AUTO–ID LABS

Enhanced RFID Mutual Authentication Scheme based on Synchronized Secret Information

Sangshin Lee, Hyunrok Lee, Tomoyuki Asano, and Kwangjo Kim

Auto-ID Labs White Paper WP-HARDWARE-032

Sangshin Lee
Researcher,
Daewoo Electronics

Hyunrok Lee
PhD Student,
Information and
Communication University

Tomoyuki Asano
Senior Researcher,
Sony Corporation

Kwangjo Kim
Professor,
Information and
Communication University

Contact: tank@icu.ac.kr

CAIS Lab (R504), Information and Communication University (ICU)
119 Munjiro, Yuseong-gu, Daejeon, 305-732, Republic of Korea

Internet: http://caislab.icu.ac.kr/

Hardware

Abstract

Radio Frequency IDentification (RFID) is an automatic identification system, relying on storing and remotely retrieving data about objects we want to manage using devices called “RFID tag”. Even though RFID system is widely useful for industrial and individual applications, RFID tag has a serious privacy problem, i.e., traceability. To protect users from tracing, we propose an RFID mutual authentication scheme which utilizes a hash function and synchronized secret information like others. To the best of our knowledge, our scheme offers the most enhanced security feature in RFID mutual authentication scheme with respect to user privacy including resistance against tag cloning allowing an additional hash operation.

Key words: *RFID, security, authentication, privacy, untraceability, cloning*

1. Introduction

Radio Frequency IDentification (RFID) is an automatic identification system, relying on storing and remotely retrieving data about objects we want to manage using devices called “RFID tag”. The RFID system is more useful for various purposes than optical barcode technology since the RFID system can identify lots of tags quickly through a RF with neither physical nor visual contact. The RFID system can be used in lots of industries such as supply chain management, inventory, storage, etc. and give facilities for individuals with a ubiquitous computing environment.

However, RFID system can have security problems inherently if the tag offer no access-control and tamper-resistance mechanisms. It can induce information leakage problems of companies and privacy problems of individuals since the RFID tag emits their data to everyone including adversaries. For example, a dishonest company may try to collect information of competing company about physical distribution. By utilizing responses from a tag, an adversary may try to get knowledge of products which an individual user carries or trace a user. In addition, we must consider an attack that an adversary earns unfair profits by responding a reader’s query with forged information. These vulnerabilities make people reluctant to use RFID technology [3, 14].

Even though there are many cryptographic primitives for similar vulnerabilities of other system, they can not be applied to the RFID system because of light-weight calculation power of a low-cost tag. Consequently, new security protocols with less calculation in the tag are required.

Therefore, we provide a protocol which can be realized in a low-cost tag. To protect users from tracing, we propose an RFID mutual authentication scheme which utilizes a hash function and synchronized secret information like others [16, 12, 8, 6, 11, 10]. We will show our scheme offers the most enhanced security feature in RFID mutual authentication scheme with respect to user privacy including resistance against tag cloning allowing one more hash operation.

1.1 Notations

We use the notations for entities and operations as summarized in Table 1 to simplify description throughout the paper.

T	RF tag, or transponder.
R	RF tag reader, or transceiver.
B	Back-end server, which has a database.

D	A database of *B*.
A	An adversary.
h()	One-way hash function.
PRNG	PseudoRandom Number Generator.
⊕	Exclusive-or (XOR) function.

Table 1. Notations

1.2 RFID System

The RFID system has three main components: *T*, *R*, and *B* [17]. *T* carries object identifying data. *T* is attached to all objects in an RFID system. *T* is typically composed of a microchip for storage and computation, and a coupling element, such as an antenna coil for communication. *T* may also contain a contact pad. Tag memory may be read-only, write-once read-many or fully rewritable. *R* not only queries *T* for its data, but also updates the contents of *T* through an RF interface. To provide additional functionality, R may contain internal storage, processing power or connections to *B*. Computation, such as cryptographic calculations, may be performed by *R* on behalf of *T*.

B stores records associated with *T*. *R* may use contents of *T* as check information to find ID of *T*. *B* may associate product information, tracking logs or key management information with a particular *T*. An independent *B* may be built by anyone with access to tag contents. This allows *A* along the supply chain to build his own applications.

