M^{2}AP: A Minimalist Mutual-Authentication Protocol for Low-cost RFID Tags

Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda

Computer Science Department, Carlos III University of Madrid,
{pperis,jcesar,jestevez,arturo}@inf.uc3m.es

Abstract. Low-cost Radio Frequency Identification (RFID) tags affixed to consumer items as smart labels are emerging as one of the most pervasive computing technologies in history. This presents a number of advantages, but also opens a huge number of security problems that need to be addressed before its successful deployment. Many proposals have recently appeared, but all of them are based on RFID tags using classical cryptographic primitives such as Pseudorandom Number Generators (PRNGs), hash functions, or block ciphers. We believe this assumption to be fairly unrealistic, as classical cryptographic constructions lie well beyond the computational reach of very low-cost RFID tags. A new approach is necessary to tackle the problem, so we propose a minimalist lightweight mutual authentication protocol for low-cost RFID tags that offers an adequate security level for certain applications, which could be implemented even in the most limited low-cost tags as it only needs around 300 gates.

Keywords: Ubiquitous Computing, RFID, Tag, Reader, Pseudonym, Privacy, Mutual-Authentication.

1 Introduction

At the moment, the most extended identification systems are barcodes. Recently, the mass deployment of Radio Frequency Identification (RFID) systems has taken place [8]. Around 5 billion barcodes are read daily, so efficiency gains from using RFID tags could substantially lower the cost of tagged items [9, 12]. The penetration of RFID systems is nowadays mainly limited by privacy concerns and by their cost, which must be between 0.05 and 0.1 € to be considered affordable.

The low cost demanded for RFID tags forces them to be very resource limited. Typically, they can only store hundreds of bits, have 5-10K logic gates, and a maximum communication range of a few meters. Within this gate counting, only between 250 and 3000 gates can be devoted to security functions. It is interesting to recall that for a standard implementation of the Advanced Encryption Standard (AES), 20-30K gates are needed. Additionally, power restrictions should be taken into account, since most RFID tags in use are passive. Furthermore, these systems are unable to store passwords securely because they are not tamper-resistant at all.

The remainder of the paper is organized as follows. A short review of the main problems associated with RFID systems is outlined in section 2. In section 3, we prose a minimalist lightweight mutual authentication protocol (M^2AP) for low-cost RFID tags. A security evaluation and performance analysis of this new protocol is presented in section 4. In section 5, the proposed architecture for implementing our protocol is explained in detail . Finally, the last section is devoted to some conclusions summarizing this work.

2 Risks and Threats

Although RFID systems may emerge as one of the most pervasive computing technologies in history, there are still a number of problems that need to be solved before their massive deployment. One of the fundamental issues still to be addressed is privacy. Products labeled with tags reveal sensitive information when queried by readers, and they do it indiscriminately.

A problem closely related to privacy is tracking, or violation of location privacy. This happens because the answers provided by tags are usually predictable: in fact, most of the times, tags provide the same identifier, allowing a third party to easily establish a link between a given tag and its holder or owner. Even in the case in which individual tags try not to reveal any kind of valuable information, this tracking can still be possible by using an assembly of tags (constellation), so non-trivial solutions must be applied in order to address these tracking problems.

Although the two aforementioned problems are the most important security questions that arise from RFID technology, there are some others worth mentioning: physical attacks, denial of service, counterfeiting, spoofing, eavesdropping, traffic analysis, etc.

3 Lightweight Protocol

The major challenge of providing security for low-cost RFID tags is that these devices are very limited computationally, even unable to perform the most basic cryptographic operations. Surprisingly, most of the proposed solutions are based on the use of hash functions. Since the work of Ohkubo [7] in 2003, there has been a huge number of solutions based on this idea [2, 3, 5]. Although this apparently constitutes a good and secure approach, engineers face the nontrivial problem of implementing cryptographic hash functions with only 250-3000 gates. In most of the proposals, no explicit algorithms are suggested and finding one is not an easy issue, since traditional hash functions (MD5, SHA-1, SHA-2) can not be used [10]. In [14], we can find a recent work on the implementation of a new hash function with a reduced number of gates, but although this proposal seems to be light enough to fit in a low-cost RFID tag, the security of this hash scheme remains an open question.

