EMAP: An Efficient 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. RFID tags are devices of very limited computational capabilities, which only have 250-3K logic gates that 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 very low-cost RFID tags. A new approach is necessary to tackle this problem, so we propose an extremely efficient lightweight mutual-authentication protocol that offers an adequate security level for certain applications and can be implemented even in the most limited low-cost RFID tags, as it only needs around 150 gates.

Keywords: Ubiquitous Computing, RFID, Tag, Reader, Privacy, Tracking, Pseudonym, Mutual-authentication

1 Introduction

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 their successful deployment. The most important security questions are privacy and tracking, but there are some others worth to mention, such as physical attacks, denial of service, 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 [9] in 2002, most of the proposed solutions [1, 2, 15] 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 between 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 [11]. In [16] 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.

The remainder of the paper is organized as follows. In Sect. 2, we prose an Efficient Mutual-Authentication Protocol ($EMAP$) for low-cost RFID tags. A security evaluation and performance analysis of this new protocol is presented in Sect. 3. In Sect. 4, the proposed architecture for implementing our protocol is explained in detail. Finally, concluding remarks appear in Sect. 5.

2 Efficient-Lightweight Protocol

Like other authors, we think that the security of low-cost RFID tags can be improved with *minimalist cryptography* [5, 12]. Following this direction, an extremely efficient lightweight mutual-authentication protocol, named EMAP, 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 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 updated, we need 480 bits of rewritable memory (EEPROM or FRAM) in total. A ROM memory to store the 96-bit static tag identification number (ID) is also required.

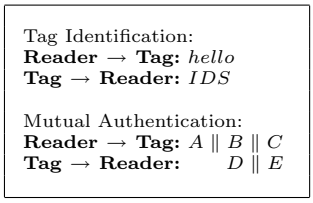
Costly operations such as random number generation will be done by readers. On the contrary, as tags are very limited devices that only have less than 1K logic gates for security functions, only simple operations are available: bitwise xor ($\oplus$), bitwise and ($\wedge$), and bitwise or ($\vee$). Multiplication have not been included because is a very costly operation [6].

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.

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 should identify the tag. The reader will send a *hello* message to the tag, which answers by sending its current *index-pseudonym* (IDS). By means of this IDS, the reader will be able to access to the secret key of the tag ($K = K1 \parallel K2 \parallel K3 \parallel K4$), which is necessary to carry out the next authentication stage.



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

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

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

$$D = (IDS^{(n)}_{tag(i)} \wedge K4^{(n)}_{tag(i)}) \oplus n2 \quad (4)$$

$$E = (IDS^{(n)}_{tag(i)} \wedge n1 \vee n2) \oplus ID_{tag(i)} \bigoplus_{I=1}^{4} KI^{(n)}_{tag(i)} \quad (5)$$

Fig. 1. EMAP Protocol

Mutual Authentication Our protocol consists in the exchange of two messages between the reader and the tag. An execution of the protocol is showed in *Figure 1.* The $\bigoplus_{I=1}^{N}$ operation represents an N-elements addiction with the bitwise xor operator $(K1 \oplus K2 \oplus ... \oplus KN)$.

- 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.
- 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 $n2$, that will be used in the index-pseudonym and key updating. Once these verifications are performed, the tag will generate the answer message. This message will be composed of two parts D and E. The submessage D will allow to authenticate the tag and by means of E its static identifier will be transmitted in a secure form.

We have analyzed the statistical properties of these five submessages with three well-known suites of randomness tests, namely ENT [13], DIEHARD [7] and NIST [10]: we have generated a 300MB-file for every message. Due to extension restrictions the reports are not shown in the paper.[1] The results point to ensure submessages are not easily distinguishable from a random source, not even for the eavesdropper/cryptanalyst. As we can verify in *Equation 5*, submessage E uses more operations than the rest. We have put particular emphasis on the properties of submessage E due to the fact that in it the tag sends its more valuable information: the static identification number (ID).

Pseudonym Index Updating Once the tag and the reader have mutually authenticated, each one has to update the index-pseudonym.

$$IDS^{(n+1)}_{tag(i)} = IDS^{(n)}_{tag(i)} \oplus n2 \oplus K1^{(n)}_{tag(i)} \quad (6)$$

[1] The whole reports are available in `http://163.117.149.208/emap/`

The statistical properties of this sequence is good owing to the use of an xor with a random number ($n2$). In connection with the speed requirements, we have only used three basic operations (bitwise xor).