1.3 Previous Work

There exist hardware-based schemes to protect user privacy such that kill command feature [1], blocker-tag [9], and Faraday case. Kill command feature is originally suggested by Auto-ID center [2]. Each *T* has a password. When *R* orders kill command to *T* with its password, *T* stops its operation permanently. This feature makes possible perfect security, but we cannot reuse *T*. *T* must be killed when *T* goes to an insecure area. After that, *T* cannot be operated any more even if *T* returns to a secure area. The blocker-tag jams all *T* when tree-walking singulation is processed. The blocker-tag also disables legitimate user who wants to collect information from *T*. The Faraday case prevents *T* from hearing the request by enclosing *T*. This method is only suitable for limited application that we can enclose products using a Faraday case [5].

Several papers suggested schemes relying on the concept of universal reencryption

[7], that re-encryptions of a message *m* are performed neither requiring nor yielding knowledge of the public key under which m has been encrypted initially. The protocol of Golle *et al*. [7] proposed the concept of universal reencryption and applied the concept to the RFID system. Saito *et al*. [13] pointed out an attack against the protocol of Golle *et al*. [7] and subsequently suggested two protocols based on the Golle *et al*.'s protocol. The first protocol is an improvement of [7] where the operations carried out by T are modified. The difference between [7] and the second protocol of Saito *et al*. is that the re-encryptions are carried out by *T* itself and no longer by *R*. All schemes based on universal reencryption are nevertheless weak against eavesdropping. Previous re-encrypted data is the output of *T* of the next session, so an eavesdropper can link each session and trace *T*.

Some schemes use hash functions to identify *T*. Weis *et al*. [16] suggested a protocol that *T* sends *h(ID||r)* and including a random number *r*, whenever *R* wants to know ID of *T* . This scheme has vulnerability that *A* can trace the previous outputs of *T* if *A* tampers with *T* [4]. Ohkubo *et al*.'s protocol [12] is a unique protocol which prevents traces by this time. In this protocol, the *i-th T* sends $G(H_{k-1}(s^1_i))$ for the *k-th* response, where *G* and *H* are different hash functions and s1i is an initial value of the *i-th T* . To find ID of *T*, *B* must search all the hash chains of each *T*, so this protocol is inefficient to be installed in a system which is not small.

There are mutual authentication schemes between *B* and *T*. Henrici *et al*. [8] suggested a protocol in which *B* and each *T* share ID and update ID whenever identification is successfully done. This scheme has a database desynchronization problem [4]. If ID of *T* is desynchronized, *T* can be easily traced because one of emitted values of *T* is always identical. Dimitriou's protocol [6] also tries to do mutual authentication based on the use of secrets which are shared between *T* and *B*, and refreshed to avoid tag tracing. This scheme also has a database desynchronization problem. *A* can desynchronize ID between *T* and *D* by doing a man-in-the-middle attack. For example, *A* can deliver all data properly until *B* identifies *T* , but *A* doesn't transfer the last data from *B* to *T* which is used to update ID in *T* . This attack desynchronizes ID between *B* and *T* permanently. Molnar *et al*.'s scheme [11] doesn't have a desynchronization problem by fixing ID, but can be cryptographically weak against tampering with *T* since ID is fixed as [16]. Lee *et al*. [10] proposed a scheme to solve the desynchronization problem by maintaining a previous ID in *D*. However, in this scheme, *A* who queries *T* actively without updating ID can trace *T* because the hashed ID is continually identical. Moreover, *A* can trace the previous event of *T* because ID is updated by XORing the previous ID with a random number emitted through RF.

1.4 Organization

This paper is organized as follows: In Section 2, we describe RFID security problems

and security requirements. In Section 3, we introduce our scheme. In Section 4, we describe security analysis of our protocol. We compare our scheme with other schemes about security and efficiency in Section 5. In the final section, we provide a summary of our work.

2. RFID Security Problems

2.1 Security Problems

Privacy and cloning of *T* must be solved for proliferation of RFID technology. Because everyone can query to a low-cost tag (which doesn't have an access control function, *e.g.,* Class I tag [15]) without recognition of the tag holder, privacy must be considered.

One of privacy problems is the information leakage on user's belongings. We don't want that our personal things are known to others. For example, exposure of expensive products can make a tag holder be a victim of a robber. A personal medicine known to another throws the user into confusion. Even though the information leakage problem is significant, it's easy to solve. It can be solved just by using the anonymous ID's that *B* only can match with the real product codes [5].

