Lightweight Authentication Protocol for Low-Cost RFID Tags

Pierre Dusart[1] and Sinaly Traoré[2]

[1] XLIM - UMR CNRS n°7252, Faculté des Sciences et Techniques,
123 Avenue Albert THOMAS, 87060 LIMOGES CEDEX, France
pierre.dusart@xlim.fr

[2] Faculté des Sciences et Techniques, USTTB, BP E 32 06 BAMAKO, MALI
sinaic2002@yahoo.fr

Abstract. Providing security in low-cost RFID (Radio Frequency Identification) tag systems is a challenging task because low-cost tags cannot support strong cryptography which needs costly resources. Special lightweight algorithms and protocols need to be designed to take into account the tag constraints. In this paper, we propose a function and a protocol to ensure pre-shared key authentication.

Keywords: RFID, Authentication.

1 Introduction

In the future, optical bar codes based systems will be replaced by Radio Frequency Identification systems. These systems are composed of two parts:

– a RFID tag which replaces the bar code;
– a RFID reader which handles information send from the tag.

The tag consists of a microchip which communicates with a reader through a small integrated antenna. Various external form factors can be used: the tag can look like a sheet of paper, like a plastic card or can be integrated below bar code for backward device's compatibility.

RFID tags offer many advantages over optical bar codes [1]:

– the use of microchip enables a range of functionalities like computing capability or readable/writable storage. The stored data, depending on the capacity of the tag, can be static identification number up to rewritable user data.
– the use of RF antenna enables communication between the reader and the tag without line of sight from a distance of several decimeters [2]. A reader can communicate sequentially with up to hundred tags per second.

To provide further functionalities than bar codes, the tag may require data storage. For example, the price of a product can be stored into the tag [3]. To know the price of a product, the customer can ask directly the tag instead of asking the database server connected with the cash register. With these new features, the adoption of RFID technology is growing: inventory without unpacking [4], prevention of counterfeiting [5], quality chain with environmental sensing [6] are deployed applications. The tag systems can be easily adapted for universal deployment by various industries with low prices.

But a new technology must also take into account problems inherited from legacy systems. For example in a shop, security problems to deal with are:

– an item is changed to another (it means for RFID to substitute a tag for a fake one);
– a price is changed without authorization by a malicious user (it means for RFID, to write a tag), ...

In addition, the privacy problem must be considered in some context i.e. an user must not reveal unintentionally information about himself. It means for RFID, the ability of a tag to reveal its identity only to authenticated partners.

To cope with security and privacy problems, the first idea is to use asymmetric cryptography (e.g. RSA [7]) like in public key infrastructures. Unfortunately tags with strong cryptography [8] and tamper resistant hardware [9] are too expensive for a wide deployment.

Hence a constraint class of cryptography [10], named Lightweight Cryptography, appears.

The aim of this paper is to propose a protocol and its related computational function. Section 2 introduces the system model and the underlying assumptions for our protocol. Then related work is presented in section 3. The protocol environment is described in section 4. Section 5 presents the protocol details and the computational functions. Section 6 provides an analysis of some security constraints and shows that the protocol satisfies the lightweight class. Section 7 illustrates how our protocol behaves against cryptographic attacks.

2 System Model and Assumptions

We consider a system with one RFID tag reading system and several low cost RFID tags. We assume that each tag shares a secret K with the reader, which is shared in a secure manner before the beginning of the communication (e.g. in manufacturing stage). The aim of the communication is to authenticate the tag i.e. find its identity and prove that it belongs to the system (by knowing the same secret).

The tag is passively powered by the reader, thus:

– the communication needs to be short (speed and simplicity of an algorithm are usually qualifying factors);
– the communication can be interrupted at any time if the reader does not supply enough energy to the tag.

For cost reasons, the standard cryptographic primitives (hash function, digital signature, encryption) are not implemented (no enough computation power is available or too much memory is required). Hence, we need a protocol using primitives with a low complexity. This property which is named "Lightweight property" [10] consists to use basic boolean operations like XOR, AND, ...

The security of protocols needs also a good random number generator [11]. This part can be assumed by the reader environment where the features can be higher and costly (e.g. a computer connected with a tag reading system).

