# Improved Privacy of the Tree-Based Hash protocols using Physically Unclonable Function

Julien Bringer[1], Hervé Chabanne[1] and Thomas Icart[1,2]

[1]Sagem Sécurité
[2]Université du Luxembourg
`firstname.name@sagem.com`

**Abstract.** In 2004, Molnar and Wagner introduced a very appealing scheme dedicated to the identification of RFID tags. Their protocol relies on a binary tree of secrets which are shared – for all nodes except the leaves – amongst the tags. Hence the compromise of one tag also has implications on the other tags with whom it shares keys. We describe a new man-in-the-middle attack against this protocol which allows to break privacy even without opening tags. Moreover, it can be applied to some other RFID protocols which use correlated keys as the one described recently by Damgård and Pedersen at CT-RSA 2008.
We introduce a modification of the initial scheme to allow us to thwart this and to strengthen RFID tags by implementing secrets with Physical Obfuscated Keys (POKs). This doing, we augment tags and scheme privacy, particularly general resistance against physical threats.
**Keywords.** RFID tags, Tree-Based Hash Protocol, POK, PUF, Privacy.

## 1 Introduction

Radio Frequency Identification (RFID) tags are made of a small chip containing a unique identification number. They communicate in the air with the system via a reader. One of their main applications is to track objects on which they are attached.

RFID systems have to deal with the scarcity of tags resources as well as the privacy needed for tag identification. In [10,11], a protocol which seems well suited to handle these two constraints has been introduced. Indeed, the identification protocol of Molnar *et al.* requires only limited cryptographic functionality and has some useful properties such as the delegation of some identifications from a Trusted Center to readers. This protocol relies on a binary tree of secrets. The secret corresponding to a leaf is uniquely associated to one tag, but all the other secrets in the tree are shared with different tags. Thus, as it is studied in [4,12,13], the compromise of the keying material of some tag leads to learn the shared keys with some other tags. If many tags are compromised, this could allow to track some non-compromised tags. This can be considered as a main threat to the privacy of the system. This problem has already been addressed in [2], the compromise of tags still leaks information about the keying material of the system.

To thwart this, we want to increase the resistance of tags against physical threats. Physical Obfuscated Keys (POKs) have been introduced by Gassend [6] as a mean to securely store a secret inside a chip. They are strongly related to Physical Unclonable Functions (PUFs). Indeed, POKs were introduced as a proposition to implement keys in a more secure manner. They are built such that their observations by an adversary corrupt the chip and then destroy them. Note that the use of PUFs inside RFID tags has already been considered in [3,15].

The main achievement of this paper is to describe how to replace each secret by two POKs during the Tree-Based Hash protocol. They are activated alternately and each one taken separately does not reveal anything on the secret. Cryptographic computations are carried out with two steps, where during a step, only one POK is activated. Moreover an adversary can gain access only to one POK by sacrificing the chip. By construction the underlying key is thus safe from this compromise of one POK.

Our paper is as follows. In Sect. 2, we recall the principles of the Tree-Based Hash protocol [11] and those of POKs. In Sect. 3 and 4, we describe our privacy model. In Sect. 5, we explain why some private informations leak with a Tree-Based Hash protocol. In fact, we show a new attack against [11] and [5] where an adversary is able to track tags even without compromising any tags. In Sect. 6, we describe our modification of the protocol. Section 7 examines the security of our proposition and Section 8 examines the privacy of our scheme to formally prove them in the random oracle model. Section 9 concludes. Security proofs and practical implementations are sketched in appendices A and B.

## 2 Preliminaries

### 2.1 The protocol [11] in a nutshell

In the following, we describe the general principles of the Tree-Based Hash protocol and invite the readers to go through [11] to get full details.

During system initialization, a Trusted Center generates a tree of secrets (keys), for instance a binary one. Each leaf is associated to a tag. A tag knows all keys $K_1, \ldots, K_d$ along the path from the root to its leaf. Let $F$ denotes an appropriate public pseudo-random function. When a tag is challenged by a reader which sends to it a random value $r$, it responds by generating a new value each time – $F_{K_1}(r, r')$, $F_{K_2}(r, r')$, …, $F_{K_d}(r, r')$ – where $r'$ is another random value generated and transmitted by the tag. The Trusted Center can easily check to which key corresponds the received value in its tree of secrets by verifying for a given $(r, r')$:

1. to which node corresponds $F_{K_1}(r, r')$,
2. between the 2 children of this node, which one is associated with $F_{K_2}(r, r')$,
3. repeat this verification, level after level from the root to the leaves,
4. and then identify which leaf (tag) comes with $F_{K_d}(r, r')$.

**A practical example.** To get a better idea of the involved figures, we take back the example given in [11]. They have $2^{20}$ tags. The binary tree is replaced by a tree with a branching factor $Q = 2^{10}$ and is made of two levels. Each tag stores two 64-bit secrets. Using a Tree-Based identification protocol enables to reduce the number of tests a Trusted Center needs to do. In this example, a Trusted Center has to compute only $2 \times 2^{10}$ times the function F, $2^{10}$ for each round, instead of $2^{20}$ without this protocol. This improvement is very interesting, because if the system's size is $S$, the number of computation for the Trusted Center is always in $O(log_Q(S)Q)$ computation.