Another problem about the user privacy is a user tracing problem. By tracing *T*, *A* can chase and identify the user. If *A* installs a vast amount of *R*'s at a wide area, each individual person's location privacy is violated by *A*. The user tracing problem is hard to solve, because we must update every response of *T* in order to evade a pursuer while a legitimate user can identify *T* without any inconvenience. Moreover, this job must be performed by *T* with small computational power.

Tag cloning also must not be ignored. *A* may try to clone a specific *T* to gain illegal benefit. For instance, if a protocol which is vulnerable to a replay attack is used, *A* can disguise an expensive product as cheap one by saving a response from *T* attached on cheap one and emitting the response while checking out.

2.2 Security Requirements

The most important security requirement for user privacy is *untraceability* [4]. Untraceability is the property that *A* can not trace *T* by using interactions with *T*. This concept includes ID *anonymity*, which is satisfied when *A* can not know a real product ID of *T* and guarantees to prevent the leakage of information of user belongings.

Even thought *A* doesn't know ID of *T*, *A* can trace *T* if *A* can find specific patterns of outputs of *T*, *e.g.,* a value increased by one for every response in [8].

For perfect untraceability, protocols must satisfy *indistinguishability* [12] and *forward security* [12] (or *forward untraceability* [4]). Indistinguishability means that values emitted by *T* must not be discriminated from the other *T*. For forward security, *A* cannot trace the data back through previous events in which *T* was involved even if *A* acquires the secret data stored in *T*.

Anti-cloning is an additional security requirement. This property means *A* cannot clone *T* without tampering with *T*. When *A* tampers with *T*, *A* has same information as *T* itself, and then we cannot prevent tag cloning. However, there are many ways to cloning without tampering with *T*, *e.g.,* the replay attack. Therefore, anti-cloning can be one of security requirements.

3. Our Protocol

3.1 Assumption and Initial Setup

B and *T* can operate the XOR calculation and a common one-way hash function, $h : \{0, 1\}^* \rightarrow \{0, 1\}^l$. *R* has a *PRNG* and a variable *s* whose length is *l*-bit. The role of *s* is to save pseudorandom number which is used in order to detect a replay attack. *T* also has a *PRNG*, which needs not be the same as one of *R*. The variable *r* whose length is *l*-bit is saved in a volatile memory of *T*. *r* is transmitted through RF and is tested for mutual authentication of components, *i.e.*, *B* and *T* . Because *r* is saved in a volatile memory, the contents of *r* are automatically deleted at the end of the authentication process. The variable *k* whose length is *l*-bit is saved in non-volatile memory of *T*. *k* is used in order to identify ID of *T* , so *k* must be different among all *T*'s all the time. The initial value of *k* of each *T* is assigned by precalculation to guarantee each *k* of *T* to be always different. Let *m* be the number of T's in a system, and let *n* be the maximum number of authentication times of each *T*. We construct a hash chain of secret information as follows: The hash chain starts from a secret seed *t*, the second one k_2 is $h(t)$, and the other *x-th* element k_x is $h(k_{x-1})$ where $3 \leq x \leq mn$. We must select *t* which makes a hash chain longer than *mn*. The initial value of *k* of each *T* is selected such that each one is far apart at least *n* in the hash chain. These initial values are saved in the each memory of *T* and maintained by *B* with the corresponding ID of *T*. *B* and *T* update *k* to the next value of the hash chain synchronouslly when the authentication process is successfully done. *D* of *B* has fields *IDR*, *K*, and K_{last}, which save the ID, the current *k*, the preceding *k* (the previous secret information which is replaced by the current *k*), respectively. Initially, *IDR* and

K are set up with ID and initial k of each T, respectively, and all values of the field K_{last} are null. The role of K_{last} is to prevent desynchronization.

