one ($h(1||\Psi||\text{tk}[\text{ID}]||h_1)$) allows the tag to authenticate the reader. The second one ($h(2||\Psi||\text{tk}[\text{ID}]||h_1) \oplus \Psi'$) transmits to the tag securely a new pseudonym which has been selected among the available ones in the database. The last one ($h(3||\Psi'||\text{tk}[\text{ID}]||h_1)$) permits the tag to check the integrity of this new pseudonym.

As explained in [1], the database can be decomposed in three logical parts. The first one, denoted M-I, can be viewed as a hash table which allows to define a direct addressing to the hash values $h(\Psi||m)$. All these values are stored in the second part of the database, denoted M-II. Finally, each of these hash values

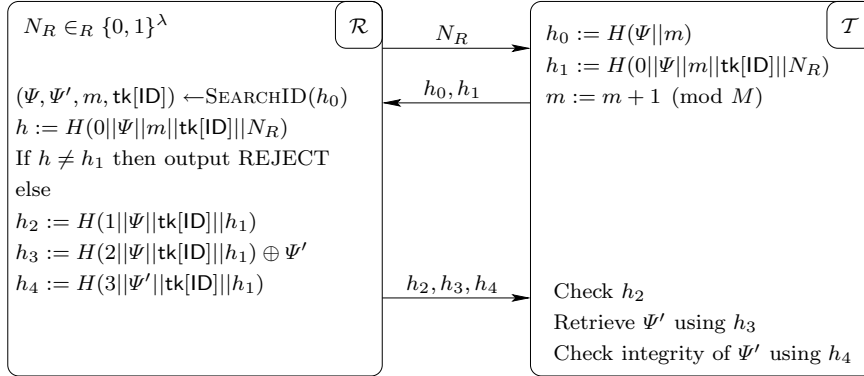


Fig. 3. Constant-Time Identification Protocol

points to one cell of the last part of the database, denoted M-III, which contains all the information related to the tag currently attached to the pseudonym Ψ .

Remark 4. There is a mistake in the description of this database in [1]. Indeed, in the cells of M-II, the authors said that this table only contains the hash value and a pointer to tag's data. However, this cell must contain the counter value m and the pseudonym which are used in the hash value. These values are essential to check (by recomputing it) the message $h(0 \parallel \Psi \parallel m \parallel \text{tk}[\text{ID}] \parallel N_R)$ sent by the tag.

In their case study, the authors of [1] only consider the size of M-I as they claim that it is the only concern for the total size. We disagree with this fact and prove it by computing the size of the M-II part. We obviously use the same parameters as those of [1]. Namely the total number of pseudonyms is $N = 2.10^9$ and the counter m is majored by $M = 10^3$. Thus, the part M-II is composed of 2.10^{12} cells. Each of them contains the hash value $h(\Psi, m)$, the counter m and a pointer to the table M-III. Note that for addressing the 2.10^9 cells of the part M-III (one for each pseudonym), a pointer of 32 bits is enough. In [1], the authors said that the hash function's output must be at least of $\lfloor \log_2 NM \rfloor \approx 41$ bits. Then, each cell of M-II contains at least 83 bits (the counter is approximatively 10 bits long). We thus obtain a total of at least³ 166.10^{12} bits, which is approximatively equal to 19 terabytes. This is obviously not negligible compared to the 12 terabytes of M-I.

Although this is still feasible in practice, it is not so practicable. Furthermore, they neglect another fact yet highlighted in the paper. As a tag can be desynchronized once, each tag should be associated to two pseudonyms. Thus the total number of pseudonyms should not be twice the total number of tags, but more than this, for example three times this number. This again increases the database size (in this example from 31 to 40 terabytes).

