RFID Authentication Protocol with Strong Resistance Against Traceability and Denial of Service Attacks

Jeonil Kang and DaeHun Nyang

Information Security Research Laboratory,
INHA University[*], Korea
dreamx@seclab.inha.ac.kr, nyang@inha.ac.kr
http://seclab.inha.ac.kr

Abstract. Even if there are many authentication protocols for RFID system, only a few protocols support location privacy. Because of tag's hardware limitation, these protocols suffer from many security threats, especially from the DoS (Denial of Service) attacks. In this paper, we discuss location privacy problem and show vulnerabilities of RFID authentication protocols. And then, we will suggest a strong authentication protocol against location tracing, spoofing attack, and DoS attack.

1 Introduction

A radio-frequency identification (RFID) system has been widely deployed mainly in supply chain management. Compared to using optical bar-code, RFID system has many benefits: quick reading, long recognition distance, obstacle-free, strength against the contamination, etc. Owing to these properties, a lot of tags can be read simultaneously during a few seconds. Also, RFID system can be effectively used in some applications such like animal tagging, high-way tolling, theft protecting, etc, whereas bar-code cannot handle these sides. RFID system will replace bar-code system very quickly.

Unfortunately, RFID system also has too many security risks specially when a high level of security is required. Generally, RFID tag has very low resources: low computing power and small memory size. Thus, it is very hard to apply existing security technologies that assumes very high computing power and large memory size to RFID tag. So, a lot of researches have been considered about security techniques for low-cost RFID systems [1-9]. We can classify security problems of RFID systems into two categories: information leakage and traceability.

A passive attacker might be able to overhear the information between a reader and some tags because the medium is the air in the RFID system. An active attacker may be able to send some bogus data that fakes the reader or tags to extract information from them. To prevent these attacks, many protocols have

[*] This work was supported by INHA UNIVERSITY Research Grant.

been proposed: blocker tag [6], RFID system using AES [9], minimalist XOR one-time cryptography [4], etc. But still the blocker tag is a very simple and strong method to protect information leakage.

On the other hand, tag's location information as well as tag's own data must be protected. Also, there are some solutions for this problem; hash-based ID variation protocol [1], hash chaining [2], universal re-encryption [7], etc. According to G. Avoine, however, it is hard to protect the threat because each RFID protocol layer (application, communication, and physical layer) has technical and actual defects [10].

In this paper, we'll discuss the location privacy of RFID system in section 2. Vulnerability of existing models of authentication protocols to location tracing and the DoS attack is shown in section 3. In section 4, we'll propose a strong authentication protocol for RFID system to solve the different early mentioned problems. Finally, we will summarize our results in section 5.

2 Location Privacy

2.1 Traceability at the Application Layer

Generally, RFID communication protocol consists of three layers; application, communication and physical layer. In the application layer, the information is handled by the user. Other RFID application programs use this information to identify objects. To make this possible, generally the information is made of product number and serial number. Also it might have a secret key or password for administrative purpose. The communication layer defines how to avoid collision might occur by the reader or the tags. This protocol is called "Collision Avoidance Protocol" or "Anti-Collision Algorithm." Collision avoidance protocol can be classified into probabilistic and deterministic methods according to the predictability of singulation time. The physical layer defines how to transmit data physically. (e.g. air interface - frequency, modulation, data encoding, synchronization, etc.)

Unfortunately, location privacy problem might occur in each layer. If an attacker can overhear all messages between the reader and tags which contents contain information that is never changed, the attacker can trace tag's movement using the information. Also, the attacker can use a type of collision avoidance protocol or tag identifier which can be used in collision avoidance protocol. Therefore, it is hard to prevent the attacker from tracing.

The one thing that we must consider to protect location privacy is accuracy (or correctness) of tracing. Accuracy of tracing using defects in communication and physical layer is lower than that in application layer. That's because there are many tags which use the same collision avoidance protocol and have same physical features in the real world. However, accuracy of tracing with messages in application layer is very high, because the databases server should distinguish each tag clearly. Therefore, it is very important to protect location privacy in the application layer. In this paper, we will propose a method to solve the location privacy problem in application layer.