3.1 Authentication process

The authentication protocol proceeds as follows:

1. R generates and saves a new pseudorandom number s by utilizing $PRNG$, and sends s to T.
2. T also generates a new pseudorandom number r_1 and sends r_1 to R. After that, T generates r_2, *i.e.*, $h(r_1 \oplus k \oplus s)$, where s was sent by R, and sends r_2 to R.
3. R delivers responses of T with the saved value s to B, *i.e.*, s, r_1, and r_2.
4. In order to find ID of T, B searches k' from the fields K and K_{last} which satisfies $h(r_1 \oplus k' \oplus s) = r_2$. – Eq(1) where r_1, r_2, and s are values sent by T. If only one k' satisfies the equality, then we can know ID corresponding to k' in D is ID of T because the above equality holds if k' and k is identical. If more than two values satisfing the equality are found because of hash collision although the probability of this case is roughly $2^{-(l-1)}m$, B informs the failure of searching ID of T to R, and orders R to query again in order to restart the process from the first step.
5. B updates information of T. If k' is found in the field K of a record, k' is copied to the field K_{last} of the record and the field K of the record is set to $h(k')$. If k' is found in the field K_{last}, we do not update D (because this situation means B already updated D at the previous authentication process but T didn't).

6. From s, r_2, and k', which are received values from T and the value found by testing whether $h(r_1 \oplus k' \oplus s) = r_2$, B calculates r_3', *i.e.*, $h(r_2 \oplus k' \oplus s)$, and sends r_3'to R. R transfers r_3' to T in order to inform the update.
7. In order to test the correctness of the value R sends, T checks whether $r_3 = r_3$'. – Eq(2) If true, it updates k to $h(k)$.

We illustrate our protocol in the Fig .1.

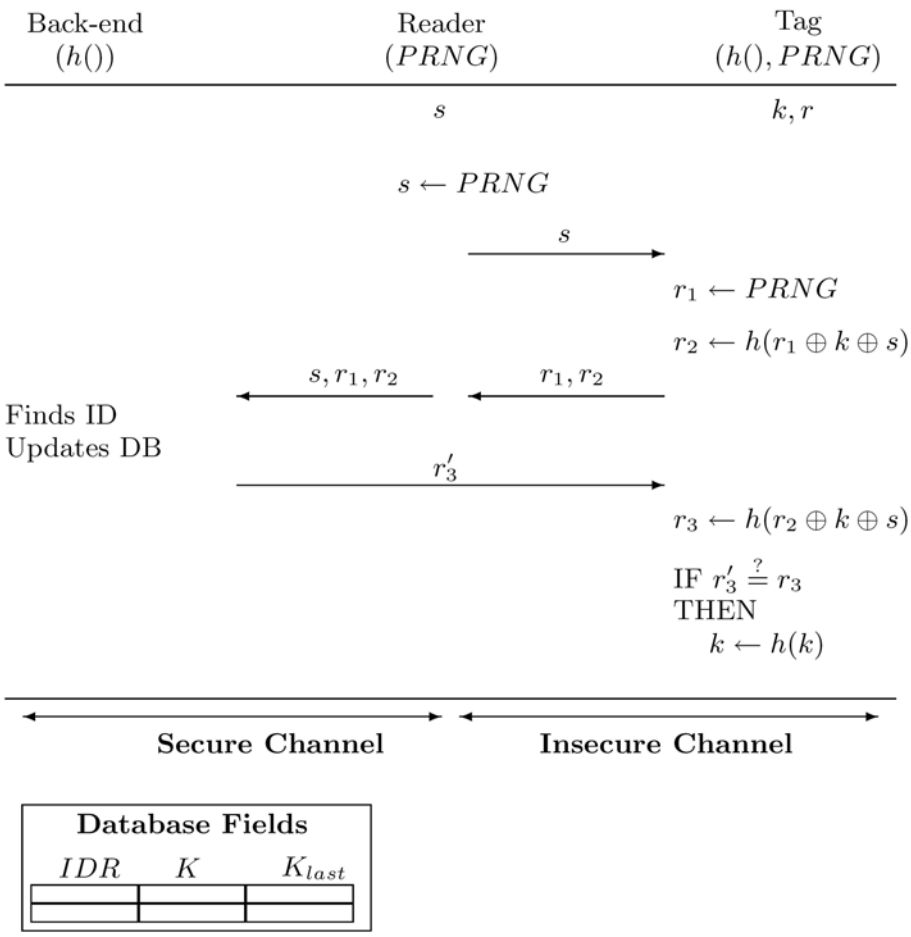

Fig.1. Our Protocol

4. Security Analysis

In this section, we analyze the security of our scheme. We show that our scheme is untraceable against all imaginable attacks except tampering with *T* in the random oracle model. Moreover, we describe that an attack based on tampering with *T* also limited by updating secure information. The followings are security analysis of our scheme against each attack.