In this paper, we propose a lightweight mutual authentication protocol between RFID readers and tags. In the following, we consider that low-cost RFID tags are devices with a very small amount of memory and very constrained computationally (< 1000 gates).

3.1 Suppositions of the Model

Our protocol is based on the use of pseudonyms, concretely on *index-pseudonym* (*IDS*). The *index-pseudonym* (96-bit length) is the index of a table (a row) where all the information about a tag is stored. Each tag has an associated key which is divided in four parts of 96 bits ($K = K1 \parallel K2 \parallel K3 \parallel K4$). As the *IDS* and the key (K) need to be actualized each time the tag is read, we need in total 480 bits of rewritable memory (EEPROM or FRAM). We also need a ROM memory to store the static tag-identification number (ID) of 96 bits, which univocally identifies the tag.

For the implementation of our protocol, all the costly computing operations are done by the reader. We suppose that readers are devices with enough computing power to generate random numbers and, in general, to perform any kind of cryptographic operations. On the contrary, tags are very limited devices that only have around 1000 logical gates for security functions. Our proposal is based on the use of simple operations: $\oplus$, $\wedge$, $\vee$, and *sum mod m*.

The communication must be initiated by readers due to the fact that low-cost tags are passive. We also suppose that both the backward and the forward channels can be listened by an attacker, despite their asymmetry. Finally, we consider that the communication channel between the reader and the database is secure.

3.2 The Protocol

We can divide the protocol in four main stages: tag singulation, mutual authentication, index-pseudonym updating and key updating. In this section, we outline how the protocol works, while in the next one a security evaluation and a performance analysis are presented.

1. ***Tag Singulation***
   Before starting the protocol for mutual authentication, the reader should identify the tag. The reader will send a *"hello"* message to the tag, who will answer the reader by sending its current *index-pseudonym* (*IDS*). By means of the *IDS*, the reader will be able to access the tag secret key ($K = K1 \parallel K2 \parallel K3 \parallel K4$), which is necessary to carry out the next authentication stage. Only an authorized reader can access this information.
2. ***Mutual Authentication***
   Our protocol consists on the exchange of two messages between the reader and the tag. An scheme of the protocol is illustrated in *Figure 1.*

Reader $\rightarrow$ Tag:
$IDS_{tag(i)}^{(n)} \oplus K1_{tag(i)}^{(n)} \oplus n1 \parallel (IDS_{tag(i)}^{(n)} \wedge K2_{tag(i)}^{(n)}) \vee n1 \parallel IDS_{tag(i)}^{(n)} + K3_{tag(i)}^{(n)} + n2$
Tag $\rightarrow$ Reader:
$(IDS_{tag(i)}^{(n)} \vee K4_{tag(i)}^{(n)}) \wedge n2 \parallel (IDS_{tag(i)}^{(n)} + ID_{tag(i)}) \oplus n1$

Fig. 1. M^{2}AP Protocol

- Reader Authentication
  The reader will generate two random numbers, $n1$ and $n2$. With $n1$ and the subkeys $K1$ and $K2$ associated to the tag, the reader will generate $A \parallel B$:

$$A \parallel B = IDS_{tag(i)}^{(n)} \oplus K1_{tag(i)}^{(n)} \oplus n1 \parallel (IDS_{tag(i)}^{(n)} \wedge K2_{tag(i)}^{(n)}) \vee n1 \quad (1)$$

  which is the part of the message that allows the tag authentication. With $n2$ and $K3$, the reader will generate the submessage C:

$$C = IDS_{tag(i)}^{(n)} + K3_{tag(i)}^{(n)} + n2 \quad (2)$$

  that will be used for updating the *index-pseudonym* (*IDS*) and the key (K). Once the three parts of the message are generated ($A \parallel B \parallel C$), they are concatenated and sent to the tag.
- Tag Authentication
  With submessages A and B, the tag will authenticate the reader. From submessage C, the tag will obtain the random number $n2$, that will allow it to update the *index-pseudonym* (*IDS*) and the key (K). Once these verifications are performed, the tag will generate the answer message. This message will be composed of two parts $D \parallel E$. Part D,

$$D = (IDS_{tag(i)}^{(n)} \vee K4_{tag(i)}^{(n)}) \wedge n2 \quad (3)$$

  will allow the reader to authenticate the tag. By means of part E,

$$E = (IDS_{tag(i)}^{(n)} + ID_{tag(i)}) \oplus n1 \quad (4)$$

  the tag is able to transmit its static identifier in a secure form.