3 Related Work

The RFID technology needs security mechanisms to ensure the tag identity. Hence a tag spoofing, where an attacker replaces the genuine tag by its own creation, is defeated if good authentication mechanisms are used. But classical authentication solutions use cryptographic primitives like AES [12] or hash functions (SHA1 [13] or MD5 [14]) which are not adapted to low cost RFID tags. It is thus necessary to look for new suitable primitives for this specific constraint resources environment. In [15–18], authors suggest some protocol families based on elementary arithmetic (e.g. binary bit addition or modular addition by a power of 2). However in [19], B. Defend *et al.* put in defect XOR and SUBSET protocols given in [15] by learning key sequence. They proved that with few resources, an attacker can recover the session keys of these two protocols. The LMAP, M^2AP and EMAP protocols proposed respectively in [16–18] allow a mutual authentication between the reader and the tag but are also completely broken [20] by key recovery attacks. In [21], the authors proposed a family of protocols, called S-protocols, based on a family of generic random number generators that they introduced in the same paper. They presented a formal proof which guarantees the resistance of the S-protocol against the attacks of desynchronization [22, 23] and impersonation [24]. With a small modification, they proposed the family of S^*-protocols, which not only has the properties of S-protocols but also allows a mutual authentication between the reader and the tag. However authors do not show that their generic functions are compatible with lightweight RFID tags. In [25], Yeh proposes a protocol corrected by Habibi [26], but attacks [27] appear using $O(2^{17})$ off-line evaluations of the main function. Recently, some protocols are also defined in ISO/IEC WD 26167-6. Since they use AES engine [28], they are out of the scope of this paper.

4 Protocol Requirements and Specifications

We want to use a very simple dedicated protocol which uses a non-invertible function h. We provide a protocol in which the tag identity is sent in a secure manner and the tag is authenticated according to a challenge given by the reader. Then the reader shows that it knows a secret key by calculating an answer to the tag challenge.

We present the authentication protocol: the reader needs to verify the identity of the tag. For the verification of the tag identity iD, the RFID reader $\mathcal{R}$ sends to the tag $\mathcal{T}$ a challenge C. Next, the tag proves its identity iD by computing a response using the common secret K, shared with the reader. We avoid taking $K = 0$ for a maximum security. Denoting by $Auth$ this response, the authentication phase is presented in the following scheme:

- $\mathcal{R} \longrightarrow \mathcal{T} : C = (C_0, C_1, \ldots, C_{15})$ where C_i are bytes randomly chosen.
- $\mathcal{T} \longrightarrow \mathcal{R} : Auth = [iD \oplus h_K(C), h_{iD}(C)]$

To verify, the reader computes $h_K(C)$ using its challenge C and the key K and then it can retrieve the identity of the tag. Next the authentication of the tag can be verified by computing $h_{iD}(C)$ using the result of previous computation and the first challenge. The protocol allows card authentication by the reader. It can be adapted to allow mutual authentication with a slightly modification: a challenge C' (which can be a counter) is sent with the tag response *Auth*. Next the reader should respond with the computation of $h_{K\oplus C'}(C \oplus iD)$.

5 Proposal Description

Our protocol uses a function h that is composed of two sub-functions S and f taking respectively one and two bytes as input. The function h used in the protocol must be lightweight (for low-cost devices) and satisfy some properties:

- must be a like a one-way function (from output, input cannot be retrieved);
- its output must seem to be random;
- its output length must be sufficient to have enough intrinsically security (to avoid replay and exhaustive authentication search).

We define an input size and an output size of 16 bytes for h and the same size for the secret key K. Output size is chosen to be presented in the 16-byte form to iterate an algorithm defined on byte. Function f which processes byte data blocks and a substitution function S are described in the following subsections.

5.1 Function Design

***f* Function.** Here we define the function f which needs two input bytes to produce an output result of one byte.

$$\begin{aligned} f : \mathbb{F}_{256} \times \mathbb{F}_{256} &\longrightarrow \mathbb{F}_{256} \\ (x, y) &\longmapsto z \end{aligned}$$

with

$$z := \big[\, [x \oplus ((255 - y) \ll 1)] + 16 \cdot [((255 - x) \oplus (y \gg 1)) \bmod 16] \,\big] \bmod 256, \quad (1)$$

where $\oplus$ is the bitwise exclusive or, $+$ represents the classical integer addition, $n \gg 1$ divides n by 2, $n \ll 1$ multiplies n by 2 and keeps the result modulo 256 by not taking into account a possible overflow and "16·" is the classical multiplication by 16. In the subsection 6.2, we explain how to keep lightweight these various operations by using 8-bit registers.

We have the following properties:

– f is non-symmetric, i.e., for all (x, y) pair in $\mathbb{F}_{256} \times \mathbb{F}_{256}$, the function verifies $f(x, y) \neq f(y, x)$;
– f has a uniform distribution of values, i.e., for all z in $\mathbb{F}_{256}$, the function verifies

$$\sharp\{(x, y) \in \mathbb{F}_{256} \times \mathbb{F}_{256} : f(x, y) = z\} = 256.$$

These properties can be easily verified. Hence we consider that the f function is one-way: one cannot retrieve the good (x, y)-entry with the z value. The function h inherits of this property.