It should be noted that this protocol is very similar to a popular RFIDs singulation algorithm: the tree walking algorithm [1]. Using this protocol leads to an optimized singulation.

### 2.2 Physical Unclonable Function and Physically Obfuscated Key

Gassend in [6] introduces the concept of PUF. A Physical Unclonable Function (PUF) is a function that maps challenges (stimuli) to responses, that is embodied by a physical device, and that has the following properties:

1. easy to evaluate,
2. hard to characterize, from physical observation or from chosen challenge-response pairs,
3. hard to reproduce.

For a given challenge, a PUF always gives the same answer. The hardness of characterization and the reproduction is hard; i.e. it is impossible to reproduce or to characterize the PUF thanks to a reasonable amount of resources (time, money, ...). PUF can thus be viewed as pseudo-random function[1] where the randomness is insured thanks to physical properties. In the rest of this paper,

[1] Note however that they can be limited in the number of possible challenge-response pairs as explained in [8].

PUFs are formalized as perfect random functions, i.e. functions with maximal output's entropy.

We also write $\mathsf{Gen}^{\mathsf{PUF}}(1^{2^k})$ for a generator of random, independent PUF$s$.

One kind of PUF, as mentioned in [15] as I-PUF for Integrated Physical Unclonable Function, has other interesting properties:

1. The I-PUF is inseparably bound to a chip. This means that any attempt to remove the PUF from the chip leads to the destruction of the PUF and of the chip.
2. It is impossible to tamper with the communication (measurement data) between the chip and the PUF.
3. The output of the PUF is inaccessible to an attacker.

These properties insure the impossibility to analyze physically a PUF without changing its output. Hence, physical attacks corrupt the PUF and the chip leaving the attacker without any information about the PUF. Silicon PUF have been already described in [7] and can be taken as relevant examples of I-PUF, they are based on delay comparison among signals running through random wires. Moreover, they only require a few resources to be implemented. A practical example of implementation is described in [14].

In [6], it is shown how to implement a key with a PUF, this implementation is called a Physically Obfuscated Key (POK), by applying a fixed hard-wired challenge to the PUF. In fact, using different challenges, several POKs can be obtained from one PUF. In the sequel, we refer to a POK as a value, stored in a tag, which is accessible only when the underlying PUF is stimulated.

## 2.3 How we use POKs

The key has to be stored digitally when involved in some computations, whatever the use of the tag is. Consequently, it could be possible to get a dump of the volatile memory and then to obtain the value of the key. This type of attack has been considered in [2] with a general line of defense for POKs: split the computations with the key in two steps. Of course, the difficulty we encounter is to cope with cryptographic computations and to find a way to split them.

A key $K$ of the tree would be hard-wired thanks to two POKs $K'$ and $K''$ such that $K = K' \oplus K''$where the two parts $K'$ and $K''$ are different for each tag.

Note that challenges used to stimulate the PUF to generate keys are stored in the tag. Because the equality $K = K' \oplus K''$ stands for all tags in the same branch, neither $K'$ and $K''$ need to be known from the outside, nor pairs of input/output from the PUF do.

## 3 Security Model

Here we propose to apply to RFID systems the following security model for completeness and soundness. This is a simplification of [16].

### 3.1 Adversary Model

We sketch the possible actions of an adversary over a system. The system contains a Trusted Center $\mathsf{TC}$ which wants to communicate with $N$ tags. We assume that the protocol is a challenge-response protocol: to authenticate a tag, the Trusted Center sends a challenge and then waits for a response from the tag.

- SendTC: this function enables the adversary to interact with the $\mathsf{TC}$. Using this function, he gets a challenge $a_0$ and he possibly tries to answer by playing the role of a tag, in order to gain information over the key material. Nevertheless, he does not receive the result of the identification.
- SendTag: this function enables an adversary to communicate with a tag. $\textsc{SendTag}(\mathcal{T}, a_0)$ means the adversary sends $a_0$ to the tag $\mathcal{T}$. This leads to the complete output from the tag.
- Result: this function allows an adversary to determine whether a bit string, taken as input by the function, is a valid communication transcript of the protocol. Result gives the authentication result the $\mathsf{TC}$ would have produced for a sent challenge and a response from a tag which are read in the input bit string.
- Corrupt: this function enables the adversary to open a tag to get all the memory, volatile and non-volatile. Corrupt enables an adversary to get keys – if any – inside a tag and to get the volatile memory at any moment of the tag computation.

We also suppose that an adversary has access to any random oracle which may be used in the protocol.