2.2 Threat Model

Before describing our authentication protocol, a threat model is defined to analyze vulnerabilities of existing authentication model. By defining the threat model, we can restrict the ability of the attacker reasonably in RFID system.

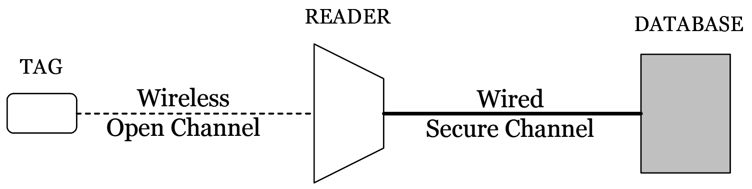

Fig. 1. RFID system threat model. An attacker cannot intercept in wireless channel because of a property of the air. Also, we assume that wired channel is secure in any case.

Like another threat model, we assume that an attacker can't intercept or modify messages. In the wireless open channel between tag and reader, the attacker can overhear all messages or insert fake messages, but he can't intercept or modify messages because the medium of transmission is the air itself. In wired secure channel between reader and database, an attacker can't eavesdrop, intercept, insert, or modify. In RFID system, we assume that reader and database server have pre-authenticated each other.

In the threat model, the attacker tries to trace location of tags and also tries to collapse system. The system collapse is caused by DoS(Denial of the Service), spoofing, or asynchronisation attack against specific, unspecific tags or against the database.

3 Vulnerabilities of Authentication Protocols

In this section, we will describe some methods to attack RFID system using structural problem of existing authentication protocols.

3.1 Un-terminated Session Attack

We cannot be sure that an attacker may always terminate the authentication session correctly. The authentication protocols - which send only hashed identifier (or hashed identifier with the nonce that was received from a reader) to a reader - are more prone to be traceable. Since the tag returns same hashed identifier in every session for attacker's queries, attacker can easily find tag's location. Unfortunately, the tag can't change its identifier to prevent this attack because the identifier always must be synchronized with that in the database. This asynchronisation (between database and tag) means that we cannot use the tag anymore.

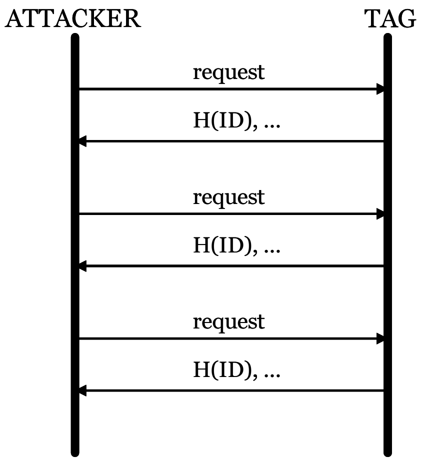

Fig. 2. Location tracing using un-terminated session. An attacker can trace target tag by sending request message and closing session.

3.2 Preemptive Locking

Consider that a tag receives a new request message, while it is already in the authentication session. The tag should choose its action: to ignore the request or to start immediately a new authentication session for the new request. The former gives a chance for the attacker to lock a tag preemptively. The attacker can make a target tag keep silence just by sending request message to the tag before it starts a legal authentication session. Because of this preemptive locking, the manufacturer should provide some mechanism with the tags to prevent this attack, or choose the latter one as the reaction of the tag.

Using timer, it looks rather easy to prevent preemptive locking. A tag has a timer and it starts the timer when session starts. When the timer is expired, the tag closes the session instantly. However, during the time that attacker possesses the session, tag cannot serve any request and thus, the attack might cause a severe degradation of the overall performance.

So, it seems to be better to start new authentication session for the newly arrived request message. But even in this case, tag is still weak against the attack described in section 3.3.

3.3 Stealth Bombing

If a tag is implemented to start new authentication session for the newly arrived request message, then it will suffer from so called, 'stealth bombing'. Stealth bombing is a kind of the DoS attacks. We assumed that the attacker can generate and insert fake messages, in section 2. What will happen if fake messages are inserted by the attacker during authentication session? If the tag which changes its response at every session using a random nonce opens a new session for a fake request message, the ongoing legal authentication session will fail by this illegal authentication trial.