Let $i \in \{0, \cdots, 15\}$ a vector index and $j \in \{1, 2, 3, 4\}$ a round index. Let $M = (M_0, \cdots, M_{15})$ a vector of 16 bytes. The function f does not use the same entries depending on a vector index i and a round index j. We define:

$$F_i^j(M) = f(M_i, M_{(i+2^{j-1}) \bmod 16}).$$

and

$$F^j(M) = (F_0^j(M), F_1^j(M), \cdots, F_{15}^j(M)).$$

A working example of these indexes can be found in the table 2.

***S* Function.** Our S function is not a new one. We choose the AES [12, 29] SubBytes function for the quality of its properties.

The SubBytes transformation is a non-linear byte substitution. For example, the eight-bits data “00000000” is transformed into $B =$ “01100011”.

To avoid attacks based on simple algebraic properties, the definition of SubBytes Transformation is the composition of the following two transformations in the finite field $\mathbb{F}_{2^8}$ with a chosen structure representation $\mathbb{F}_{2^8} \approx \mathbb{F}_2(X)/(X^8 + X^4 + X^3 + X + 1)$.

The first transformation is the multiplicative inverse in Galois Field $GF(2^8)$, known to have good non-linearity properties. Then the multiplicative inverse of each element is taken (the 8bit-element “00000000”, or $\{00\}$ in hexadecimal format, is mapped to itself). Next, the previous result is combined with an invertible affine transformation:

$$x \mapsto Ax \oplus B,$$

where A is a 8×8 fixed matrix over $GF(2)$ and B is the number defined above and $\oplus$ operates “Exclusive Or” on the individual bits in a byte.

The SubBytes Transformation is also chosen to avoid any fixed point ($S(a) \neq a$), any opposite fixed point ($S(a) \neq \bar{a}$) and also any self invertible point ($S(a) \neq S^{-1}(a)$).

Because it is based on many mathematical objects, the SubBytes function could seem difficult to implement but the transformation could be reduce in an 8-bit substitution box. Hence for any element the result can be found by looking up in a table (see the Figure 7 of [12]: substitution values for the byte $\{xy\}$ (in hexadecimal format)).

We define by S the following transformation: let $M = (M_0, \cdots, M_{15})$ a 16-byte vector. Let S the function which associates M with the vector

$$S(M) = (\text{SubBytes}(M_0), \cdots, \text{SubBytes}(M_{15})).$$

5.2 Description of the Authentication Function $h : (C, K) \longrightarrow H$

Formally, we will follow the tag computation. First, we add the challenge to key by Xor operation, i.e. we calculate $D = C \oplus K = (C_0 \oplus K_0, \ldots, C_{15} \oplus K_{15})$. Then we apply the substitution S to D. The first state M^0 is initialized by $M^0 = S(D)$. Then, we calculate the following values:

$$\begin{aligned} M^1 &= S(F^1(M^0)) \oplus K, \\ M^2 &= S(F^2(M^1))) \oplus K, \\ M^3 &= S(F^3(M^2))) \oplus K, \\ M^4 &= S(F^4(M^3))) \oplus K. \end{aligned}$$

Finally, the function returns $H = M^4 = (M^4_0, \ldots, M^4_{15})$. We denote the result H by $h_K(C)$.

The figure 1 summarizes this description and a more classical definition can be found through the algorithm 1.

```
**Input** : $C, K$
**Output** : $H$
  $M^0 = S(C \oplus K)$
**for** $j = 1$ to 4 do
  $M^j = S(F^j(M^{j-1})) \oplus K$
**end for**
  $H = M^4$
**return** $H$
```

Fig. 1. Authentication Function

6 Analysis

6.1 Protocol Security

The identity of the tag is not revealed directly: the tag's identity iD is masked by $h_K(C)$, output of h function which appears random. But the reader can still determine the iD identity using the shared secret key K. The reader verifies that this identity has been used to compute the second part of authentication. At this state, the reader is sure that the tag with iD identity knows the secret key K.

But as aforementioned section 4, a mutual authentication can be set by adding the following steps. The reader shows that it knows K and iD by computing $h_{K \oplus C'}(C \oplus iD)$ where C' is the challenge given by the tag. The tag authenticates the reader by computing in the same way and comparing the proposed result with the computed one. If they are equal, the mutual authentication is achieved.

Algorithm 1. Tag computations

```
Input: $C = (C_0, \ldots, C_{15})$, $K = (K_0, \ldots, K_{15})$
Output: $H = (H_0, \ldots, H_{15})$
  {Comment: Computation of $M^0 = S(C \oplus K)$}
  for $i = 0$ to 15 do
    $M_i \leftarrow S(C_i \oplus K_i)$
  end for
  {Comment: Computation of $S(F^j(M^{j-1}))$}
  for $j = 1$ to 4 do
    for $i = 0$ to 15 do
      $k \leftarrow M_i \oplus ((M_{i+2^{j-1} \bmod 16}) \ll 1)$
      $l \leftarrow (255 - M_i) \oplus (M_{i+2^{j-1} \bmod 16} \gg 1) \bmod 16$
      $t \leftarrow (k + 16\, l) \bmod 256$
      $Temp_i \leftarrow S(t)$
    end for
    {Comment: Computation of $M^{j+1} = M^j \oplus K$}
    for $i = 0$ to 15 do
      $M_i \leftarrow Temp_i \oplus K_i$
    end for
  end for
  for $i = 0$ to 15 do
    $H_i \leftarrow M_i$
  end for
  return $H$
```