### 3.2 Completeness

**Definition 1.** *The scheme is* **complete** *when the probability of a genuine tag to fail during the identification process is negligible. I.e. for all tags $\mathcal{T}$,*

$$\Pr\left(\textsc{Result}\left(a_0^{\mathsf{TC}}, \textsc{SendTag}\left(\mathcal{T}, a_0^{\mathsf{TC}}\right)\right) = false \,|\, a_0^{\mathsf{TC}} = \textsc{SendTC}\left(\right)\right)$$

*is negligible.*

### 3.3 Soundness

**Definition 2.** *The scheme is* **sound***, if any polynomially bounded adversary $\mathcal{A}$ cannot produce a valid communication transcript $\mathcal{C}_{\mathcal{A}}$, except with a negligible probability. Furthermore, $\mathcal{C}_{\mathcal{A}}$ must neither lead to the identification of a corrupted tag nor be an eavesdropped communication. I.e.*

$$\Pr\left(\textsc{Result}\left(\mathcal{C}_{\mathcal{A}}\right) = \mathit{true}\right)$$

*is negligible.*

These definitions are the adaptation of the usual correctness and soundness in the model. Correctness ensures a legitimate tag identifies itself with an overwhelming probability. Soundness ensures that no adversary can impersonate a tag. Nevertheless, in the definition of soundness, we assume that adversaries are active. For instance, they can impersonate a $\mathsf{TC}$ or eavesdropped communications or even corrupt tags to get information on secrets of the system.

## 4 Privacy Model

We present here our model of privacy. To define privacy, we define a game. An adversary relevant against privacy is able to win this game with a non negligible probability.

Thanks to the experiment described in Fig. 1, $\mathcal{A}$ is an adversary who wants to find a privacy leakage in the protocol (where $\overset{\mathcal{R}}{\leftarrow}$ denotes an element taken at random). The privacy is defined as the advantage of the adversary over two tags amongst two systems of tags he had chosen. If the advantage of $\mathcal{A}$ is negligible, this means he is not able to link any tag inside $S_1$ and $S_2$. If $\mathcal{A}$ is relevant for this game, he is able to construct subsystems with a special property: given a tag, he can determine in which subsystem it belongs. This definition is more general than anonymity and untraceability. If tags can be identified from an adversary or can be traced, it is easy for an adversary to construct subsystems in order to succeed at our game.

**Definition 3.** *A protocol in a RFID system is* **private** *if for a polynomially bounded adversary $\mathcal{A}$ following the experiment $\mathsf{Exp}^{priv}_{\mathcal{A},S}$, then*

$$\left|\Pr[b' = b] - \Pr[b' \neq b]\right|$$

*is negligible.*

Experiment $\mathsf{Exp}^{priv}_{\mathcal{A},S}$:

**Setup:**

1. Initialize one system $S$.

**Phase 1 (learning):**

1. $\mathcal{A}$ may do the following in any interleaved order:
   (a) make arbitrary SENDTAG queries to any tag in $S$,
   (b) make arbitrary SENDTC queries,
   (c) make arbitrary RESULT queries,
   (d) make arbitrary CORRUPT queries to any tag in $S$,
   (e) make arbitrary calls to the random oracle.

**Phase 2 (challenge):**

1. $\mathcal{A}$ selects two subset of $S$: $S_1$ and $S_2$,
2. $\mathcal{A}$ selects two non corrupted tags $\mathcal{T}_1 \in S_1$ and $\mathcal{T}_2 \in S_2$.
3. Remove $\mathcal{T}_1$ and $\mathcal{T}_2$ from $S_1$ and $S_2$.
4. Let $b \xleftarrow{\mathcal{R}} \{1,2\}$ to select $\mathcal{T}_b$ one of these tags.
5. $\mathcal{A}$ may do the following in any interleaved order:
   (a) make arbitrary SENDTAG queries to any tag in $S_1 \backslash \mathcal{T}_1$, $S_2 \backslash \mathcal{T}_2$ and $\mathcal{T}_b$,
   (b) make arbitrary SENDTC queries,
   (c) make arbitrary RESULT queries,
   (d) make arbitrary CORRUPT queries to any tag in $S_1 \backslash \mathcal{T}_1$, $S_2 \backslash \mathcal{T}_2$,
   (e) make arbitrary calls to the random oracle.
6. $\mathcal{A}$ outputs a guess index $b'$

$\mathsf{Exp}^{priv}_{\mathcal{A},S}$ succeeds if $b = b'$.

**Fig. 1.** Privacy experiment

In each step, $\mathcal{A}$ is allowed to use the random oracle, but we omit it to simplify.

This privacy definition is more general than the privacy definition of Juels and Weis in [9]. This is a consequence of the possibility to consider shared keys inside tags whereas it is not taken in account in their model. Indeed in their model, they suppose keys inside tags are all independent. In this case, it is unnecessary to consider the whole system to determine whether an adversary has advantages on distinguishing two tags, whereas it is an important threat to consider in Tree-Based protocols. Furthermore, the original Tree-Based Hash protocol is private in their model although it is not in ours (cf. Sect. 5).

Vaudenay in [16] defines a new model of privacy. Privacy is defined as a leakage of information of the whole system. In the Vaudenay's model, a system of tags is private if it is possible to perfectly simulate the system. An adversary should not be able to distinguish whether he is attacking a legitimate system or a simulated one. From now on, this seems to be the most general model as it is clear that a privacy leakage is a gain of information on the system. Nevertheless, a system could not be perfectly simulated – as it is the case for our scheme introduced in Sect. 6 when we allow the adversary to use the RESULT oracle – without implying that there exists a way to obtain information over tags inside the system. That is why we introduce our privacy definition which can be seen as a kind of trade-off between [9] and [16].

