LMAP: A Real Lightweight 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 are devices of very limited computational capabilities, where only 250-3K logic gates can be devoted to security-related tasks. Many proposals have recently appeared, but all of them are based on RFID tags using classical cryptographic primitives such as PRNGs, hash functions, block ciphers, etc. We believe this assumption to be fairly unrealistic, as classical cryptographic constructions lie well beyond the computational reach of these constrained systems. A new approach is necessary to tackle this problem, so we propose a real lightweight mutual authentication protocol for low-cost RFID tags that offers an adequate security level and can be implemented even in the most limited systems, as it only needs around 300 gates.

Keywords: RFID, Tag, Reader, Pseudonym, Privacy, Mutual Authentication.

1 Introduction

In spite of the recent beginning of the mass deployment of RFID systems their penetration is nowadays mainly limited by privacy concerns: products labeled with tags reveal sensitive information when queried by readers, and they do it indiscriminately. An issue closely linked to privacy is tracking, or the violation of location privacy. This happens because the answers coming from tags are usually predictable: in fact, the average low-cost tag always provides the same identifier, allowing a third party to easily establish a link between a given tag and its holder or owner. There are some other problems worth to mention: physical attacks, denial of service (DoS), counterfeiting, spoofing, eavesdropping, traffic analysis, etc.

The low cost demanded for RFID tags (0.05-0.1€) forces the lack of resources for performing true cryptographic operations. Typically, these systems can only store hundreds of bits and have 5K-10K logic gates, but only 250-3K can be devoted to security tasks. Despite these restrictions, since the work of Sarma et. all [16] in 2002, most of the proposed solutions [2, 3, 8, 15, 12, 23] are based on the use of hash functions. Although this apparently constitutes a good and secure solution, engineers face the non-trivial problem of implementing cryptographic

hash functions with only 250-3K 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) cannot be used [18]. In [24] we 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 as an open question.

Interestingly, there are solutions which exclusively use non-cryptographic primitives. The authors of [19] propose a set of extremely lightweight challenge-response authentication algorithms. These can be used for authenticating the tags, but they may be easily broken by a powerful adversary. In [9], Juels proposes a solution based on the use of pseudonyms, without using any hash function. The RFID tag stores a short list of pseudonyms: it rotates them, releasing a different one on each reader query. After a set of authentication sessions, the list of pseudonyms will need to be reused or updated through an out-of-band channel, which limits the practicality of this scheme.

Furthermore, there are solutions based on the concept of human-computer authentication [7, 10, 21]. The security of the most promising protocols of this kind (HB, HB$^+$) is related to the learning parity with noise problem (LNP), whose hardness over random instances, still remains as an open question.

2 Efficient-Lightweight Protocol

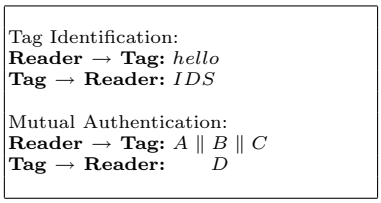
Like other authors, we think that the security of low-cost RFID tags can be improved with *minimalist cryptography* [9, 19]. A real lightweight mutual authentication protocol between RFID readers and tags, named LMAP, is proposed in this paper.

2.1 Suppositions of the Model

Our protocol is based on the use of pseudonyms, concretely on *index-pseudonyms* (IDSs). An *index-pseudonym* (96-bit length) is the index of a table (a row) where all the information about a tag is stored. Each tag is associated to a 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) must be updated, we need 480 bits of rewritable memory (EEPROM or FRAM) in total. A ROM memory to store the 96-bit static identification number (ID) is also required.

For the implementation of our protocol, costly operations such as random-number generation will be done by the reader. On the contrary, as tags are very limited devices that only have less than 1K logical gates for security functions, only simple operations are available: bitwise XOR ($\oplus$), bitwise OR ($\vee$), bitwise AND ($\wedge$), and addiction mod 2^m ($+$). We have not included the multiplication because is a very costly operation [13].