Now we consider two cases:

– Fake Tag: the tag receives the challenge C. It can choose arbitrarily a number iD to enter into the system. But it does not know K to compute the first part of authentication response.
– Fake reader: the reader chooses and sends C. Next it receives a proper tag authentication. It cannot find iD thanks to $h_{iD}(C)$ (because h is a one-way function) nor K.

6.2 Lightweight

We have to establish that function could be programmed using usual assembler instructions. We refer to ASM51 Assembler [30]. First we use 8-bit registers. To represent an entry of 128 bits, eight registers or space blocks must be reserved.

Next we can implement the f function defined by (1) using very simple instructions using a register named A and a carry C:

– The computation of $A \ll 1$ can be translated by `CLR` C (Clear Carry) followed by `RLC` A (Rotate Left through Carry). The computation of $A \gg 1$ can be translated by `RRC` A (Rotate Right through Carry).
– The computation of $255 - A$ can be translated by `CPL` A, the complemented value.

- The bitwise-xor is classically translated by `XRL`.
- The modular reduction by 16 can by translated by `AND 0x0F`.
- The multiplication by 16 can be translated by four left shift or by `AND 0x0F` followed by `SWAP` which swaps nibbles.
- The modular addition (mod 256) can be translated simply by `ADD` without taking care of possible carries of an 8-bit register.

The SubBytes function can be implemented by looking up in a table as explain in the Figure 7 of [12]. This part of AES algorithm can be computed with a few gates compared to the whole AES (The most penalizing part being the key expansion according to the table 3 of [31]).

Now we claim that properties of h function presented in section 5 are satisfied:

- the overflows of f are intended and contribute to the non-reversibility of the h function,
- the output seems random (subsection 6.4),
- the avalanche criterion (subsection 6.3) shows that the outputs distribution of f is well reported to h outputs.

6.3 Strict Avalanche Criterion

The strict avalanche criterion was originally presented in [32], as a generalization of the avalanche effect [33]. It was introduced for measuring the amount of nonlinearity in substitution boxes (S-boxes), like in the Advanced Encryption Standard (AES).

The avalanche effect tries to reflect the intuitive idea of high-nonlinearity: a very small difference in the input producing a high change in the output, thus an avalanche of changes.

Denote by HW the Hamming weight and $DH(x, y) = HW(x \oplus y)$ the Hamming distance.

Mathematically, the avalanche effect can be formalized by

$$\forall x, y | DH(x, y) = 1, \quad \text{average}(DH(F(x), F(y))) = \frac{n}{2},$$

where F is candidate to have the avalanche effect.

So the output of a n-bit random input number and one generated by randomly flipping one of its bits should be, on average, $n/2$. That is, a minimum input change (one single bit) is amplified and produces a maximum output change (half of the bits) on average.

First we show that if an input bit is changed then the modification will change an average of one half of the following byte. The input byte x will be changed to x' with a difference Δx of one bit. After the first SubBytes transformation, the difference will be

$$S(x \oplus k) \oplus S(x' \oplus k) = S(y) \oplus S(y + \Delta x),$$

with $y = x \oplus k$. We have in average

$$\frac{1}{256 \cdot 8} \sum_{y} \sum_{\Delta x, HW(\Delta x)=1} HW(S(y) \oplus S(y + \Delta x)) \approx 4,$$

where HW is the Hamming weight. Hence an average of four bits will change if the difference is of one bit. Furthermore, for any difference Δx,

$$\frac{1}{256 \cdot 256} \sum_{y} \sum_{\Delta x} HW(S(y) \oplus S(y + \Delta x)) = 4.$$

Our function satisfies the avalanche effect as

$$\frac{1}{256^2} \sum_{x} \sum_{y} HW(x \oplus S(f(x, y))) \approx 4.$$

Next we show that if an input bit is changed then the modification will be spread over all the bytes of the output. Suppose that a bit of the k^{th} byte M_k^0 is changed ($1 \leq k \leq 16$). Then M^1 is also changed as the SubBytes substitution is not a constant function. At the first round, the bytes k and $k + 1$ will be modified. At the second round, the bytes k, $k + 2$, $k + 1$ and $k + 3$ will be modified. Furthermore, eight bytes will be modified and at the end, the whole 16 bytes will be modified.