We thus propose a modification of this database. The hash values should not be stored in M-II which then only contains the counter m and the pointer. In

³ Recall that the hash function's output should be greater than 41 bits in practice.

average this will not be a problem because, as presented in [1], in most cases, pointers of M-I are attached to only one cell. As the reader must check the correctness of $h(0||\Psi||m||tk[ID]||N_R)$, it will be able to differentiate two tags which have the same address in M-I. We recognize that our optimization will increase the number of computations that the reader has to do to identify a tag. For example, if a hash value points to two different tags in M-II, the reader may compute two hash values to authenticate the tag. However this allows to decrease the size of M-II from 28 to 14 terabytes which is not negligible and the collision event happens with a small probability. Note that now, the authentication time is no longer constant. Nevertheless, an adversary is not able to predict if a tag, and more precisely a pseudonym, will collide in the M-II table with another pseudonym with a given counter. Thus, she is not able at all to trace a tag using this difference of time.

Despite this huge amount of data, this protocol is however really efficient in terms of time and gives a solution to our time attack by providing a constant-time authentication. As a conclusion, this scheme is nowadays the best solution in terms of efficiency and security as it reaches the TIMEFUL-WEAK privacy property while having a constant-time SEARCHID procedure.

5 Conclusion

In this paper, we have exhibited and modeled a new attack based on the time required for a tag authentication. Lots of existing protocols are not resistant to this kind of attack, even when the adversary is not able to compromise the secret key of a tag (*i.e.* a WEAK adversary). However, we have also displayed some solutions to solve this problem. Some of them reduce the efficiency of the protocol, some others increase tremendously the data storage. To the best of our knowledge, there are nowadays no solutions (in secret-key infrastructure) which are at the same time efficient, TIMEFUL private and require a really small database. To our point of view, this problematic is really interesting and solutions to this problem should be found.

References

1. B. Alomair, A. Clark, J. Cuellar, and R. Poovendran. Scalable RFID Systems: a Privacy-Preserving Protocol with Constant-Time Identification. In *40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – DSN’10*, Chicago, IL, USA, 2010. IEEE.
2. G. Avoine. Adversary Model for Radio Frequency Identification. Technical Report LASEC-REPORT-2005-001, Swiss Federal Institute of Technology (EPFL), Security and Cryptography Laboratory (LASEC), Lausanne, Switzerland, 2005.
3. G. Avoine, E. Dysli, and P. Oechslin. Reducing Time Complexity in RFID Systems. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography – SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 291–306, Kingston, Canada, 2005. Springer.