Due to the fact that most low-cost tags are passive, the communication must be initiated by readers. We also suppose that both the backward and the forward channel can be listened by an attacker. Finally, we assume that the communication channel between the reader and the database is secure.



$$A = IDS^{(n)}_{tag(i)} \oplus K1^{(n)}_{tag(i)} \oplus n1 \quad (1)$$

$$B = (IDS^{(n)}_{tag(i)} \vee K2^{(n)}_{tag(i)}) + n1 \quad (2)$$

$$C = IDS^{(n)}_{tag(i)} + K3^{(n)}_{tag(i)} + n2 \quad (3)$$

$$D = (IDS^{(n)}_{tag(i)} + ID_{tag(i)}) \oplus n1 \oplus n2 \quad (4)$$

Fig. 1. LMAP Protocol

2.2 The Protocol

We can split our protocol proposal in four main stages: tag identification, mutual authentication, index-pseudonym updating, and key updating. In this section, we outline how the protocol works, while in the next one a security and performance analysis is presented.

Tag Identification Before starting the protocol for mutual authentication, the reader has to identify the tag. The reader will send a *hello* message to the tag, which will answer by sending its current *index-pseudonym* (IDS). By means of this IDS, only an authorized reader will be able to access the tag secret key (K=$K1\|K2\|K3\|K4$), which is necessary to carry out the next authentication stage.

Mutual Authentication Our protocol basically consists on the exchange of two messages between the reader and the tag. The protocol works as follows:

1. Reader Authentication
   The reader will generate two random numbers $n1$ and $n2$. With $n1$ and the subkeys $K1$ and $K2$ the reader will generate the submessages A and B. With $n2$ and $K3$, it will generate the submessage C.
2. Tag Authentication
   With the submessages A and B, the tag will authenticate the reader and obtain $n1$. From the submessage C, the tag will obtain the random number $n2$. The random numbers $n1$ and $n2$ will be used in the *index-pseudonym* and key updating. Once these verifications are performed, the tag will generate the answer message D to authenticate and transmit its static identifier in a secure form.

We have analyzed the statistical properties of these four messages with three well-known suites of randomness tests, namely ENT [20], DIEHARD [14] and NIST [17]: we have generated a 300MB-file for every message and obtained the results shown in *Table 1*. Due to the huge amount of p-values generated by the NIST statistical battery, the report is not shown in this paper[1]. As we can see, it

[1] The whole report is available in `http://163.117.149.208/lmap/`

Table 1. Results obtained with ENT, DIEHARD and NIST over the messages sequence

	A	B	C	D
Entropy (bits/byte)	7.999999	7.999999	7.999999	7.999999
Compression Rate	0%	0%	0%	0%
χ^2 Statistic	250.98 (50%)	255.71 (50%)	244.46 (50%)	255.18 (50%)
Arithmetic Mean	127.5062	127.4977	127.4946	127.5030
Monte Carlo π Estimation	3.1413 (0.01%)	3.1417 (0.0%)	3.1417 (0.0%)	3.1413 (0.01%)
Serial Correlation Coefficient	-0.000040	0.000010	-0.000077	-0.000036
Diehard Battery (Overall p-value)	0.227765	0.775516	0.641906	0.410066
Nist Battery	$\surd$	$\surd$	$\surd$	$\surd$

points to ensure messages are not easily distinguishable from a random source, not even for the eavesdropper/cryptanalyst. We have put particular emphasis on the properties of the message D due to the fact that in it the tag sends its more valuable information: the static identification number - ID. As we can verify in *Equation 4*, the message D uses an xor of the two new random numbers ($n1$, $n2$) sent by the reader.