3. ***Index-Pseudonym Updating***
   Once the tag and the reader have been mutually authenticated, the *index-pseudonym* must be updated as follows:

$$IDS_{tag(i)}^{(n+1)} = (IDS_{tag(i)}^{(n)} + (n2 \oplus n1)) \oplus ID_{tag(i)} \quad (5)$$

4. ***Key Updating***
   Another important security issue is key updating. After the reader and the tag have been authenticated, the key updating stage must be carried out. As

tags are very computationally contrained devices, this task can be made only by using efficient operations ($\oplus$, $\wedge$, $\vee$, and *sum mod m*). These operations have to be already implemented in the tag for the normal protocol running, so its use will not imply an increase in the gate counting. Nevertheless, we can not either forget the temporary requirements of the tag which must be able to answer 100 times/sec (see section 5) at least. These speed requirements put a limitation on the number of possible operations that can be performed with each component of the key (Ki). Taking, all these considerations into account, the proposed equations for key updating are the following ones:

$$K1_{tag(i)}^{(n+1)} = K1_{tag(i)}^{(n)} \oplus n2 \oplus (K3_{tag(i)}^{(n)} + ID_{tag(i)}) \tag{6}$$
$$K2_{tag(i)}^{(n+1)} = K2_{tag(i)}^{(n)} \oplus n2 \oplus (K4_{tag(i)}^{(n)} + ID_{tag(i)}) \tag{7}$$
$$K3_{tag(i)}^{(n+1)} = (K3_{tag(i)}^{(n)} \oplus n1) + (K1_{tag(i)}^{(n)} \oplus ID_{tag(i)}) \tag{8}$$
$$K4_{tag(i)}^{(n+1)} = (K4_{tag(i)}^{(n)} \oplus n1) + (K2_{tag(i)}^{(n)} \oplus ID_{tag(i)}) \tag{9}$$

If we analyze the previous equations, we will obtain that the probability of zeros and ones for every Ki is approximately 0.5 and the Hamming distance between $K_{tag(i)}^{n}$ and $K_{tag(i)}^{n+1}$ is 47.5 (on average). According to the temporary requirements, for the worst case, which is the architecture of 8 bits, we are well into the limit of 100 answers/sec, so we successfully fulfill the temporary requirements in all the cases.

4 Evaluation

4.1 Security Analysis

Once we have presented the proposed mutual-authentication protocol, we will evaluate its security, studying the same properties that Yang analyzes in [13], in order to be able to compare their characteristics.

1. ***User Data Confidentiality***
   Tag ID must be kept secure to guarantee user privacy. The tag sends it in the message E ($E = (IDS_{tag(i)}^{(n)} + ID_{tag(i)}) \oplus n1$) where the ID is added with the *index-pseudonym*, then the result xored with the nonce $n1$. This hides tag ID to a nearby eavesdropper equipped with an RFID reader.
2. ***Tag Anonymity***
   As tag ID is static, we should send it, and all other interchanged messages in seemingly random wraps (i.e. to an eavesdropper, only random numbers are sent). As we have seen, readers generate the message $A \parallel B \parallel C$. This message will serve, as well as to authenticate the reader, to transmit in a secure form the random numbers $n1$ and $n2$ to the tag. The first random number ($n1$) will be used to hide the tag ID and the combination $n1 \oplus n2$