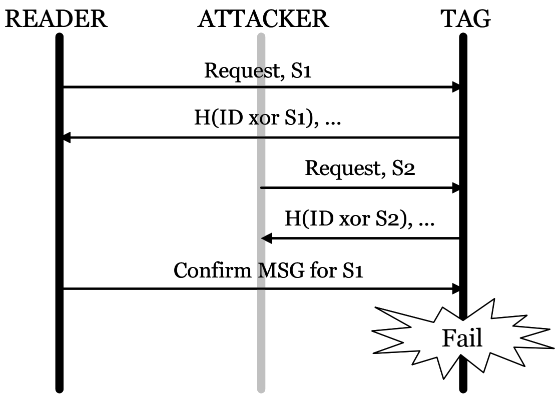

Fig. 3. Stealth bombing attack. An attacker can attempt DoS attack by sending request message to the tag in session. The tag has to decide its action against the request message.

Also, we can think another attacking method. Instead of sending a fake request message, the attacker might send an invalid confirm message for a legal session request. Some type of tags might abort the authentication session after receiving this confirm message.

Though this stealth bombing attack seems to be very hard to try because the communication layer will discard if the frame does not have a valid identifier defined by the communication layer, it is possible to send a fake request message that includes the valid identifier since it is disclosed during the normal communication between reader and tag.

3.4 DoS Attack Against the Database

Some authentication protocols use status-based flag to prevent problems of section 3.1. If the previous session terminated unsuccessfully, the tag responds using the message created with spare key and hint for the real identifier. These protocols must use large computational resource for recovering correct status and identifier. For another example, Ohkubo's protocol [2] must calculate a lot of hash function to search tag's identifier because the database server must compute $s_i = h(h(\cdots h(s_1)\cdots))$ from s_1. Since RFID tag has only small computing power, their approach to move tag's computation to database server is reasonable.

Because of this computational inefficiency, the attacker can try the DoS attack with very small effort against these protocols. It is enough to send some garbage messages toward the database server. Then, the database server will start to search identifier until finding it or retrieving all records. The attacker doesn't need to check whether the identifier exists or not when he makes fake messages.

In conclusion, a strong authentication protocol against DoS attacks should not search a large space for finding the hidden identifier. It seems to be incompatible to the solution of the problems in section 3.1.

4 Proposed Protocol

In this section, we will propose an authentication protocol which has a tolerance against DoS attack, provides location privacy, and strengthens other weakness.

4.1 Tag and Database Structure

A tag using this protocol has fields shown in below.

- SFlag (session flag): indicates whether tag is under an authentication session or not. When a session starts, it is set to true, and when a session ends either normally or abnormally, it is set to false. SFlag is initialized to false immediately after a tag comes to be active.
- ID (identifier): used to distinguish a tag from another tags during authentication session. $\{0,1\}^n$
- CWD (confirm word): used to confirm the ID of tag by the database. $\{0,1\}^n$
- R1 (Random nonce 1): changed at each session. If SFlag is true, it is replaced by saved R1. Or it is generated by the tag newly. $\{0,1\}^n$
- C (Counter): increased or randomly changed at each session. If SFlag is true, it is replaced by saved C. Or it is changed by the tag. $\{0,1\}^m$
- THR_COUNT (threshold counter): indicates how many trials have happened. When THR_COUNT reaches THR_MAX, the tag terminates current session and starts a new session. It can report DoS attack optionally.

The database has the structure shown in figure 4. It must prepare the same number of slots as that of all possible H(ID||C), while H denotes cryptographic hash function, and || denotes concatenation. When ID is constant, there are 2^m numbers of slots that have the same ID. If H(ID||C)of different ID conflicts, they are 'linked' in the same slot. If m is 10, the database server has to calculate 1024 (=2^{10}) hash values in advance. Though it seems to require much computation of database server, hash values are computed after authentication in the idle time. It is possible to use special purpose unit for computing hash values. Actually, the computations spend small of time (about 0.283 second in internal md5 testing, 0.017 second if only hashing), and we have thought it is not a problem at all if a reasonable m is used.