Index-Pseudonym and Key Updating After the reader and the tag have been authenticated mutually, the index-pseudonym and the key updating stage must be carried out. As tags are very computationally constrained devices, this task should only use efficient operations: bitwise XOR ($\oplus$), bitwise OR ($\vee$), bitwise AND ($\wedge$), and addiction mod 2^m ($+$). These operations have already been implemented in the tag for the normal running of the protocol, so its use will not imply an increase in the gate counting. Nevertheless, we should keep in mind the temporary requirements which limit the number of operations that can be performed. Taking all these considerations into account, the proposed equations for the index-pseudonym and the key updating are the following ones:

$$IDS_{tag(i)}^{(n+1)} = (IDS_{tag(i)}^{(n)} + (n2 \oplus K4_{tag(i)}^{(n)})) \oplus ID_{tag(i)} \tag{5}$$

$$K1_{tag(i)}^{(n+1)} = K1_{tag(i)}^{(n)} \oplus n2 \oplus (K3_{tag(i)}^{(n)} + ID_{tag(i)}) \tag{6}$$

$$K2_{tag(i)}^{(n+1)} = K2_{tag(i)}^{(n)} \oplus n2 \oplus (K4_{tag(i)}^{(n)} + ID_{tag(i)}) \tag{7}$$

$$K3_{tag(i)}^{(n+1)} = (K3_{tag(i)}^{(n)} \oplus n1) + (K1_{tag(i)}^{(n)} \oplus ID_{tag(i)}) \tag{8}$$

$$K4_{tag(i)}^{(n+1)} = (K4_{tag(i)}^{(n)} \oplus n1) + (K2_{tag(i)}^{(n)} \oplus ID_{tag(i)}) \tag{9}$$

The statistical properties of the four subkey updating sequences are good because in all of them an xor with a random number ($n1$ or $n2$) is done. Owing to the fact that in the index-pseudonym updating we have used a sum instead of a xor, we have analyzed this sequence in the same way we did with the message sequence (A, B, C, and D): we have generated a 300MB-file and obtained the results shown in *Table 2*. The whole report is also available in `http://163.117.149.208/lmap/`. From these results, there is no evidence to ensure that the IDS is different from a random variable.

Table 2. IDS Analysis

	IDS
Entropy (bits/byte)	7.999999
Compression Rate	0%
χ^2 Statistic	251.75 (50%)
Arithmetic Mean	127.4930
Monte Carlo π Estimation	3.1417 (0.0%)
Serial Correlation Coefficient	0.000078
Diehard Battery (Overall p-value)	0.619483
Nist Battery	$\surd$

3 Evaluation

3.1 Security Analysis

Once the proposed mutual authentication protocol has been presented, we will evaluate its security, studying the same properties that Yang analyzes in [23].

1. ***User Data Confidentiality***
   The tag ID must be kept secure to guarantee users privacy. The tag sends the message D ($D = (IDS^{(n)}_{tag(i)} + ID_{tag(i)}) \oplus n1 \oplus n2$) hiding the tag ID to a nearby eavesdropper equipped with an RFID reader.
2. ***Tag Anonymity***
   As the ID of the tag is static, we should send it, and all other interchanged messages, in seemingly random wraps (i.e. to an eavesdropper, random numbers are sent). As we have seen, the reader generates the message $A||B||C$. This message will serve to authenticate him, as well as to transmit, in a secure form the random numbers $n1$ and $n2$ to the tag. This two random numbers will be used to hide tag ID as well as to update the index-pseudonym and the associated key. 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 hypothesis is that 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 with 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 the IDS from it as an answer. 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 failed, the IDS will 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 does not changed, it needs to send more than 100 answers/second 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. Unfortunately, for the moment, there is no known solution for it (instead of spread spectrum).