Attack based on eavesdropping. *A* can collect *s*, r_1, r_2, and r_3 during one authentication process by eavesdropping. Because s and r_1 are random values, they are useless to trace *T*. Moreover, if *h()* is a random oracle, *A* who doesn't get knowledge of *k* in *T* cannot distinguish r_2 and r_3 from a random value. Since the inputs of *h()* to generate r_2 and r_3 are random because of *s*, the probability that the inputs are identical among each session is negligible. Because the random oracle is

assumed to be a function with the property that if a value in its domain is not queried before then the corresponding function value is a random value, r_2 and r_3 look like random value to *A*. Therefore, *A* cannot trace *T* by eavesdropping.

Cloning. *A* may try to do a replay attack by eavesdropping legitimate interactions, but *A* cannot success cloning by a replay attack because s is different for each session. Moreover, because *A* doesn't know *k* of *T*, *A* cannot generate r_2 from r_1.

Attack based on controlling *s*. *A* may try to make responses of *T* have some traceable patterns by controlling *s* as the role of *R*. If *A* can guess r_1 before *A* sends *s*, *A* can make responses of *T* be always identical by sending the value r_1 as *s* repeatedly. In that case, r_2 is always $h(r_1 \_k \_r_1)$, *i.e.,* $h(k)$. Therefore, *A* can trace *T*. However, because r_1 is a pseudorandom number, *A* cannot predict r_1. Therefore, *A* cannot trace *T* by means of controlling *s*.

Attack based on updating *k* in *D*. *A* may try to do a man-in-the-middle attack in order to desynchronize *k* between *D* and *T*. We assume *T* is apart from *R* and *A* can relay the data between these two components. When *R* starts the authentication process, *A* transfers *s* from *R* to *T* and then transfers r_1 and r_2 to *R*. Then, *B* updates *k* of *T* in *D*. After that, *B* sends r'_3 though *R* in order to update *k* in *T*. At this time, if *A* doesn't deliver r'_3 to *T*, *k* becomes different between *D* and *T*. If *B* only uses the latest *k* to find *ID*, this situation makes *T* useless. However, our scheme saves another *k* which was replaced by the latest *k* in the field K_{last}, and *B* can identify *T* in this situation. Because *B* doesn't update *k* when *T* is identified using field K_{last}, two fields *K* and K_{last} are sufficient to defend the system from a desynchronization attack.

Attack based on updating *k* in *T*. *A* may try to do a desynchronizing attack by updating secret information of *T* while *B* is ignorant of this situation. However, without a help of *B*, *A* cannot know r_3 because *A* doesn't know current *k* of *T* which is required in order to calculate r_3 from r_2. Therefore, *A* cannot success this attack.

Attack based on tampering. We can image an attack that after *A* obtains *k* by tampering with *T* and has eavesdropped plenty of interactions between *B* and various *T* 's, *A* tries to distinguish the response of *T* from others. *A* tests Eq.(1) by using the tampered *k* and collected interactions. If an interaction passes the test, *A* can guess the tampered *T* took part in that interaction. However, this forward trace is impossible over successive authentication. Even though *A* knows the current *k*, *A* cannot know previous *k* of *T* because *k* is updated by a oneway function $h()$. Therefore, the forward trace is limited to a short period and is impossible whenever the successive authentication process is done.

5. Comparison

Table 2 shows security and efficiency comparisons of our protocol with other mutual authentication schemes. The notation X means a scheme doesn't satisfy a given security requirement. The notation 4 means *A* cannot trace *T* over successive authentication but can trace *T* within successive authentication. The notation O means a scheme satisfies a given security requirement.

Protocol	Comp.	Indist.	Forward Sec.
Henrici [8]	*O(1)*	X	Δ
Dimitriou [6]	*O(1)*	X	Δ
Lee [10]	*O(1)*	Δ	X
Molnar [11]	*O(m)*	O	X
Our scheme	*O(m)*	O	Δ

* *m* : the number of *T*'s in the system

O : satisfy

Δ : partially satisfy

X : do not satisfy

Table.2. Efficiency and Security

We don't state anti-cloning in Table 2 because all schemes satisfy anti-cloning. Required computation of our scheme to find ID of a given *T* at *B* is *O(m)*, which means required computation is increased as the number of *T* 's in the system is increased. However, Table 2 shows our scheme offers the most enhanced security in RFID mutual authentication schemes with respect to user privacy. Our scheme is perfectly indistinguishable and almost forward secure. As compared with [11] whose computation at *B* is also *O(m)*, our scheme enhances user privacy with respect to forward security.