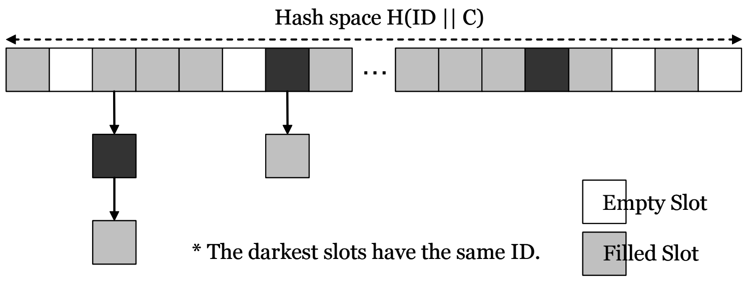

Fig. 4. Example of the database

4.2 Basic Protocol

Our protocol is shown in figure 5 and 6. The protocol solves the security problems referred in section 3.1-3.4.

In order to prevent un-terminated session attack, it must reserve the freshness of response message from tag between sessions. For reserving the freshness of messages, all messages should be generated with secure random nonce at every session. However, if insecure random nonces are used or the attacker can use his number as nonce in authentication session, it can't prevent this attack. Because the tag can't know where the random number included in the request message is from, the random number from a legal reader also isn't trusted. Therefore, only random number from the tag itself is trusted. Consequently, the tag should generate random nonces newly when new session starts.

In order to prevent preemptive locking and stealth bombing attack at once, it needs to handle the session very carefully. Preemptive locking attack is possible because of the session-preemptive feature of the authentication protocol, and stealth bombing attack is mountable because of the session-nonpreemptive feature of the authentication protocol. An authentication protocol cannot be preemtive and non-preemptive at the same time.

Even though the tag receives request message during authentication session, the attacker can not preempt the session if the tag sends the same response message repeatedly until normal termination. We can also frustrate stealth bombing attack using the repeated transmission of the same response message. Note that this method is possible only if random nonces are generated by the tag. By reserving the identicalness of response message from a tag during one session, our protocol can be very robust against preemptive locking and stealth bombing attack. Our protocol provides a strategy that dose not frustrate the attacks, but ignore them.

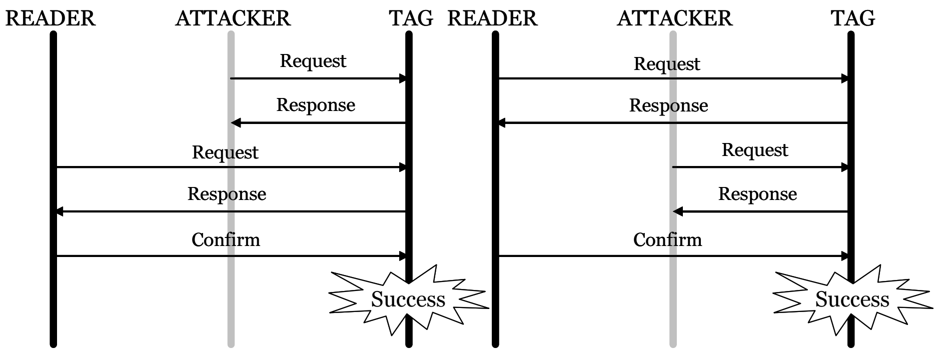

Fig. 5. How to solve preemptive locking and stealth bombing. All response messages have the same value. If a tag can respond with the same response in one session, these two attack can be thwarted.