3. ***Data Integrity***
   A part of the memory of the tag 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 (message $A \parallel B \parallel C$) and tag-to-reader authentication (message D).
5. ***Forward Security***
   Forward security is the property that guarantees that the security of messages sent today will be valid tomorrow [15]. Since the key updating is fulfilled after the mutual authentication, a future security compromise on an RFID tag will not reveal data previously transmitted.
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 ($n1$, $n2$), refreshed with 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 $A||B||C$ 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 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 (bitwise XOR ($\oplus$), bitwise OR ($\vee$), bitwise AND ($\wedge$), and addiction mod 2^m ($+$)) with random numbers ($n1$, $n2$). Therefore the simple copy of information of the tag by eavesdropping is not possible.
9. ***Data Recovery***
   The intercepting or blocking of messages is a DoS attack preventing 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 particularly focus on providing data recovery.

Table 3 shows a comparison of the security requirements of different proposals in the literature made by Yang [23]. We have added our proposal (LMAP) in the last column.

Table 3. Comparison between protocols

Protocol	HLS [22]	EHLS [22]	HBVI [8]	MAP [23]	**LMAP**
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

3.2 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 in the previous section, we will consider the same overheads (computation, storage, and communication) as in Yang [23].

1. ***Computation Overhead***
   Low-cost RFID tags are very limited devices, with only a small amount of memory and very constrained computationally (only 250-3K logic 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 4*, this problem is also presented in Yang's solution [23]. 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 all components are L-bit sized, that the RNG and the hash function are $h, h_k : \{0,1\}^* \rightarrow \{0,1\}^{\frac{1}{2}L}$ and $r \ \epsilon_U \ \{0,1\}^L$. Our protocol is based on L-bit *index-pseudonyms* ($IDSs$) and a 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 the tag (T) and the reader (R+B), requiring only four rounds. As we can see in *Table 4*, other protocols require at least one or two additional messages. 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 a reasonable number of rounds for mutual authentication in RFID environments.

Table 4. Computational loads and required memory

Protocol	Entity	HLS [22]	EHLS [22]	HBVI [8]	MAP [23]	**LMAP**
No. of Hash Operation	T	1	2	3	2	¬
	B	¬	n	3	2Nt	¬
No. of Keyed Hash Operation	R	¬	¬	¬	1	¬
	B	¬	¬	¬	1	¬
No. of RNG Operation	T	¬	1	¬	¬	¬
	R	¬	¬	¬	1	¬
	B	¬	¬	1	¬	¬
No. of Basic Operation[1]	T	¬	¬	¬	4	19
	R+B	¬	¬	¬	2(Nt+1)	21
No. of Encryption	B	¬	¬	¬	1	¬
No. of Decryption	R	¬	¬	¬	1	¬
Number of Authentication Steps		6	5	5	5	4
Required Memory Size[2]	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: ¬: Not require Nt: Number of tags L: Size of required memory
[1]Basic Operations: $\oplus$: Bitwise XOR $\vee$: Bitwise OR
$\wedge$: Bitwise AND $+$: Addiction mod 2^m

4 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 this message for authenticating the reader. Once it is done, the tag will send the message D to authenticate itself.

One of the first and most 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 at once. 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 all the message at the same time, 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 length (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 sequencer. Due to the fact that messages consists of three or more 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 make the following m-bit word length operations: bitwise XOR ($\oplus$), bitwise OR ($\vee$), bitwise AND ($\wedge$), and addiction mod 2^m ($+$). The ALU has two inputs, one of these values is stored in the memory and another is selected (c_1) between

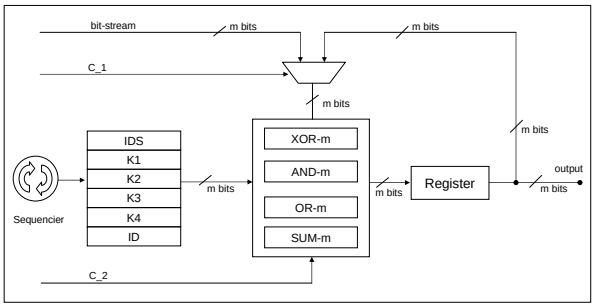


Fig. 2. Logic scheme

the bitstream and the value stored in the register_1. The control signal c_2 will select the operation that will be used in the ALU.