Table 3 compares our scheme with other two schemes which don't have a desynchronization problem about efficiency in *T*. Because our scheme requires another hash operation, it takes more time for authentication. However, our scheme reduces non-volatile memory by half. Since non-volatile memory is an expensive unit in *T*, it will cut down a cost of *T*.

Protocol	Lee [10]	Molnar [11]	Ours
Hash operations	2	2	3
Communication complexity	3*l*	4*l*	4*l*
Non-volatile memory	2*l*	2*l*	*l*

* *l* : the number of bits of a data unit

Table 3. Efficiency in *T*

6. Conclusion

In order to protect user privacy, we propose an RFID mutual authentication scheme which utilizes a hash function and synchronized secret information like many published schemes. We propose an RFID mutual authentication scheme which utilizes a hash function and synchronized secret information like others. To the best of our knowledge, our scheme offers the most enhanced security feature in RFID mutual authentication scheme with respect to user privacy including resistance against tag cloning allowing an additional hash operation. Moreover, our scheme reduces required non-volatile memory by half. Since non-volatile memory is an expensive unit in tags, it will cut down the tag's cost. Therefore, our scheme can be one of good options for diverse systems.

References

1. Auto-ID Center, “860MHz-960MHz Class I radio frequency identification tag radio frequency & logical communication interface specification proposed recommendation Version 1.0.0”, Technical Report MIT-AUTOID-TR-007, November 2002.
2. Auto-ID Center. http://www.autoidcenter.org/.
3. C.A.S.P.I.A.N. http://www.nocards.org/.
4. Gildas Avoine “Radio frequency identification: adversary model and attacks on existing protocols”, Technical Report LASEC-REPORT-2005-001, EPFL, Lausanne, Switzerland, Septemper 2005.
5. Gildas Avoine and Philippe Oechslin, “RFID traceability: a mulilayer problem”, Financial Cryptography − FC, pp. 125-140, March 2005.
6. Tassos Dimitriou, “A lightweight RFID protocol to protect against traceability and cloning attacks”, Conference on Security and Privacy for Emerging Areas in Communication Networks − SecureComm, September 2005.
7. Philippe Golle, Markus Jakobsson, Ari Juels, and Paul Syverson, “Universal reencryption for mixnets”, The Cryptographers’ Track at the RSA Conference − CTRSA, pp. 163-178, February 2004.
8. Dirk Henrici and Paul M¨uller, “Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers”, IEEE International Workshop on Pervasive Computing and Communication Security − PerSec, pp. 149-153, March 2004.
9. Ari Juels, Ronald Rivest, and Michael Szydlo, “The blocker tag: selective blocking of RFID tags for consumer privacy”, ACM Conference on Computer and Communications Security − ACM CCS, pp. 103-111, October 2003.
10. Su-Mi Lee, Young Ju Hwang, Dong Hoon Lee, and Jong In Lim, “Efficient authentication for low-cost RFID systems”, International Conference on Computational Science and its Applications - ICCSA, pp. 619-627, May 2005.
11. David Molnar and David Wagner, “Privacy and security in library RFID: issues, practices, and architectures”, ACM Conference on Computer and Communications Security − ACM CCS, pp. 210-219, October 2004.
12. Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita, “Cryptographic approach to ‘privacy-friendly’ tags”, RFID Privacy Workshop, November 2003.
13. Junichiro Saito, Jae-Cheol Ryou, and Kouichi Sakurai, “Enhancing privacy of universal re-encryption scheme for RFID tags”, Embedded and Ubiquitous Computing − EUC, pp. 879-890, August 2004.
14. Spychips. http://www.spychips.com/.
15. UHF wireless tag, Auto-ID Center, http://www.autoidcenter.org/research/mitautoid-tr007.pdf.
16. Stephen Weis, Sanjay Sarma, Ronald Rivest, and Daniel Engels, “Security and privacy aspects of low-cost radio frequency identification systems”, International Conference on Security in Pervasive Computing − SPC, pp. 454-469, March 2003.
17. Stephen Weis, “Security and privacy in radio-frequency identification devices”, Masters Thesis, MIT, May 2003.