## 5 A new privacy leakage against Tree-Based Hash protocols

The original Tree-Based Hash protocol proposed in [11] had been proved to have some privacy leakage in [4,12,13]. Opening a tag, while keys are not protected, leads to the knowledge of shared keys in the system. Note that in one version of the protocol in [11](the one described in section 2.1), there are cases where it is possible to determine whether two tags share keys even without getting physically the keys.

Let us denote $C_1^{\mathcal{T}}(r) = F_{K_1}(r, r'), \ldots, C_d^{\mathcal{T}}(r) = F_{K_d}(r, r')$ the outputs of the tag $\mathcal{T}$ for the challenge $r$. $C_i^{\mathcal{T}}(r) = F_{K_i}(r, r')$ is the output needed to authenticate at the depth $i$ in the tree of keys. Suppose the $C_i^{\mathcal{T}}(r)$ are independent from each other. As a consequence, $C_i^{\mathcal{T}}(r)$ can be computed without the knowledge of $C_1^{\mathcal{T}}(r), \ldots, C_{i-1}^{\mathcal{T}}(r), C_{i+1}^{\mathcal{T}}(r), \ldots, C_d^{\mathcal{T}}(r)$. In this case, using one Result query and two SendTag queries, it is possible to determine whether two tags share one key.

Getting a random challenge $r$ from SendTC, the adversary can use SendTag$(\mathcal{T}, r)$ and SendTag$(\mathcal{T}', r)$. He is then in possession of two communications $C_1^{\mathcal{T}}(r), \ldots, C_d^{\mathcal{T}}(r)$ and $C_1^{\mathcal{T}'}(r), \ldots, C_d^{\mathcal{T}'}(r)$. For instance, to test whether the two tags have the same first key, the adversary uses the Result query on the communication $\left(r, C_1^{\mathcal{T}'}(r), C_2^{\mathcal{T}}(r), \ldots, C_d^{\mathcal{T}}(r)\right)$. If this communication is an admissible one, this means $\mathcal{T}$ and $\mathcal{T}'$ share the same first key. Otherwise, they do not. Of course, it is possible to do the same for a key at a different position.

This attack is practically feasible as the adversary only needs to interact with two tags and a reader. In fact, it is a general privacy threat that concerns RFID systems using correlated keys inside tags. As soon as the different components of a response (the $C_i$ above) are not linked together, an adversary can mix the answers of several tags to learn if they share keys.

For instance it is the case of the new protocol recently introduced in [5]: It is a protocol with correlated keys, but unlike [11] it does not rely on a tree of secrets in order to increase the possible choices of tuples of keys associated to tags, which allows to increase the resistance against corruption. However messages answered by a tag are still independent and the technique above still attacks the privacy of the scheme. In the next section, our protocol is constructed to avoid also this kind of vulnerability.

## 6 Our Proposition

### 6.1 System Parameters

Because of PUF and use of different random values for each key inside a tag, our protocol strengthens tags against the privacy leakage described in the previous section (see section 8 for this result).

We now give the parameters of our scheme. Our RFID system is made of $N$ tags, the tree of key has a branching factor $Q$ and a depth $d$. We use a pseudo-random function $H$ implemented by a hash function.

The length of the random challenge $a_0$ sent by the TC is $l_r$, the length of keys is $l_K$ and the length of the output of the hash function is $l_H$. The number of tags $N$ is usually smaller than $2^{40}$. A probability is considered negligible as soon as it is negligible in at least one of the following parameters: $N, l_r, l_K, l_H$. These systems is denoted $S(N, Q, d, l_r, l_H, l_K)$.

*Setup.* To create our system of tags, we need a generator function: $\mathsf{Gen}(1^k)$ outputs a random element of size $k$. To create our system of tags, we first use $Q + Q^2 + \ldots + Q^d$ times the function Gen to create our tree of keys. Each key is an output of $\mathsf{Gen}(1^{l_K})$. During the creation of a new tag, a set of keys is given, which enabled it to identify itself. The set of keys is made thanks to our tree of keys, which means it represents a path from the root to a leaf. All the tags have of course different sets of keys, with possibly $d-1$ keys shared. For one tag $\mathcal{T}$, it is denoted as $K_1^{\mathcal{T}}, \ldots, K_d^{\mathcal{T}}$. A tag is created with a new PUF obtained from $\mathsf{Gen}^{\mathsf{PUF}}(1^{2^{l_K}})$. As shown before, each key is implemented inside a tag via two POKs. To generate the value of these POKs, we once more use Gen. For each $K_i^{\mathcal{T}}$, $\mathsf{Gen}(1^{l_c})$ outputs a challenge $c$. This challenge is hard-wired with the PUF and outputs $\mathsf{PUF}(c) = K_i'^{\mathcal{T}}$. The couple of POKs associated to the key $K_i^{\mathcal{T}}$ is $(K_i'^{\mathcal{T}}, K_i''^{\mathcal{T}} = K_i'^{\mathcal{T}} \oplus K_i^{\mathcal{T}})$. As the PUF is considered to be a perfect random-function, $K_i'^{\mathcal{T}}$ and $K_i''^{\mathcal{T}}$ are considered to be random values of entropy $l_K$.