will serve to update the index-pseudonym. By means of this mechanism, we are able to make almost all the computational load to fall on the side of RFID readers, since one of our hypothesys is that very low-cost tags can not generate random numbers. Thus, tag anonymity is guaranteed, and the location privacy of a tag owner is not compromised either. There is one interesting scenario that we will explain in more detail in the following, as one could think that in this case, the tracking of a tag owner is possible. In this scenario, the attacker sends *"hello"* messages to the tag and receives as answer the *IDS* from it. Then, he stops the authentication step. A little time later he repeats the process, hoping that the *IDS* has not changed yet. We know that, if the authentication process fail the *IDS* can not be updated. The attacker can not generally track the owner tag because it is very probable that between two successive requests of the attacker, the tag is read by one or several legitimate readers, who will update the *IDS*. If an intruder wants to guarantee that the *IDS* has not changed, it needs to send more than 100 answers/sec in order to saturate the tag, so not allowing a legitimate reader to access it. In this case, this attack would be considered a DoS attack, which is an inherent problem in RFID technology as it happens in other technologies that use the radio channel to which, for the moment, there is no known solution (apart from spread spectrum).

3. ***Data Integrity***
   A part of the tag memory is rewritable, so modifications are possible. In this part of the memory, the tag stores the *index-pseudonym* (IDS) and the key (K) associated with itself. If an attacker does succeed in modifying this part of the memory, then the reader would not recognize the tag and should implement the updating protocol of the database.
4. ***Mutual Authentication***
   We have designed the protocol with both reader-to-tag authentication, which is achieved by message $A \parallel B \parallel C$, and tag-to-reader authentication, obtained with message $D \parallel E$.
5. ***Forward Security***
   Forward security is the property that security of messages sent today will be valid tomorrow [7]. Since key updating is fulfilled after the mutual authentication, a future security compromise on an RFID tag will not reveal any previously transmitted data.
6. ***Man-in-the-middle Attack Prevention***
   A man-in-the-middle attack is not possible because our proposal is based on a mutual authentication, in which two random numbers $n1$ and $n2$, refresh in each iteration of the protocol, are used.
7. ***Replay Attack Prevention***
   An eavesdropper could store all the messages interchanged between the reader and the tag (different protocol runs). Then, he could try to impersonate a reader, re-sending the message $A \parallel B \parallel C$ seen in any of the protocol runs. It seems that this could cause the losing of synchronization between the database and the tag, but this is not the case because

Table 1. Comparison between Protocols

Protocol	HLS [11]	EHLS [11]	HBVI [5]	MAP [13]	M^2AP
User Data Confidentiality	×	△	△	○	○
Tag Anonymity	×	△	△	○	○
Data Integrity	△	△	○	○	△
Mutual Authentication	△	△	△	○	○
Forward Security	△	△	○	○	○
Man-in-the-middle Attack Prevention	△	△	×	○	○
Replay Attack Prevention	△	△	○	○	○
Forgery Resistance	×	×	×	○	○
Data Recovery	×	×	○	○	×

†† Notation: ○ Satisfied △ Partially Satisfied × No Satisfied

after the mutual authentication, the *index-pseudonym* (*IDS*) and the key K ($K = K1 \parallel K2 \parallel K3 \parallel K4$) were updated.

8. ***Forgery Resistance***
The information stored in the tag is sent operated ($\oplus$, $\wedge$, $\vee$, or *sum mod m*) with random numbers ($n1$, $n2$). Therefore, the simple copy of information of the tag by eavesdropping is not possible.
9. ***Data Recovery***
Intercepting or blocking messages is a DoS attack that prevents tag identification. As we do not consider that these attacks can be a serious problem for very low-cost RFID tags, our protocol does not provide data recovery. In those scenarios in which this problem is considered important, an extended version of the protocol is possible and straighforward. In this implementation, each tag will have $l + 1$ database records, the first one associated with the actual *index-pseudonym* (n) and the others associated with the potential next index-pseudonyms (n+1, n+2, ... , n+l). Moreover, each tag will need k additional bits of ROM memory to store the associated data-base entry like in [5]. As before, the reader will use the *IDS* to access to all the information associated with the tag. The reader will store a potential *index-pseudonym* each time the answer of the tag is blocked. Once the tag and the reader had mutually authenticated, the potencial *index-pseudonyms* could be deleted. The storage of the potencial *index-pseudonyms* will allow to easily recover from the lose or interception of messages.

Table 1 shows a comparison made by Yang [13] of the security requirements of different proposals, our proposal (M^2AP) is added in the last column.