For example, if the first input byte is changed (M_0^0 is changed). Then M_0^0 is used for compute M_0^1 and M_{15}^1, hence a difference appears in M_0^1 and M_{15}^1, and so on. We trace the difference diffusion in the following table:

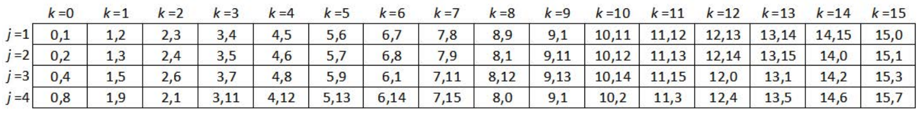
Table 1. Diffusion table

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
First Xor	x															
$j = 1$	x															x
$j = 2$	x	x													x	x
$j = 3$	x	x									x	x	x	x	x	x
$j = 4$	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Last Xor	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

If another byte is changed, the same remark works by looking in the dependence table 2.

Hence for any input difference, the modification will change an average of one half of the output.

Table 2. Dependency table



	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$	$k=14$	$k=15$
$j=1$	0,1	1,2	2,3	3,4	4,5	5,6	6,7	7,8	8,9	9,1	10,11	11,12	12,13	13,14	14,15	15,0
$j=2$	0,2	1,3	2,4	3,5	4,6	5,7	6,8	7,9	8,1	9,11	10,12	11,13	12,14	13,15	14,0	15,1
$j=3$	0,4	1,5	2,6	3,7	4,8	5,9	6,1	7,11	8,12	9,13	10,14	11,15	12,0	13,1	14,2	15,3
$j=4$	0,8	1,9	2,1	3,11	4,12	5,13	6,14	7,15	8,0	9,1	10,2	11,3	12,4	13,5	14,6	15,7

6.4 Security Quality

To evaluate the security quality, we take $Y = 1$ et $X = 0$. We consider the iterated outputs of the authentication function. Hence we test the series $h_Y(X)$, $h_Y(h_Y(X))$, ... like a random bitstream with the NIST test suite [34]. The bitstream satisfies all the tests (parameters of NIST software: 10^6 input bits, 100 bitstreams).

Table 3. NIST Statistical Test Results

Test Name	Percentage of passing sequences with Significance level $\alpha = 0.01$
1. Frequency Test (Monobit)	99/100
2. Frequency Test (Block)	100/100
3. Runs Test	100/100
4. Longest Run of Ones	99/100
5. Binary Matrix Rank Test	98/100
6. Discrete Fourier Transform Test	98/100
7. Non-Overlapping Template	98/100
8. Overlapping Template	98/100
9. Maurers Universal Statistical	100/100
10 Linear Complexity Test	100/100
11. Serial Test	99/100
12. Approximate Entropy Test	100/100
13. Cumulative Sums (Cusum) Test	98/100
14. Random Excursions Test	90/93
15. Random Excursion Variant Test	91/93

6.5 Hardware Complexity: Implementation and Computational Cost

We choose a 8bit-CPU Tag for cost reasons. We implement the authentication function on a MULTOS Card [35] without difficulties. This card is not a low-cost card but we only test the implementation with basic instructions. The code size of the authentication function (with S-box table) without manual optimization is 798 bytes.

We can optimize the memory usage:

– the S-box table can be placed in Read-Only memory area: 256 bytes needed for AES SubBytes Table.
– the variables placed in the Random Access Memory Memory can be optimized. For internal state computation, one have to represent M with 16 bytes and we need two supplementary temporary bytes: at each round, a state byte value M_i is used twice to compute the next state. In fact M_i^j is used for compute M_i^{j+1} and $M_{i+2^{j-1} \bmod 16}^{j+1}$. After computation of these two variables, the space allocation for the variable M_i^j can be reused. Next we compute the value $M_{i+2^{j-1} \bmod 16}^{j+1}$ depending on $M_{i+2^{j-1} \bmod 16}^{j}$ and another

byte. Now we can delete the memory space for $M^j_{i+2^{j-1} \bmod 16}$ and compute another byte of M_{j+1}, step by step. Hence we use only two additional bytes to compute the next state of M.

We evaluate the computational time with a PC computer (Intel CoreDuo T9600 2.8Ghz): 30 s for 10^7 authentications for a program in a C language, i.e. $3\mu s$ per authentication.

6.6 Privacy

Even if RFID technology is used for identify something in tracing system, in many cases this technology would merely cause infringements of private rights. We do not prevent the tracing system from recording informations but we need to protect the tag iD from external recording. Hence if an attacker records all transactions between tag and a reader, he cannot retrieve if the same tag has been read one or many times. Contrarily, a fake reader can determine if it has previously ask a tag by sending always the same challenge and recording responses, but it cannot know the real iD of the tag.