Key Updating The key updating will be carry out, as will the index-pseudonym updating, after the mutual authentication. As tags are very computationally constrained devices, this task should be made only by using efficient operations: bitwise xor ($\oplus$), bitwise and ($\wedge$), and bitwise or ($\vee$). These operations have already been implemented in the tag for the normal protocol running, so its use will not imply an increase in the gate counting. In order to improve the security of the key updating algorithm, a parity function will be used.[2] Nevertheless, the speed requirements of tags should be kept in mind; a tag must be able to answer 50 times/sec (see Sect. 4). These speed requirements put a limit on the number of 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:

$$K1_{tag(i)}^{(n+1)} = K1_{tag(i)}^{(n)} \oplus n2 \oplus (ID_{tag(i)}(1:48)||F_p(K4_{tag(i)}^{(n)})||F_p(K3_{tag(i)}^{(n)})) \quad (7)$$

$$K2_{tag(i)}^{(n+1)} = K2_{tag(i)}^{(n)} \oplus n2 \oplus (F_p(K1_{tag(i)}^{(n)})||F_p(K4_{tag(i)}^{(n)})||ID_{tag(i)}(49:96)) \quad (8)$$

$$K3_{tag(i)}^{(n+1)} = K3_{tag(i)}^{(n)} \oplus n1 \oplus (ID_{tag(i)}(1:48)||F_p(K4_{tag(i)}^{(n)})||F_p(K2_{tag(i)}^{(n)})) \quad (9)$$

$$K4_{tag(i)}^{(n+1)} = K4_{tag(i)}^{(n)} \oplus n1 \oplus (F_p(K3_{tag(i)}^{(n)})||F_p(K1_{tag(i)}^{(n)})||ID_{tag(i)}(49:96)) \quad (10)$$

The statistical properties of these four sequences are good because of in each sequence there is an xor with a random number ($n1$ or $n2$). According to the speed requirements, for the worst case, which is obtained on the 8 bit architecture, a tag can authenticate 89 times per second, so we are able to successfully fulfill the speed requirements in all cases (see Sect. 4).

3 Evaluation

3.1 Security Analysis

Once we have presented the proposed mutual-authentication protocol, we will evaluate its security, studying the same properties that Yang analyzes in [15].

1. ***User Data Confidentiality***
   The tag ID must be kept secure to guarantee user privacy. The tag sends in the message E ($E = (IDS_{tag(i)}^{(n)} \wedge n1 \vee n2) \oplus ID_{tag(i)} \bigoplus_{I=1}^{4} KI_{tag(i)}^{(n)}$) hiding the tag ID to a nearby eavesdropper equipped with an RFID reader.

[2] Parity function ($F_p(X)$): The 96-bit number X is divided in twenty four 4-bit blocks. For each block we obtain a parity bit, getting 24 parity bits. See Sect. 4 for more details.

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, readers generate 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 ($n1$, $n2$) will be used to hide the 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 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 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 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 failed, 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 50 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. 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* and the key 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 $D \parallel E$).
5. ***Forward Security***
   Forward security is the property that privacy of messages sent today will be valid tomorrow [8]. Since 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.

Table 1. Comparison Between Protocols

Protocol	HLS [14]	EHLS [14]	HBVI [4]	MAP [15]	**EMAP**
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 × Not Satisfied

7. ***Replay Attack Prevention***
   An eavesdropper could store all the messages interchanged between the reader and the tag (different protocol runs). Then, he can 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 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 and ($\wedge$), and bitwise or ($\vee$)) 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 of messages is a denial-of-service 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.
   In those scenarios in which this problem is considered important, an extended version of the protocol is possible and quite straightforward. 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* ($n + 1$, ... , $n + l$). Moreover, each tag will need k bits additionally of ROM memory to store the Associated Data Base Entry like in [4]. As before, the reader will use the IDS to access all the information associated with the tag. The reader will store a potential IDS each time the answer of the tag is blocked (uncertainty state). Once the tag and the reader have been authenticated mutually, the potential IDS could be deleted (synchronized state). The storage of the potential IDS will allow to easily recover from the lose or interception of messages.