4. G. Avoine and P. 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, HI, USA, 2005. IEEE.
5. J. Bringer, H. Chabanne, and T. Icart. Efficient Zero-Knowledge Identification Schemes which respect Privacy. In *ACM Symposium on Information, Computer and Communication Security – ASIACCS’09*, Sydney, Australia, 2009. ACM.
6. M. Burmester, B. de Medeiros, and R. Motta. Robust, Anonymous RFID Authentication with Constant Key-Lookup. *Cryptology ePrint*, Report 2007/402, 2007.
7. M. Burmester, B. de Medeiros, and R. Motta. Anonymous RFID authentication supporting constant-cost key-lookup against active adversaries. *Journal of Applied Cryptography*, 1(2):79–90, 2008.
8. M. Burmester, T. v. Le, and B. d. Medeiros. Provably Secure Ubiquitous Systems: Universally Composable RFID Authentication Protocols. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm*, Baltimore, MD, USA, 2006. IEEE.
9. S. Canard and I. Coisel. Data Synchronization in Privacy-Preserving RFID Authentication Schemes. In *RFIDSec’08*, Budapest, Hungary, 2008.
10. S. Canard, I. Coisel, and J. Etrog. Lighten Encryption Schemes for Secure and Private RFID Systems. In *1st International Workshop on Lightweight Cryptography for Resource-Constrained Devices – WLC’10*, Lecture Notes in Computer Science, Tenerife, Spain, 2010. Springer.
11. I. Coisel. Authentification et Anonymat à Bas-Coût : Modélisations et Protocoles, 2009. Thèse, Université de Caen.
12. T. Dimitriou. A Lightweight RFID Protocol to protect against Traceability and Cloning attacks. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm*, Athens, Greece, 2005. IEEE.
13. M. Girault, L. Juniot, and M. Robshaw. The feasibility of on-the-tag public key cryptography. In *RFIDSec’07*, Malaga, Spain, 2007.
14. T. Instruments. Animal Tracking. <http://www.ti.com/rfid/shtml/apps-anim-tracking.shtml>.
15. A. Juels and S. Weis. Defining Strong Privacy for RFID. In *International Conference on Pervasive Computing and Communications – PerCom 2007*, pages 342–347, New York City, NY, USA, 2007. IEEE.
16. T. V. Le, M. Burmester, and B. de Medeiros. Universally Composable and Forward-secure RFID Authentication and Authenticated Key Exchange. In *ACM Symposium on Information, Computer and Communications Security – ASIACCS 2007*, pages 242–252, Singapore, 2007. ACM.
17. D. Molnar and D. Wagner. Privacy and Security in Library RFID: Issues, Practices, and Architectures. In B. Pfitzmann and P. Liu, editors, *Conference on Computer and Communications Security – ACM CCS*, pages 210–219, Washington, DC, USA, 2004. ACM.
18. P. Oechslin. Making a faster cryptanalytic time-memory trade-off. In *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630, Santa Barbara, CA, USA, 2003. Springer.
19. M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic Approach to “Privacy-Friendly” Tags. In *RFID Privacy Workshop*, MIT, MA, USA, 2003.
20. Y. Oren and M. Feldhofer. WIPR - a Public Key Implementation on Two Grains of Sand. In *RFIDSec’08*, Budapest, Hungary, 2008.
21. Y. Oren and M. Feldhofer. A Low-Resource Public-Key Identification Scheme for RFID Tags and Sensor Nodes. In *Second ACM Conference on Wireless Network Security – WiSec’09*, Zurich, Switzerland, 2009. ACM.

22. I. C. A. Organization. Machine Readable Travel Documents, Doc 9303, Part 1, Machine Readable Passports, Fifth Edition (2003).
23. N. Semiconductors. MIFARE Smartcards ICs. http://www.nxp.com/products/identification/card_ics/mifare.
24. T. van Deursen, S. Mauw, and S. Radomirović. Untraceability of RFID Protocols. In *Information Security Theory and Practices. Smart Devices, Convergence and Next Generation Networks*, volume 5019 of *Lecture Notes in Computer Science*, pages 1–15, Sevilla, Spain, 2008. Springer.
25. S. Vaudenay. On Privacy Models for RFID. In *Advances in Cryptology - ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87, Kuching, Malaysia, 2007. Springer.
26. Verayo. Anti-Counterfeiting Solution for Pharma, Liquor, Cigarettes, Food, Luxury Products. <http://www.verayo.com/solution/anti-counterfeiting.html>.
27. S. Weis, S. Sarma, R. Rivest, and D. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *First International Conference on Security in Pervasive Computing – SPC 2003*, volume 2802 of *Lecture Notes in Computer Science*, pages 454–469, Boppard, Germany, 2003. Springer.
28. J. Wu and D. Stinson. How to Improve Security and Reduce Hardware Demands of the WIPR RFID Protocol. In *IEEE International Conference on RFID – RFID 2009*, Orlando, FL, USA, 2009.

A The Optimization for O-FRAP

Burmeister, de Medeiros and Motta have introduced in [6, 7] an improvement of the O-FRAP protocol, here denoted O-FRAP.v2, where the SEARCHID procedure is from now on in a constant-time. To obtain this result, they introduce in the O-FRAP protocol a new value for each tag which can be viewed as a pseudonym. During the protocol, the tag sends this value joined with an authentication value. As this scheme is build to ensure the unlinkability, this pseudonym must change between each (successful or not) authentication protocol. It happens in two different ways, depending if the tag suspects an attack or not. To prevent entrapment attacks, if the tag does not receive the confirmation that a reader authenticates itself, it will not update its pseudonym but a counter and compute on-the-fly a pseudonym based on the counter.