7 Attacks

The attacker's aim is to validate its tag identity. He can do this by producing a response to a challenge. If he can exploit the attack in a feasible way, then we say that the protocol is broken. Such a success of the attacker might be achieved with or without recovering the secret key shared by the reader and the tag. Hence a large key size is not enough to prove that the protocol cannot be broken with brute force attack. We might also take into account other attacks where the attacker can record, measure and study the tag responses. The necessary data could be obtained in a passive or in an active manner. In case of a passive attack, the attacker collects messages from one or more runs without interfering with the communication between the parties. In case of an active attack, the attacker impersonates the reader and/or the tag, and typically replays purposefully modified messages observed in previous runs of the protocol.

7.1 Recording Attacks

Replay Attack by Recording: An attacker tries to extract the secret of a tag. He uses a reader and knows the commands to perform exchanges with the tag. He asks the tag many times. By listening to different requests, one can record n complete answers. A complete record is composed of a challenge C and the associated response *Auth*. Next if a recording challenge C is used or reused, then the attacker knows the correct response *Auth*. This attack works but

- The attacker must have time to record all the possibilities;
- To create a fake tag, the tag must have $2^{128} \cdot (2 \cdot 2^{128})$ bits (e.g. 10^{60} To) of memory to store the previous records and have the good answer. If this type of tag exists, it is not a commercial one.

– The challenge C, generated by the reader environment, is supposed to be random. So for a fixed C, the probability to have the good answer is very low.

Relay Attack [36]: the attacker makes a link between the reader and tag; it's a kind of Man-in-the-Middle attack. He creates independent connections with reader and tag and relays messages between them. Hence a tag can be identified without being in the reader area. The problem can be treated by security environment protections. A partial solution to protect tag against this attack [37] is to limit its communication distance, but this countermeasure limits the potential of RFID tags. A better way is to activate a distance-bounding protocol [38].

Man-In-The-Middle Attack: A man-in-the-middle attack is not possible because our proposal is based on a mutual authentication, in which two random numbers (C, C'), refreshed at each iteration of the protocol, are used. One cannot forge new responses using challenge differences because $h_{iD}(C+\Delta) \neq h_{iD}(C)+\Delta$ and $h_K(C+\Delta) \neq h_K(C)+\Delta$. In the same way, $h_{K\oplus C'\oplus\Delta}(C\oplus iD) \neq h_{K\oplus C'}(C\oplus iD) \oplus \Delta$.

7.2 Side Channels Attacks

Timing Attack: a timing attack [39] is a side channel attack in which the attacker attempts to compromise a cryptosystem by analyzing the time taken to execute cryptographic algorithm. The attack exploits the fact that every operation in a computer takes a dedicated time to execute. If the time cost of operation depends on key value or input values, on can retrieve these secret values by timing attack. Hence, during the implementation, we must be aware of the timing attack. For the computation of tag authentication, the time cost of the operations is the same whatever the value of the key. Next for the reader authentication, the tag must compare the reader response with its own computation. With poor security implementation but unfortunately "classical", if a difference between two bytes is found, the algorithm stops and return the information "Authentication failed". This kind of program is sensible to timing attack. The execution time is different according if the value is rapidly found or not found. To be immune from this attack, we make always a fixed number of steps; the response is send when all the response is verified. One can also add dummy cycles to equilibrate the parts of an implementation. Hence our function is resistant to Timing attack.

Power Consumption Attack: an attacker studies the power consumption [40] of the tag. He can do it by monitoring the delivery power from the reader to the tag. As the consumption of the chip depends on the executed instructions, the attacker can observe (SPA) the different parts of an algorithm. Here the algorithm does not need to be secret and the operations do not depend on the key values. One can also use random dummy cycles to disrupt the observation of the same part of program execution. Hence our function is SPA-resistant.

7.3 Mathematical Attacks

Lucky Authentication: A attacker tries to have a good authentication with a fake tag. He sends (C_1, C_2) as $Auth$. The first part $C_1 = iD \oplus h_K(C)$ of the response can be decoded as a existing iD if there is enough tags. But the second part $C_2 = h_{iD}(C)$ is fixed by the decoding iD and the challenge C. The size of C_2 is 16 bytes. Hence

$$P(\text{Authentication OK/False Tag}) \leq \frac{1}{2^{128}}.$$

Nowadays, this probability is sufficient for a good security.

Active Attack: Suppose that an attacker queries the tag $\mathcal{T}$ by sending $C = 0$ as challenge. Then, to determine the secret K, it must solve the equation

$$S(F^4(S(F^3(S(F^2(S(F^1(S(K))) \oplus K)) \oplus K)) \oplus K)) \oplus K = H, \tag{2}$$

where H is the response of $\mathcal{T}$ and the unknowns are the bytes of K. Since each round of the algorithm operations are performed modulo 16 or modulo 256 and the results from these transactions are processed by substitution tables, the equation 2 is very difficult to analyze algebraically.