### 6.2 The Protocol

In Fig. 2 is the description of our new protocol, where $\hat{K}_i^j$ denotes the key at the depth $i$ on the $j^{th}$ branch of the tree. The TC sends to the tag a challenge $a_0$, which is a random value. The tag $\mathcal{T}$ computes a random value $r_1^{\mathcal{T}}$ and sends $a_1 = H(a_0, r_1^{\mathcal{T}})$. The tag switches on the first POK to get $K_1'^{\mathcal{T}}$ and computes $A' = r_1^{\mathcal{T}} \oplus K_1'^{\mathcal{T}}$. This operation erases in volatile memory $r_1^{\mathcal{T}}$. The second POK is switched on to get $K_1''^{\mathcal{T}}$, and this erases $K_1'^{\mathcal{T}}$. Finally the tag computes $A'' = A' \oplus K_1''^{\mathcal{T}}$ and sends $r_1^{\mathcal{T}} \oplus K_1^{\mathcal{T}}$. The tag picks a random value

| Tag $\mathcal{T}$ | | TC |
|---|---|---|
| pick $r_1^{\mathcal{T}}$ | $\xleftarrow{a_0}$ | pick $a_0$ |
| $a_1 = H(a_0, r_1^{\mathcal{T}})$ | | |
| The first POK is switched on to get $K_1'^{\mathcal{T}}$ | | |
| $A' = r_1^{\mathcal{T}} \oplus K_1'^{\mathcal{T}}$ | $\xrightarrow{a_1,\ r_1^{\mathcal{T}} \oplus K_1^{\mathcal{T}}}$ | |
| The second POK is switched on to get $K_1''^{\mathcal{T}}$ | | |
| $A'' = A' \oplus K_1''^{\mathcal{T}}$ $= r_1^{\mathcal{T}} \oplus K_1^{\mathcal{T}}$ | | |
| $\vdots$ | $\vdots$ | |
| pick $r_d^{\mathcal{T}}$ | | |
| $a_d = H(a_{d-1}, r_d^{\mathcal{T}})$ | | |
| $\vdots$ | | |
| | $\xrightarrow{a_d,\ r_d^{\mathcal{T}} \oplus K_d^{\mathcal{T}}}$ | |
| | | *for* $i = 1$ *to* $d$ |
| | | *for* $j = 1$ *to* $Q$ |
| | | $r_i^{j\prime} = \hat{K}_i^j \oplus r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}}$ |
| | | check *if* $a_i = H(a_{i-1}, r_i^{j\prime})$ |
| | | *then* go to the next stage associated to the found key |
| | | *if* no match, fails |
| | | *end for* |
| | | *end for* |

**Fig. 2.** The identification protocol

$r_2^{\mathcal{T}}$ and computes $a_2 = H(a_1, r_2^{\mathcal{T}})$ and sends it to the TC. It computes $r_2^{\mathcal{T}} \oplus K_2^{\mathcal{T}}$ using the same tricks as before. It repeats these operations $d - 1$ times.

Then the Trusted Center (TC) tries amongst all the key $\hat{K}_1^j$ in the tree's first level whether it gets the equality $H(a_0, r_1^{\mathcal{T}} \oplus K_1^{\mathcal{T}} \oplus \hat{K}_1^j) = a_1$. If it finds one correct key, it searches the next key amongst the possible keys in the tree. If the operation is successful for the $d$ levels then the tag is authenticated.

This protocol has all the advantages of the Tree-Based Hash protocols: it allows delegation and to have less computation for the TC than the exhaustive search but with an increased time of computation for the tag.

## 7 Security Analysis

In our protocol, SendTag outputs $(a_1, r_1^{\mathcal{T}} \oplus K_1^{\mathcal{T}}, \dots, a_d, r_d^{\mathcal{T}} \oplus K_d^{\mathcal{T}})$ and Result returns whether the $2d+1$-tuple $(a_0, a_1, r_1^{\mathcal{T}} \oplus K_1^{\mathcal{T}} \dots, a_d, r_d^{\mathcal{T}} \oplus K_d^{\mathcal{T}})$ is correct.

### 7.1 Restriction on the Corrupt query due to POKs

In our case, we make the hypothesis that corrupting a tag $\mathcal{T}$ leads an adversary to the knowledge of only one of the three possible type of sets:

1. $a_{i-1}$, $r_i^{\mathcal{T}}$ and $a_i = H(a_{i-1}, r_i^{\mathcal{T}})$,
2. $a_{i-1}$, $r_i^{\mathcal{T}}$, $a_i = H(a_{i-1}, r_i^{\mathcal{T}})$ and $r_i^{\mathcal{T}} \oplus K_i'^{\mathcal{T}}$,
3. $a_{i-1}$, $a_i = H(a_{i-1}, r_i^{\mathcal{T}})$, $r_i^{\mathcal{T}} \oplus K_i''^{\mathcal{T}}$ and $r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}}$.