4.2 Performance Analysis

It is important to carefully analyze the performance of the proposed scheme, to show that it can safely be implemented even in low-cost tags. As mentioned in section 3.1, we have assumed that the connection between the reader and the database is secure. Also, readers and databases are devices with non-limited computing and storing capabilities. Due to these reasons we can collapse the notion of the reader and the back-end database into a single entity ($R+B$). So, in the performance analysis of our protocol, we will consider that reader and database form a single entity. As in the previous section, we will consider the same overheads (computation, storage and communication) used by Yang [13].

1. ***Computation Overhead***
   Low-cost RFID tags are very limited devices,with only a small amounts of memory, and very constrained computationally (only between 250 and 3000 logics gates can be devoted to security-related tasks). Additionally, one of the main drawbacks that hash-based solutions have is that the load on the server side ($R+B$) is proportional to the number of tags. As we can see in *Table 2* this problem is also present in Yang's solution [13]. On the other hand, in our proposal, we have completely solved this problem by using an *index-pseudonym* that allows a tag to be univocally identified.
2. ***Storage Overhead***
   As Yang does, we assume that the sizes of all components are L bits, that the PRNG and the hash function are $h, h_k : \{0,1\}^* \rightarrow \{0,1\}^{\frac{1}{2}L}$ and $r \in_U \{0,1\}^L$. Our protocol is based on pseudonyms, concretely on an L-bit *index-pseudonym* (*IDS*), so each tag has to store it. For the implementation of our protocol, each tag should have an associated key of length $4L$, which is used for mutual authentication between the reader and the tag. Moreover, the tag has to store an unique identification number of length L. The reader has to store the same information, so it requires a memory of $6L$ bits.
3. ***Communication Overhead***
   The proposed protocol accomplishes mutual authentication between tag (T) and reader ($R+B$), requiring only four rounds. As we can see in *Table 2*, other protocols require, at least, one or two additional messages to be exchanged. Taking into account that low cost tags are passive, and that the communication can only be initiated by a reader, four rounds may be considered as a reasonable number of rounds for mutual authentication in RFID environments.

5 Implementation

In this section, we will explain in detail the proposed architecture for implementing our protocol: the reader sends the message $A \parallel B \parallel C$, which is received by the tag. The tag will check the authenticity of this message for authenticating the reader. Once the tag has authenticated the reader, it will send the message $D \parallel E$ to authenticate itself.

One of the first and more relevant subjects to consider is whether to choose a serial or a parallel implementation. Serial means processing the bitstream bit by bit, and parallel means processing the whole message, for example, $A \parallel B \parallel C$ at

Table 2. Computational Loads and Required Memory

Protocol	Entity	HLS [11]	EHLS [11]	HBVI [5]	MAP [13]	M^2AP
No. of Hash Operations	T	1	2	3	2	$\neg$
	B	$\neg$	Nt	3	$2Nt$	$\neg$
No. of Keyed Hash Operations	R	$\neg$	$\neg$	$\neg$	1	$\neg$
	B	$\neg$	$\neg$	$\neg$	1	$\neg$
No. of PRNG Operations	T	$\neg$	1	$\neg$	$\neg$	$\neg$
	R	$\neg$	$\neg$	$\neg$	1	$\neg$
	B	$\neg$	$\neg$	1	$\neg$	$\neg$
No. of Basic Operations^1	T	$\neg$	$\neg$	$\neg$	4	19
	R+B	$\neg$	$\neg$	$\neg$	$2(Nt+1)$	21
Number of Authentication Steps		6	5	5	5	4
No. of Encryptions	B	$\neg$	$\neg$	$\neg$	1	$\neg$
No. of Decryptions	R	$\neg$	$\neg$	$\neg$	1	$\neg$
Required Memory Size	T	$1\frac{1}{2}L$	$1L$	$3L$	$2\frac{1}{2}L$	$6L$
	R+B	$2\frac{1}{2}L$	$1\frac{1}{2}L$	$9L$	$9\frac{9}{2}L$	$6L$

†† Notation:
$\neg$: Not Require Nt: Number of Tags L: Size of Required Memory
^{1} Basic Operations: $\oplus$, $\wedge$, $\vee$, and $+$

the same time. It is a common assumption that a minimum of 100 tags should be authenticated per second. As in [4], due to the low-power restrictions of RFID tags, the clock frequency must be set to 100 KHz. So, a tag may use up to 1000 clock cycles to answer a reader. Due to these characteristics, it is not necessary to resort to a parallel implementation. As we can see in *Figure 2*, we have decided not to process at the same time all the message, but to do it in blocks of m bits.