Linear [41] or Differential [42] Attacks: These attacks depend especially on properties of the substitution function. First remember that for a function g from $\mathbb{F}_{2^m}$ to $\mathbb{F}_{2^m}$, a differential pair (α, β) is linked with the equation $g(x \oplus \alpha) \oplus g(x) = \beta$. The differential attack is based on finding pairs where the probability

$$P(\sharp\{x \in \mathbb{F}_{2^m} : g(x \oplus \alpha) \oplus g(x) = \beta\})$$

is high. If such pair exists then the attack is feasible. Our function is well resistant to this attack. Indeed the substitution function S is constructed by composing a power function with an affine map, which avoid from differential attacks. Our h function inherits from these properties: considering the output z of $f(x, y)$ describes in the paragraph 5.1, it is easy to verify (like in the paragraph 6.3) that

$$\text{for all} \quad \alpha, \beta \in \mathbb{F}_{256}, \quad \sharp\{z \in \mathbb{F}_{256} : S(z \oplus \alpha) \oplus S(z) = \beta\} \leq 4.$$

It allows to avoid the existence of differential pair such that the probability

$$P(\sharp\{x \in \mathbb{F}_{256} : S(x \oplus \alpha) \oplus S(x) = \beta\})$$

be high.

To achieve a linear attack, it aims at awarding credibilities to the equations of the type

$$\langle \alpha, x \rangle \oplus \langle \beta, S(x) \rangle = 0, \quad \text{with} \quad \alpha, \beta \in \mathbb{F}_{256}.$$

We know that for all α and β not identically equal to zero, the equation has a number of solutions close to 128 which makes expensive the linear attack.

7.4 Desynchronizing Attack

In a desynchronization attack, the adversary aims to disrupt the key update leaving the tag and reader in a desynchronized state in which future authentication would be impossible. Compared to some other protocols, the key does not change in our authentication protocol. It is not a lack of security, the key may change during stocktaking or subscription renewal, by changing tag by another with the new key.

8 Conclusion

We have presented a lightweight authentication protocol for low-cost RFID tags. The internal functions are well adapted for 8-bit CPU with few memory and without cryptoprocessor, even if it is true that a precise evaluation of the building cost and performance of a tag supporting our protocol (i.e. very few CPU functions and less than 1Kbytes of memory) should be evaluated with a manufacturer.

We use the security qualities of the AES S-Boxes to build a function, specifically dedicated to the authentication, which keeps them. The notions of privacy and the classic attacks are addressed. The proposed version is light in terms of implementation and in a reduced cost what makes it usable on RFID systems. Even if these systems are intended for simple applications as secure counter of photocopies or stock management in a small shop, the security level reached here allows to envisage more ambitious applications.

Acknowledgements. The authors want to thank the anonymous reviewers for their constructive comments which were helpful to improve this paper and Damien Sauveron for proofreading of preliminary versions.

References

1. Agarwal, A., Mitra, M.: RFID: Promises and Problems (April 2006)
2. Weis, S.A.: Rfid (radio frequency identification): Principles and applications
3. Nath, B., Reynolds, F., Want, R.: Rfid technology and applications. IEEE Pervasive Computing 5(1), 22–24 (2006)
4. Östman, H.: Rfid - 5 most common applications on the shop floor (2012), `http://www.rfidarena.com/2012/12/13/rfid-%E2%80%93-5-most-common-applications-on-the-shop-floor.aspx`
5. James, J.: Fda, companies test rfid tracking to prevent drug counterfeiting. AIDS Treat News (417), 5–8 (2005)
6. Miles, S., Sarma, S., Williams, J.: RFID Technology and Applications. Cambridge University Press (2011)
7. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21(2), 120–126 (1978)
8. Feldhofer, M., Wolkerstorfer, J.: Strong crypto for rfid tags - a comparison of low-power hardware implementations. In: ISCAS, pp. 1839–1842. IEEE (2007)