Note that in the two first cases, an adversary does not learn the final output. Thanks to the possible actions of the adversary as defined in Sect. 3.1, we can prove:

**Theorem 1.** Corrupt *queries leak at most as many information on the key material as* SendTag *queries.*

*Proof.* Getting $a_{i-1}$, $r_i^{\mathcal{T}}$ and $a_i$ is trivially of no interest. Getting $a_{i-1}$, $r_i^{\mathcal{T}}$, $a_i$ and $r_i^{\mathcal{T}} \oplus K_i'^{\mathcal{T}}$ is equivalent as getting $a_{i-1}$, $r_i^{\mathcal{T}}$ and $K_i'^{\mathcal{T}}$. As $a_{i-1}$, $r_i^{\mathcal{T}}$, and $K_i'^{\mathcal{T}}$ are random, this leaks no information about $K_i^{\mathcal{T}}$. Finally, getting $a_{i-1}$, $a_i$, $r_i^{\mathcal{T}} \oplus K_i''^{\mathcal{T}}$ and $r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}}$ is equivalent as getting $a_{i-1}$, $a_i$, $K_i''^{\mathcal{T}}$ and $r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}}$. Because $K_i''^{\mathcal{T}}$ is random, this is equivalent as getting $a_{i-1}$, $a_i$, and $r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}}$ which is exactly a part of an output of a SendTag query. □

In the sequel, we do not distinguish Corrupt from SendTag in proofs.

*Remark 1.* Formalizing Corrupt this way is convenient for our model and our proofs. The reality behind this formalization is still an open implementation issue. More concretely, the ability of an adversary to obtain a key from a POK without destroying the tag has to be evaluated precisely. This topic is however outside the scope of this paper.

### 7.2 Completeness and Soundness

**Theorem 2.** *Our scheme is complete. If $H$ is a random oracle, then our scheme is sound.*

The proofs are available in Appendix A.

## 8 Privacy Analysis

As shown before in Sect. 5, this is an important point to determine whether an adversary gains any advantage using different outputs from different tags while he is using Result queries. In this paper, the protocol described has the property that an adversary cannot use different outputs from tags to make a new one which has a good probability of being admissible. This is shown in the following.

**Proposition 1.** *An adversary, by mixing different outputs from different tags in a* Result *query gets a positive answer only with a negligible probability.*

*Proof.* $\mathcal{A}$ uses the SendTag query on tags in $S$. Then he uses the Result query. His query is of the form $a_0, \ldots, a_{i-1}^{\mathcal{T}^1}, r_{i-1}^{\mathcal{T}^2} \oplus K_{i-1}^{\mathcal{T}^2}, a_i^{\mathcal{T}^3}, r_i^{\mathcal{T}^4} \oplus K_i^{\mathcal{T}^4}, \ldots$ To be a valid communication, it has to exist a key $\hat{K}$ such that $a_i^{\mathcal{T}^3} = H(a_{i-1}^{\mathcal{T}^1}, r_i^{\mathcal{T}^4} \oplus K_i^{\mathcal{T}^4} \oplus \hat{K})$. We also have the equality $a_i^{\mathcal{T}^3} = H(a_{i-1}^{\mathcal{T}^3}, r_i^{\mathcal{T}^3})$. If the first equality occurs, while $\mathcal{T}^1$, $\mathcal{T}^3$ and $\mathcal{T}^4$ represent different tags, this leads to a collision on the output of the random oracle as $r_i^{\mathcal{T}^3}$ and $r_i^{\mathcal{T}^4}$ are generated randomly and $a_{i-1}^{\mathcal{T}^1}$ and $a_{i-1}^{\mathcal{T}^3}$ are outputs from the random oracle. So this proves this communication is valid with a negligible probability. □
Hence, in the sequel, we suppose an adversary never uses different SendTag outputs in one Result query.

*Remark 2.* Furthermore, if $\mathcal{A}$ tries some Result queries on a randomly modified communication from one tag, he gets a positive answer with a negligible probability. Consequently, Result query can just be used to verify whether a communication from **one** tag is valid or not.

As above, the random oracle $H$ represents the hash function used in our protocol and we assume that the random generator in each tag is perfect.

**Theorem 3.** *Our protocol, in the random oracle model, is private.*

*Proof.* We denote $M$ the maximum number of computation made by $\mathcal{A}$. Let $L_{\mathcal{C}}^{i,lp}$ be the list of all the communications of $\mathcal{A}$ with tags in $S_i$ during the learning phase and $L_{\mathcal{C}}^{i,cp}$ during the challenge phase except $\mathcal{T}_b$. Let $L_{\mathcal{C}}^{b}$ be the communication with $\mathcal{T}_b$. Let $L_{\mathrm{R}}^{lp}$ and $L_{\mathrm{R}}^{cp}$ the Result (or SendTC or Corrupt) queries used in the experiment. Let $L^{lp} = L_{\mathcal{C}}^{1,lp} \cup L_{\mathcal{C}}^{2,lp} \cup L_{\mathrm{R}}^{lp}$ and $L^{cp} = L_{\mathcal{C}}^{1,cp} \cup L_{\mathcal{C}}^{2,cp} \cup L_{\mathrm{R}}^{cp}$. Let $L^1$ be $L_{\mathcal{C}}^{1,lp} \cup L_{\mathcal{C}}^{1,cp}$ and $L^2$ be $L_{\mathcal{C}}^{2,lp} \cup L_{\mathcal{C}}^{2,cp}$.

To determine whether $\mathcal{T}_b$ is in $S_1$ or $S_2$, $\mathcal{A}$ has to determine whether $\mathcal{T}_b$ shares keys with tags in $S_1$ or in $S_2$. To achieve this, either he made some queries to the random oracle or not.