The proposed architecture is independent of the word length used. We have analyzed the features of five different word lengths ($m = 8, 16, 32, 64, 96$). In *Figure 2*, we can see a scheme of the proposed architecture. On the left of the figure, we have the memory, which is filled with the *index-pseudonym* (IDS), the key K ($K1 \parallel K2 \parallel K3 \parallel K4$) and the ID. The access to the memory is controlled by a sequencier. Due to the fact that the messages are build up of three components, we will need a m-bit register to store intermediate results. In the middle of the figure we have the Arithmetic Logic Unit (ALU). This unit will made the following operations of size m bits (word length): $\oplus$, $\wedge$, $\vee$, and *sum mod m*. The ALU has two inputs, one of these values is stored in the memory, and another which is selected (C_1) between the bitstream and the value stored in the register. The control signal C_2 will select the operation that will be used in the ALU.

In the worst case of our protocol ($m = 8$), we need 1000 clock cycles for implementing the mutual authentication. So, if we consider that the clock fre-

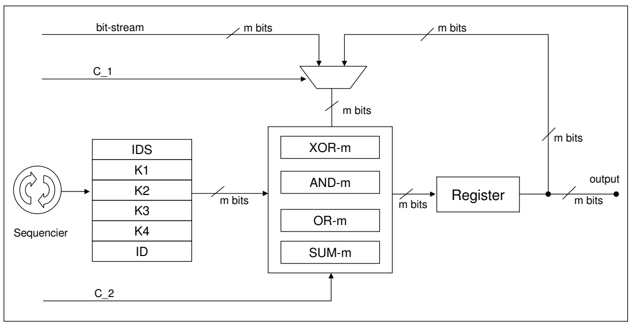

Fig. 2. Logic Scheme

quency is set to 100KHz, this means that the tag answers in 10 millisecond. A tag can authenticate 100 times per second, so the temporary requirements are fulfilled in all the cases.

Another important aspect to study is the number of logical gates necessary for implementing our protocol. The functions $\oplus$, $\wedge$, and $\vee$ will be implemented with the same number of logic gates like the word length (m). For the implementation of the adder circuit with carry, a parallel architecture is proposed ($S = A \oplus [B \oplus C_{ENT}]$; $C_{SAL} = BC_{ENT} + AC_{ENT} + AB$). Six logic gates are needed for each bit that is added in parallel. Additionally, a 20% of logic gates are considered for control functions. The following table summarizes the features of the proposed architecture:

Table 3. Features

Word Length (m)		**8-bit**	**16-bit**	**32-bit**	**64-bit**	**96-bit**
Number of	ALU	72	144	288	576	864
Gates	Control	14	29	58	115	173
	Total	86	173	346	691	1037
Number of Clock Cycles		960	480	240	120	80
Answer/sc		104	208	416	833	1250

As we can see in the previous table, in the best case ($m = 8$), our protocol needs around 100 gates. In *Table 4*, we show also the number of logical gates needed for implementing various hash functions and AES encryption. A traditional hash function such as MD5 or SHA needs more than 16K gates, which is by far higher than the capabilities of low-cost RFID tags [10]. An efficient implementation of AES encryption has been recently published [6], which does not need many logical gates (only 3595), but it needs a coprocessor. Unfortunately, this significantly increases the price of RFID tags, and is not attainable in low-cost RFIDs. Additionally, there is also a proposal of an implementation of

Table 4. Core Comparison

Solutions	Implementation	Gate Counting
Hash	Universal Hash Yksel [14]	1.7K Gates
	MD5 Helion [10]	16K Gates
	Fast SHA-1 Helion [10]	20K Gates
	Fast SHA-256 Helion [10]	23K Gates
AES Unit^2	JungFL [6] E/D/RK^2	3089 Gates +/+/+
	Feldhofer [4] E/D/RK^1	3595 Gates +/+/+
	Amphion CS5265 [1] E/D/RK^1	25000 Gates +/+/+

†† Notation:
^{1}AES with Coprocessor
^{2}E= Encryption, D=Decryption, RK= Round Key Generation

a new universal hash function for ultra low-power cryptographic hardware applications. Although this solution only needs around 1.7K gates, a deeper security analysis of the hash function is needed, and has not been accomplished yet.