In the worse case of our protocol ($m = 8$ bits), we need 864 clock cycles for running the protocol (mutual authentication, pseudonym updating, and key updating). So, if we consider that the clock frequency is set to 100KHz, this means that the tag answers in 8,64 milliseconds. A tag can authenticate 115 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 that the word length (m). For the implementation of the adder circuit with carry a parallel architecture is proposed. Six logic gates are needed for each bit added in parallel[1]. Additionally, a 20% of logic gates are considered for control functions.

Table 5. Proposed architectures features

Word length		**8-bit**	**16-bit**	**32-bit**	**64-bit**	**96-bit**
Number of Gates	ALU	72	144	288	576	864
	Control	14	29	58	115	173
	Total	86	173	346	691	1037
Number of clock cycles[1]		864	432	216	108	72
Answers/second		115	231	462	925	1388

As we can see in *Table 5*, in the best case (m=8) the protocol only needs around 100 gates. In *Table 6*, 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 [18]. An efficient implementation of AES encryption has been recently published [11], which needs around 3400 logic-gates [5, 6]. Additionally, there is also a proposal of an implementation of a new universal hash function for ultra low-power cryptographic hardware applications.

[1] Add one bit with carry: $S = A \oplus [B \oplus C_{ENT}] \qquad C_{SAL} = BC_{ENT} + AC_{ENT} + AB$

Table 6. Core Comparison

Solutions	Implementation	Gate Count
Hash	Universal Hash Yksel [24]	1,7K gates
	MD5 Helion [18]	16K gates
	Fast SHA-1 Helion [18]	20K gates
	Fast SHA-256 Helion [18]	23K gates
AES Unit[2]	JungFL [5, 6]	3400 gates
	Feldhofer [4]	3595 gates
	Amphion CS5265 [1]	25000 gates

Although this solution only needs around 1.7K gates, a deeper security analysis of the hash function is needed and has not yet been accomplished.

Finally, although we have not implemented the circuit physically yet, 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.

5 A Naif Extension: $LMAP^+$

In those scenarios in which the intercepting or blocking of messages is considered an important problem, an extended version of the protocol is possible and quite straightforward. The tag will have an state associated in the database: synchronized or uncertainty. Furthermore, 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* ($n+1$, $n+2$, ... , $n+l$). The factor l is the number of times that a tag can remain in the uncertainty state. Moreover, each tag will need k additional bits of memory to store the Associated Data Base Entry like in [8]. As before, the reader will use the IDS to access all the information associated with the tag. The reader will store a potential *index-pseudonym* each time the tag's answer is blocked (uncertainty state). Once the tag and the reader have been mutually authenticated, the potential *index-pseudonyms* will be deleted (synchronized state). The storage of the potential *index-pseudonyms* will allow to easily recover from the lose or interception of messages. We can see an execution of the protocol in *Figure 3*.

6 Conclusions

Low-cost RFID tags are very limited devices, with only a small amount of memory, and very constrained computationally: there are incapable of performing true cryptographic operations. Surprisingly, most of the security solutions proposed are based on RFID tags using classical cryptographic primitives such as PRNGs, hash functions, block ciphers, etc. For this reason, we propose a real