Table 1 shows a comparison of the security requirements made by Yang [15], as met by different proposals in the literature. We have added our proposal (EMAP) in the last column.

Table 2. Computational Loads and Required Memory

Protocol	Entity	HLS [14]	EHLS [14]	HBVI [4]	MAP [15]	**EMAP**
No. of Hash Operation	T	1	2	3	2	¬
	B	¬	Nt	3	2Nt	¬
No. of Keyed Hash Operation	R	¬	¬	¬	1	¬
	B	¬	¬	¬	1	¬
No. of RGN Operation	T	¬	1	¬	¬	¬
	R	¬	¬	¬	1	¬
	B	¬	¬	1	¬	¬
No. of Basic Operation[1,2]	T	¬	¬	¬	4	22
	$R+B$	¬	¬	¬	2(Nt+1)	25
No. of Encryption	B	¬	¬	¬	1	¬
No. of Decryption	R	¬	¬	¬	1	¬
Number of Authentication Steps		6	5	5	5	4
Required Memory Size	T	$1\frac{1}{2}\ L$	$1L$	$3L$	$2\frac{1}{2}\ L4$	$6L$
	$R+B$	$2\frac{1}{2}L$	$1\frac{1}{2}L$	$9L$	$9\frac{1}{2}L$	$6L$

†† Notation: ¬ : Not require Nt: Number of Tags L: Size of Required Memory
[1]Basic Operations: Bitwise xor ($\oplus$), Bitwise and ($\wedge$), and Bitwise or ($\vee$)
[2]Parity function has been included as a basic operation

3.2 Performance Analysis

Before evaluate the security of the protocol a performance analysis will be presented (see *Table 2*), considering the following overheads (computation, storage, and communication) as in Yang [15].

1. ***Computation Overhead***
   Low-cost RFID tags are very limited devices, with only a small amounts of memory, and very constrained computationally (<1K logic gates 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 it happens in Yang's solution [15]. Our proposal (EMAP) have completely solved this problem by using an *index-pseudonym*.
2. ***Storage Overhead***
   As Yang does, we assume that all components are L-bits sized, that the RNG and the hash function are $h, h_k : \{0,1\}^* \rightarrow \{0,1\}^{\frac{1}{2}L}$ and $r\ \epsilon_U\ \{0,1\}^L$. As we see in Sect. 2.1 each tag has to store an L-bit *index-pseudonym* (IDS) and an associate key (K) of four L-bit components. Moreover, the tag has to store an unique L-bit identification number (ID). The reader has to store the same information, so it requires a memory of $6L$ bits.
3. ***Communication Overhead***
   As we can see in *Table 2*, according the number of interchanged messages to accomplish mutual authentication tag-reader, our protocol is the most efficient. As 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.