- **Case 1**: $\mathcal{A}$ did not make any random oracle query. So $\mathcal{A}$ can obtain a clue that $\mathcal{T}_b$ is in $S_i$ just by looking at $L^{lp}$, $L^{cp}$ and $L^b_{\mathcal{C}}$. We already proved in Sect. 8 that use of RESULT only helps to verify whether a communication from one tag is valid or not. To get a useful information, $\mathcal{A}$ has to compare the communications in $L^1$, $L^2$ and $L^b_{\mathcal{C}}$. Amongst triplets $(a^{\mathcal{T}}_{i-1}, a^{\mathcal{T}}_i, r^{\mathcal{T}}_i \oplus K^{\mathcal{T}}_i)$, $\mathcal{A}$ has to find one triplet of $L^b_{\mathcal{C}}$ and one of $L^1 \cup L^2$ such that they have two elements equal. In this case, if the third elements are equal then the two tags share one key, otherwise they have different keys. Because there are only polynomially many triplets, and because the probability to get such a collision is negligible, the overall probability to find a collision is negligible, as the advantage of $\mathcal{A}$ in this case is.
- **Case 2**: $\mathcal{A}$ made some random oracle queries but none of the form $H(a^{\mathcal{T}}_i, r^{\mathcal{T}}_i \oplus K^{\mathcal{T}}_i \oplus \hat{K})$ where $\hat{K}$ is a key in $S_1$ or $S_2$ and $a^{\mathcal{T}}_i, r^{\mathcal{T}}_i \oplus K^{\mathcal{T}}_i$ is a part of an output from a tag. In this case, $\mathcal{A}$ has no more information than in the previous case, and the conclusion is the same
- **Case 3**: $\mathcal{A}$ made some random oracle queries and one of them is of the form $H(a^{\mathcal{T}}_i, r^{\mathcal{T}}_i \oplus K^{\mathcal{T}}_i \oplus \hat{K})$. This means $\mathcal{A}$ got a key of this system. As we already proved in a previous remark, this event has a negligible probability to happen.

The overall advantage of $\mathcal{A}$ is negligible in the security parameters for a polynomially bounded adversary. $\square$

## 9 Conclusion

Following a general trend in inserting PUFs inside RFIDs, we modify the Tree-Based Hash protocols to allow the integration of POKs. Because of the fact that keys inside a tag are now physically obfuscated, we show that an adversary is not able to impersonate a tag. Moreover, we prove our tag system has no privacy leakage. We thus believe that our work helps to strengthen the security of the overall protocol.

## References

1. Auto-ID Center. *Draft protocol specification for a 900 MHz Class 0 Radio Frequency Identification Tag*, 2003.
2. G. Avoine, L. Buttyán, T. Holczer, and I. Vajda. Group-based private authentication. In Proceedings of the International Workshop on Trust, Security, and Privacy for Ubiquitous Computing (TSPUC 2007), IEEE., 2007.
3. Leonid Bolotnyy and Gabriel Robins. Physically Unclonable Function-based security and privacy in RFID systems. In *International Conference on Pervasive Computing and Communications – PerCom 2007*, pages 211–220, New York, USA, March 2007. IEEE Computer Society Press.

4. Levente Buttyán, Tamás Holczer, and István Vajda. Optimal key-trees for tree-based private authentication. In *Privacy Enhancing Technologies*, pages 332–350, 2006.
5. Ivan Damgård and Michael Østergaard Pedersen. RFID security: Tradeoffs between security and efficiency. In *CT-RSA 2008*, 2008.
6. Blaise Gassend. Physical random functions. Master's thesis, Computation Structures Group, Computer Science and Artificial Intelligence Laboratory, MIT, 2003.
7. Blaise Gassend, Dwaine E. Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In Vijayalakshmi Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 148–160. ACM, 2002.
8. Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer, 2007.
9. Ari Juels and Stephen A. Weis. Defining strong privacy for RFID. In *PERCOMW'07*, pages 342–347, Washington, DC, USA, 2007. IEEE Computer Society.
10. D. Molnar and D. Wagner. Privacy and security in library RFID: issues, practices, and architectures. In *Proceedings of the ACMConference on Computer and Communications Security*, pages 210–219, 2004.
11. David Molnar, Andrea Soppera, and David Wagner. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In *Selected Areas in Cryptography*, pages 276–290, 2005.
12. Yasunobu Nohara, Sozo Inoue, Kensube Baba, and Hiroto Yasuura. Quantitative evaluation of unlinkable id matching systems. In *Workshop on Privacy in the Electronic Society*, 2006.
13. Karsten Nohl and David Evans. Quantifying information leakage in tree-based hash protocols (short paper). In *ICICS*, pages 228–237, 2006.
14. G. Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *DAC*, pages 9–14. IEEE, 2007.
15. Pim Tuyls and Lejla Batina. RFID-tags for anti-counterfeiting. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 115–131. Springer, 2006.
16. Serge Vaudenay. On privacy models for rfid. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87. Springer, 2007.

## A Security Proofs

### Completeness

In our scheme, errors could occur because of collisions in the output of the hash function. For instance, a part of a communication $(a_{i-1}, H(a_{i-1}, r_i^{\mathcal{T}}), r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}})$ could lead to an error when for a tag $\mathcal{T}'$, we get the equality $H(a_{i-1}, r_i^{\mathcal{T}}) = H(a_{i-1}, r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}'})$. This could appear with a probability at most $\frac{Q}{2^{l_H}}$. Because there are $d$ stages, the overall probability to fail in the identification is $O(\frac{dQ}{2^{l_H}})$ which is negligible in the parameters.