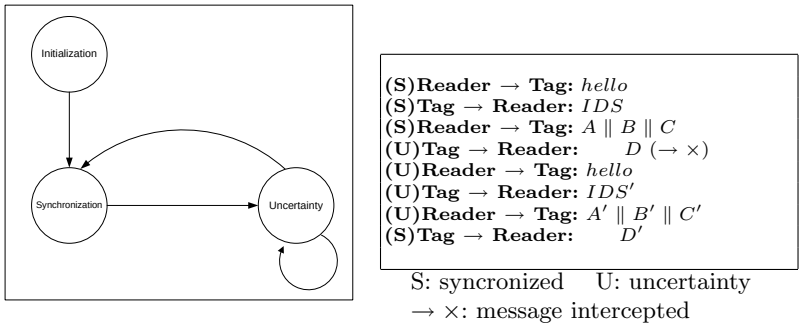


Fig. 3. Extended protocol: LMAP$^+$

lightweight mutual authentication protocol that could be implemented in low-cost tags (<1K 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 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) have been consider in this article and solved efficiently (around 1K gates are needed in the worst implementation, in our case $m = 96$ bits). As shown in *Table 4*, our protocol displays superior benefits to many of the solutions based on hash functions. So, not only we have been able to avoid 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 do not require exhaustive search in the back-end database. Furthermore, only two messages need to be exchanged in the identification 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. M. Feldhofer, K. Lemke, E. Oswald, F. Standaert, T. Wollinger, and J. Wolkerstorfer. State of the art in hardware architectures. In *Technical report, ECRYPT Network of Excellence in Cryptology*, 2005.
6. M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. AES implementation on a grain of sand. In *IEEE Proc. on Information Security*, volume 152, pages 13–20, 2005.

7. H. Gilbert, M. Robshaw, and H. Sibert. An active attack against HB^+ - A provably secure lightweight authentication protocol. Manuscript, 2005.
8. 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.
9. A. Juels. Minimalist cryptography for low-cost RFID tags. In *Proc. of SCN'04*, volume 3352 of *LNCS*, pages 149–164. Springer-Verlag, 2004.
10. A. Juels and S. Weis. Authenticating pervasive devices with human protocols. In *Proc. of CRYPTO'05*, volume 3126 of *LNCS*, pages 293–308. IACR, Springer-Verlag, 2005.
11. 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.
12. S.M. Lee, Y.J. Hwang, D.H. Lee, and J.I.L. Lim. Efficient authentication for low-cost RFID systems. In *Proc. of ICCSA'05*, volume 3480 of *LNCS*, pages 619–627. Springer-Verlag, 2005.
13. T. Lohmmann, M. Schneider, and C. Ruland. Analysis of power constraints for cryptographic algorithms in mid-cost RFID tags. In *Proc. of CARDIS'06*, volume 3928 of *LNCS*, pages 278–288, 2006.
14. G. Marsaglia and W.W. Tsang. Some difficult-to-pass tests of randomness. *Journal of Statistical Software*, Volume 7, Issue 3:37–51, 2002.
15. M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *RFID Privacy Workshop*, 2003.
16. Sanjay E. Sarma, Stephen A. Weis, and Daniel W. Engels. RFID Systems and Security and Privacy Implications. In *Proc. of CHES'02*, volume 2523, pages 454–470. LNCS, 2002.
17. C. Suresh, Charanjit J., J.R. Rao, and P. Rohatgi. A cautionary note regarding evaluation of AES candidates on smart-cards. In *Second Advanced Encryption Standard (AES) Candidate Conference*. http://csrc.nist.gov/encryption/aes/round1/conf2/aes2conf.htm, 1999.
18. Datasheet Helion Technology. MD5, SHA-1, SHA-256 hash core for Asic. http://www.heliontech.com, 2005.
19. I. Vajda and L. Buttyán. Lightweight authentication protocols for low-cost RFID tags. In *Proc. of UBICOMP'03*, 2003.
20. J. Walker. ENT Randomness Test. http://www.fourmilab.ch/ random/, 1998.
21. S. Weis. Security parallels between people and pervasive devices. In *Proc. of PERSEC'05*, pages 105–109. IEEE Computer Society Press, 2005.
22. 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 *Proc. of Security in Pervasive Comp.*, volume 2802 of *LNCS*, pages 201–212, 2004.
23. 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.
24. K. Yksel, J.P. Kaps, and B. Sunar. Universal hash functions for emerging ultra-low-power networks. In *Proc. of CNDS'04*, 2004.