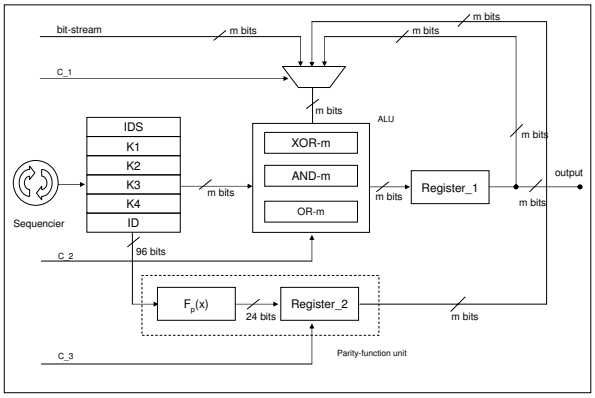


Fig. 2. Logic Scheme

4 Implementation

In this section, we will explain in detail the proposed architecture for implementing our protocol. The proposed architecture is independent of the word length used. We have analyzed the features of four different word length (m = 8, 24, 48, 96 bits). 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 static identification number (ID). The access to the memory is controlled by a sequencier. Due to the fact that messages are build up 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 operations: bitwise xor ($\oplus$), bitwise and ($\wedge$), and bitwise or ($\vee$). The ALU has two inputs, one of these values stored in the memory and another which is selected (c_1) between one of these three values: the bitstream, the value stored in the register_1 and the the result of the parity-function unit. The control signal c_2 will select the operation that will be used in the ALU. At the bottom of the figure we can see the parity-function unit. This unit will be used each time the key is updated, in particular twice with each part of K ($K1 \parallel K2 \parallel K3 \parallel K4$). In order to carry out the temporal requirements we have decided to implement this function in just one block. This function has an input length of 96 bits and a 24-bits output. The input is divided in blocks of 4 bits, which are processed to obtain an output bit. For example, for the first four bits, x1 is xored with x2, x3 is xored with x4, and finally the corresponding outputs are xored ($(x1 \oplus x2) \oplus (x3 \oplus x4)$). So for 96 bits of input, we need 72 logical gates (24x3) for implementing the parity function. The output of this function is stored in the register_2 of dimension m (m = 24, 48, 96 bits). The control signal c_3 will select when a 24-bit shift has to be done in the register.

It is a common assumption that a maximum of 50 tags can be authenticated per second. As in [3], 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 2000 clock cycles to answer a reader. In the worse case of our protocol ($m = 8$ bits), we need 1120 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 [3], this means that the tag answers in 11.2 milliseconds. A tag can authenticate 89 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 the proposed protocol. The functions bitwise xor ($\oplus$), bitwise and ($\wedge$), and bitwise or ($\vee$) will be implemented with the same number of logic gates like the word length (m). As seen above, 72 logical gates will be needed for implementing the parity function. Additionally, an extra 30% of logic gates are added up for control functions. In the worst case ($m = 96$ bits) the protocol only needs around 500 gates. Moreover, 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.

Table 3. Features

Word's length		**8-bits**	**24-bits**	**48-bits**	**96-bits**
Number of. Gates	ALU	24	72	144	288
	Parity Function	72	72	72	72
	Control	29	43	65	108
	Total	**125**	**187**	**281**	**468**
Number of Clock Cycles		1120	416	240	152
Answer/sec		**89**	**240**	**417**	**658**

5 Conclusions

RFIDs tags are devices limited to hundreds of bits of store, and with roughly 250-3K gates devoted to security-related tasks. Cryptographic primitives such as PRNGs, block ciphers, and hash functions lie well beyond the computational reach of very low cost RFID tags, but until now, most of the security solutions for RFID are based on those. A new approach must be taken to tackle the problem, at least for low-cost RFID tags. For this reason, we propose an extremely efficient lightweight mutual-authentication protocol ($EMAP$) 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

(less than 500 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 the man-in-the-middle attack, replay attack, 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. E.Y. Choi, S.M. Lee, and D.H. Lee. Efficient RFID authentication protocol for ubiquitous computing environment. In *Proc. of SECUBIQ'05*, 2005.
2. T. Dimitriou. A lightweight RFID protocol to protect against traceability and cloning attacks. In *Proc. of SECURECOMM'05*, 2005.
3. 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.
4. 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.
5. A. Juels. Minimalist cryptography for low-cost RFID tags. In *Proc. of SCN'04*, volume 3352 of *LNCS*, pages 149–164. Springer-Verlag, 2004.
6. 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.
7. G. Marsaglia and W.W. Tsang. Some difficult-to-pass tests of randomness. *Journal of Statistical Software*, Volume 7, Issue 3:37–51, 2002.
8. M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *RFID Privacy Workshop*, 2003.
9. S.E. Sarma, S.A. Weis, and D.W. Engels. RFID Systems and Security and Privacy Implications. In *Proc. of CHES'02*, volume 2523, pages 454–470. LNCS, 2002.
10. 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.
11. Datasheet Helion Technology. MD5, SHA-1, SHA-256 hash core for Asic. http://www.heliontech.com, 2005.
12. I. Vajda and L. Buttyán. Lightweight authentication protocols for low-cost RFID tags. In *Proc. of UBICOMP'03*, 2003.
13. J. Walker. ENT Randomness Test. http://www.fourmilab.ch/ random/, 1998.
14. 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.
15. 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.
16. K. Yksel, J.P. Kaps, and B. Sunar. Universal hash functions for emerging ultra-low-power networks. In *Proc. of CNDS'04*, 2004.