### Soundness.

We first remark the following. Let us define a family of functions $H_{a,b}$ derived from the random oracle $H$. $b$ is a bit string of size $l_b$. $H_{a,b}$ is a function from $\{0,1\}^{l_b}$ to $\{0,1\}^{l_H}$ such that $H_{a,b}(x) = H(a, b \oplus x)$. As $H$ is a random oracle, we can consider that an adversary has a negligible probability to find a preimage of $H_{a,b}(x)$ whatever $a$ and $b$ are. If there are polynomially many $a_i$ and $b_j$, an adversary has a negligible probability to find $x$ even if he knows $H_{a_i,b_j}(x)$ for all $a_i$ and $b_j$.

Now, we denote $L_1$ the list of all the communications produced via the SENDTAG queries, $L_2$ the communications sent to the $\mathsf{TC}$ either with the SENDTC query or the RESULT query.

To simplify the notation, we prove that the scheme is sound with $d = 1$. This is a sufficient condition, as the difficulty to authenticate increases with $d$. We denote $M$, the maximum number of operations made by $\mathcal{A}$.

Assume $\mathcal{A}$ has received the challenge $a_0^{\mathsf{TC}}$ and outputs the couple $(a_1, x_1)$. We overestimate the probability of success of $\mathcal{A}$. There are two cases:

- **Case 1**: $\mathcal{A}$ did not use the random oracle to output $a_1$. This means:
  - either $a_0^{\mathsf{TC}}$ had been tested by $\mathcal{A}$ thanks to the SENDTAG query, this could arise with a probability less than $\frac{M}{2^{l_r}}$,
  - or he tried a random answer. In this sub-case, he has a probability of success less than $\frac{M.Q}{2^{l_H}}$.
- **Case 2**: $\mathcal{A}$ used the random oracle to output $a_1$. So we denote $a_1 = H(a_0^{\mathsf{TC}}, x_1')$. This means
  - either there is no key $\hat{K}$ like $x_1' = x_1 \oplus \hat{K}$. The probability of success is thus less than $\frac{M.Q}{2^{l_H}}$,
  - or there is a key $\hat{K}$ in the key material such that $x_1' = x_1 \oplus \hat{K}$. Consequently $\mathcal{A}$ possesses one key. He could achieve this only using the information from $L_1$ and $L_2$. $\mathcal{A}$ only knows triplets of the form $\left(a_0, H(a_0, r^{\mathcal{T}}),\ r^{\mathcal{T}} \oplus K^{\mathcal{T}}\right)$ for some tags $\mathcal{T}$. A change of variable leads to: $a_0, H_{a_0,r^{\mathcal{T}}}(K^{\mathcal{T}}),\ r^{\mathcal{T}}$. Thanks to the previous remark, we can conclude that the probability $\mathcal{A}$ got one key is negligible.

We can conclude that our scheme is sound as the overall probability of any adversary is negligible.

## B Practical example

We propose for our protocol, as an example, the following parameters:

- the size of the reader challenge $l_r$ is 64,
- the size of any $\mathsf{POK}$ $l_K$ is 100
- the size of the output of $H$ $l_H$ is 64. For instance, the first 64 bits of $AES_{a_{i-1,1..28}||r_i}(a_{i-1}||r_{i,1..64})$.

They have been chosen to minimize the non-volatile memory inside the tag and the communication between tags and readers, but they should lead to a sufficient security to insure the secrecy of the keys and the impossibility to authenticate without the knowledge of the keys. We use $AES$ as it is possible to implement it with not too many gates and because the problem to find a preimage or any collision is usually believed intractable.

Security can be improved by increasing the parameters:

- the size of the reader challenge $l_r$ is 64,
- the size of any POK $l_K$ is 116
- the size of the output of $H$ $l_H$ is 64. For instance, the first 64 bits of $AES_{a_{i-1,1..12}||r_i}(a_{i-1}||r_{i,1..64})$.

The two tables Tab. 1 and Tab. 2 summarize the concrete resources used in our scheme in the two previous cases for some example parameters. We use a branching factor of $2^{10}$ in all cases.

| Tag | | | Tag → TC | TC |
|---|---|---|---|---|
| numbers | non-volatile memory | computation | communication | computation |
| $2^{20}$ | 200 bits | 2 $AES$ and 2 random | 328 bits | $2 \times 2^{10}$ |
| $2^{30}$ | 300 bits | 3 $AES$ and 3 random | 492 bits | $3 \times 2^{10}$ |

**Table 1.** Resources needed in the first case

| Tag | | | Tag → TC | TC |
|---|---|---|---|---|
| numbers | non-volatile memory | computation | communication | computation |
| $2^{20}$ | 232 bits | 2 $AES$ and 2 random | 360 bits | $2 \times 2^{10}$ |
| $2^{30}$ | 348 bits | 3 $AES$ and 3 random | 540 bits | $3 \times 2^{10}$ |

**Table 2.** Resources needed in the second case