The database contains for each tag its identifier, its secret-key, and all its possible pseudonyms ($2M + 3$ values, where M is the highest value of the counter m). By storing this huge amount of data, a reader is able to perform a really fast SEARCHID procedure as all the possible tag answers are already precomputed in the database. The price to pay to obtain this result is a large database where the size is parametrized by M (and the number of tags). On the other hand, when the reader receives an answer from a tag, the SEARCHID procedure only verifies that the received value belongs to the database. If it is the case, the reader has authenticated the tag and the end of the protocol consists of an update procedure of the database keys and of the tag.

As presented in [6, 7], the search procedure is in constant-time, but under certain conditions, the reader will spend more or less time to output its result. For example, if the tag uses an entrapment value for ps (i.e. $g(\text{tk}[\text{ID}]; q||\text{IV}||M)$),

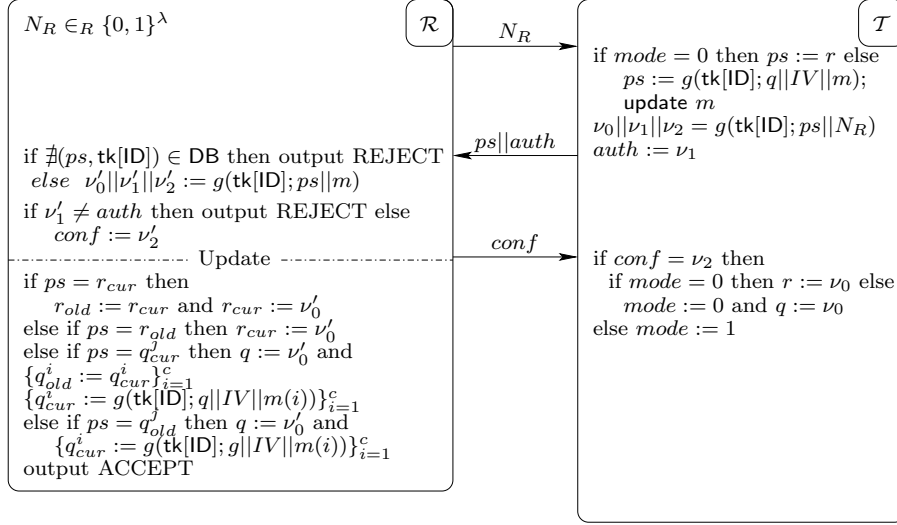


Fig. 4. Constant-Lookup Protocol Based on O-FRAP

the reader has to compute c values to replace each of the q_{cur}^i . On the other hand, if $ps = r$, the reader has no computation to realize. As a consequence, the time to output the result is different between these two cases and the adversary is thus able to distinguish if the reader has performed or not this computation. By stopping the protocol before the tag receives $conf$, the adversary forces the tag to use an entrapment value and will consequently be able to distinguish this tag of a “synchronized” one (*i.e.* where $mode = 0$) during the next authentication protocol. A trivial solution to this attack is to output the result before processing to the values’ update.

However, this protocol suffers of security flaws which allow a FORWARD adversary to trace all the previous authentications of a tag, and a WEAK adversary to rely two authentications of a tag.

For the first attack, it is sufficient to notice that an adversary who learns the secret key k of a tag is able to recompute all the previous values $auth$ of this tag, as she can recompute $g(k; ps || c)$ (because ps and c are sent in clear).

The second attack is based on the value ps sent by the tag. During a standard protocol, a tag sends $ps = r$. If the adversary blocks the last message, the tag updates nothing except its value $mode$ which is instantiated to 1. During the next protocol, the tag uses an entrapment value as $mode = 1$. When the protocol ends, the tag changes $mode$ to 0 and updates q to ν_0 . Consequently, during the next protocol, the tag will send $ps = r$ where the value r is the same as in the first protocol because it has never been updated. Thus, the adversary can trivially recognize this tag.