Finally, although we have not implemented the circuit physically, due to the known fact that power consumption and circuit area are proportional to the number of logical gates, it seems that our implementation will be suitable even for very low-cost RFID tags.

6 Conclusions

RFIDs tags are devices with very limited computational capabilities, which only have between 250 and 3000 logics gates that can be devoted to security-related tasks. Cryptographic primitives such as PRNGs, block ciphers and hash functions lie well beyond the computational capabilities of very low-cost RFID tags, but until now, most of the security solutions for RFID are based on them.

A new approach must be taken to tackle the problem, at least for low-cost RFID tags. For this reason, we propose a very lightweight mutual authentication protocol that could be implemented in low-cost tags (<1000 logic gates). In order to be able to use our proposal, tags should be fitted with a small portion of rewritable memory (EEPROM or FRAM) and another read-only memory (ROM). The assumption of having access to rewritable memory is also implicitly made in all the existing solutions based on hash functions.

In spite of being very limited in resources, the main security aspects of RFID systems (privacy, tracking, etc.) have been consider in this article, and solved efficiently (less than 1000 gates are needed, even in the worst implementation, in our case $m = 96$ bits). As shown in *Table 2*, our protocol displays superior benefits to many of the solutions based on hash functions. So, not only we have been able to avoid the privacy and tracking problems, but also many other attacks such as man-in-the-middle attack, forwarding replay, etc.

Finally, another paramount characteristic of our scheme is its efficiency: tag identification by a valid reader does not require exhaustive search in the back-end database. Furthermore, only two messages need to be exchanged in the singulation stage, and another two in the mutual-authentication stage.

References

1. Amphion: CS5265/75 AES Simplex encryption/decryption. *http://www.amphion.com*, 2005.
2. E.Y. Choi, S.M. Lee, and D.H. Lee. Efficient RFID authentication protocol for ubiquitous computing environment. In *Proc. of SECUBIQ'05*, LNCS, 2005.
3. T. Dimitriou. A lightweight RFID protocol to protect against traceability and cloning attacks. In *Proc. of SECURECOMM'05*, 2005.
4. M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In *Proc. of CHES'04*, volume 3156 of *LNCS*, pages 357–370, 2004.
5. D. Henrici and P. Müller. Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. In *Proc. of PERSEC'04*, pages 149–153. IEEE Computer Society, 2004.
6. M. Jung, H. Fiedler, and R. Lerch. 8-bit microcontroller system with area efficient AES coprocessor for transponder applications. Ecrypt Workshop on RFID and Lightweight Crypto, 2005.
7. M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *RFID Privacy Workshop*, 2003.
8. C.M. Roberts. Radio frequency identification (RFID). *Computers and Security*, 25(1):18–26, 2006.
9. W. Sean and L. Thomas. Automatic identification and data collection technologies in the transportation industry: BarCode and RFID. Technical report, 2001.
10. Datasheet Helion Technology. High Performance MD5. Fast SHA-1. Fast SHA-256. hash core for ASIC, 2005.
11. S.A. Weis, S.E. Sarma, R.L. Rivest, and D.W. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *Security in Pervasive Comp.*, volume 2802 of *LNCS*, pages 201–212, 2004.
12. Kirk H.M. Wong, Patrick C.L. Hui, and Allan C.K. Chan. Cryptography and authentication on RFIDnext term passive tags for apparel products. *Computers in Industry*, 57(4):342–349, 2006.
13. J. Yang, J. Park, H. Lee, K. Ren, and K. Kim. Mutual authentication protocol for low-cost RFID. Ecrypt Workshop on RFID and Lightweight Crypto, 2005.
14. K. Yksel, J.P. Kaps, and B. Sunar. Universal hash functions for emerging ultra-low-power networks. In *Proc. of CNDS'04*, 2004.