9. Kömmerling, O., Kuhn, M.G.: Design principles for tamper-resistant smartcard processors. In: Proceedings of the USENIX Workshop on Smartcard Technology, p. 2. USENIX Association (1999)
10. Poschmann, A.: Lightweight cryptography - cryptographic engineering for a pervasive world. IACR Cryptology ePrint Archive 2009, 516 (2009)
11. Hellekalek, P.: Good random number generators are (not so) easy to find. Math. Comput. Simul. 46(5-6), 485–505 (1998)
12. NIST: Advanced encryption standard (aes), fips 197 (November 2001), `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`
13. Eastlake, D.E., Jones, P.E.: US Secure Hash Algorithm 1 (SHA1), `http://www.ietf.org/rfc/rfc3174.txt?number=3174`
14. Rivest, R.L.: The MD5 Message-Digest Algorithm (RFC 1321), `http://www.ietf.org/rfc/rfc1321.txt?number=1321`
15. Vajda, I., Buttyán, L.: Lightweight authentication protocols for low-cost rfid tags. In: 2nd Workshop on Security in Ubiquitous Computing, in conjunction with Ubicomp 2003 (October 2003)
16. Peris-Lopez, P., Hern, J.C., Tapiador, J.M.E., Ribagorda, A.: Lmap: A real lightweight mutual authentication protocol for low-cost rfid tags. In: Proc. of 2nd Workshop on RFID Security, Ecrypt, p. 06 (2006)
17. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: M^2AP: A minimalist mutual-authentication protocol for low-cost RFID tags. In: Ma, J., Jin, H., Yang, L.T., Tsai, J.J.-P. (eds.) UIC 2006. LNCS, vol. 4159, pp. 912–923. Springer, Heidelberg (2006)
18. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: EMAP: An efficient mutual-authentication protocol for low-cost RFID tags. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM Workshops 2006, Part I. LNCS, vol. 4277, pp. 352–361. Springer, Heidelberg (2006)
19. Defend, B., Fu, K., Juels, A.: Cryptanalysis of two lightweight rfid authentication schemes. In: PerCom Workshops, pp. 211–216. IEEE Computer Society (2007)
20. Li, T., Wang, G.: Security analysis of two ultra-lightweight RFID authentication protocols. In: Venter, H., Eloff, M., Labuschagne, L., Eloff, J., von Solms, R. (eds.) New Approaches for Security, Privacy and Trust in Complex Environments. IFIP, vol. 232, pp. 109–120. Springer, Boston (2007)
21. Lee, J., Yeom, Y.: Efficient rfid authentication protocols based on pseudorandom sequence generators. IACR Cryptology ePrint Archive 2008, 343 (2008)
22. Lo, N.W., Yeh, K.H.: De-synchronization attack on rfid authentication protocols. In: International Symposium on Information Theory and its Applications (ISITA), pp. 566–570 (October 2010)
23. van Deursen, T., Radomirovic, S.: Security of rfid protocols - a case study. Electr. Notes Theor. Comput. Sci. 244, 41–52 (2009)
24. Sixth International Conference on Availability, Reliability and Security, ARES 2011, Vienna, Austria, August 22-26. IEEE (2011)
25. Yeh, T.C., Wang, Y.J., Kuo, T.C., Wang, S.S.: Securing rfid systems conforming to epc class 1 generation 2 standard. Expert Syst. Appl. 37(12), 7678–7683 (2010)
26. Habibi, M.H., Alagheband, M.R., Aref, M.R.: Attacks on a lightweight mutual authentication protocol under EPC C-1 G-2 standard. In: Ardagna, C.A., Zhou, J. (eds.) WISTP 2011. LNCS, vol. 6633, pp. 254–263. Springer, Heidelberg (2011)
27. Hernandez-Castro, J.C., Peris-Lopez, P., Safkhani, M., Bagheri, N., Naderi, M.: Another fallen hash-based RFID authentication protocol. In: Askoxylakis, I., Pöhls, H.C., Posegga, J. (eds.) WISTP 2012. LNCS, vol. 7322, pp. 29–37. Springer, Heidelberg (2012)

28. Song, B., Hwang, J.Y., Shim, K.A.: Security improvement of an rfid security protocol of iso/iec wd 29167-6. IEEE Communications Letters 15(12), 1375–1377 (2011)
29. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
30. Intel: Mcs-51 instruction set summary (1979)
31. Hamalainen, P., Alho, T., Hannikainen, M., Hamalainen, T.D.: Design and implementation of low-area and low-power aes encryption hardware core. In: Proceedings of the 9th EUROMICRO Conference on Digital System Design, DSD 2006, pp. 577–583. IEEE Computer Society, Washington, DC (2006)
32. Forré, R.: The strict avalanche criterion: Spectral properties of boolean functions and an extended definition. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 450–468. Springer, Heidelberg (1990)
33. Webster, A.F., Tavares, S.E.: On the design of S-boxes. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 523–534. Springer, Heidelberg (1986)
34. NIST: A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications. NIST Special Publication 800-22rev1a (April 2010)
35. Multos: Multos developer's guide (2012)
36. Kasper, T., Carluccio, D., Paar, C.: An embedded system for practical security analysis of contactless smartcards. In: Sauveron, D., Markantonakis, K., Bilas, A., Quisquater, J.-J. (eds.) WISTP 2007. LNCS, vol. 4462, pp. 150–160. Springer, Heidelberg (2007)
37. Schneier, B.: Rfid cards and man-in-the-middle attacks. Schneier Security Blog (2006)
38. Hancke, G.P., Kuhn, M.G.: An rfid distance bounding protocol. In: SecureComm, pp. 67–73. IEEE (2005)
39. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
40. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
41. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
42. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991)