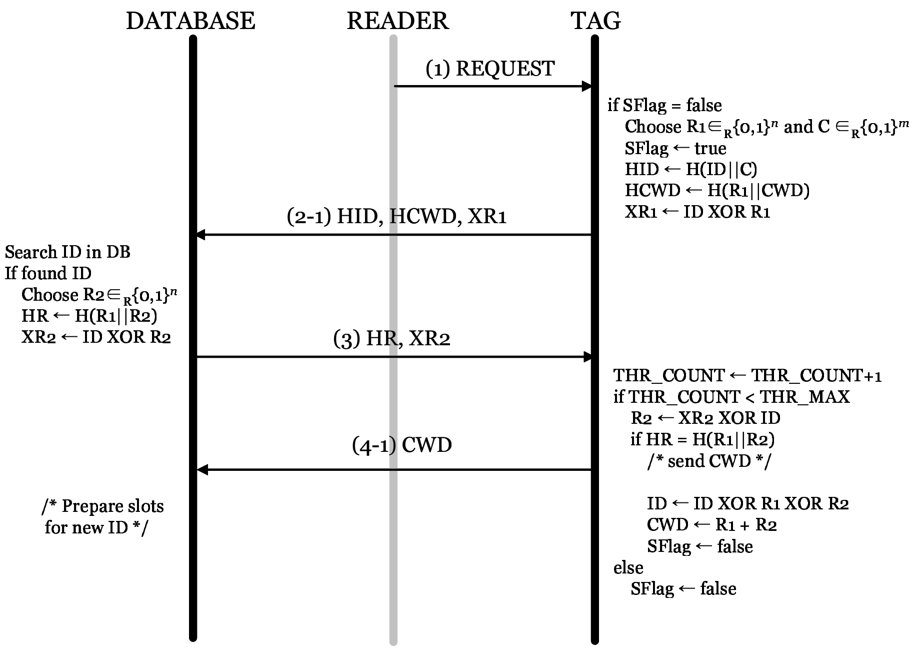

Fig. 6. Proposed protocol

However, to prevent DoS attack using insertion of an invalid confirm message, we need another strategy. To prevent this attack, tag have to wait for a valid confirm message for reserved time. Because it is too expensive to use a timer in RFID tag, threshold counter can be used instead of the timer.

Our protocol illustrated in figure 6 runs as the following:

step 1. When a reader needs tag's data, the reader sends message (1) {REQUEST} to tag

step 2. When a tag receives REQUEST message, it checks SFlag. If SFlag is true, the tag uses R1 and C which are already in memory. Else, the tag chooses random numbers R1 $\in_R \{0,1\}^n$ and C $\in_R \{0,1\}^m$. Also, the tag sets SFlag to true. And then, the tag computes and sends message (2-1) {HID←H(ID||C), HCWD←H(R1||R3||CWD),XR1←ID⊕R1} to the database server through the reader.

step 3. Using HID of message (2-1), the server can get candidates for ID. Also the server gets candidates for R1 by computing XR1⊕(candidates for ID). The server can find a real ID of the tag by checking whether HCWD is the same as H((candidates for R1)||R3||(candidates for CWD)) or not.

step 4. If the server can't find any satisfied ID in previous step, the server regards this message as an attack and ignores it. When the server finds ID, the server chooses another random nonce R2. And then, the server generates and sends message (3) {HR←H(R1||R2),XR2←ID⊕R2} to the tag.

step 5. If the tag receives message (3), it increases THR_COUNT. Then the tag checks whether H(R1||(ID⊕XR2)) is the same as HR or not if only if THR_COUNT < THR_MAX. If it matches, the tag has to send message (4-1) {CWD} to the database server and continues step 7.

step 6. If the tag has some reason for reject or message (3) is not valid, the tag should wait another messages until THR_COUNT expires. If THR_COUNT reaches THR_MAX (or expires), the tag sets SFlag to false, ignores all next messages and waits until another reader opens a new session.

step 7. If the database server receives correct message (4-1), the server changes its ID and CWD by new ones. Otherwise, the server consider previous message (2-1) to replay attack and halts the process. If the tag has no error to send message (4-1), the tag changes its ID and CWD, and sets SFlag to false. Also the server must prepare hash space for all possible H(ID||C). Here, CWD consists of R1 and R2 in the previous session. That is,

$$\mathrm{CWD}_i = (\mathrm{R1}_{i-1} + \mathrm{R2}_{i-1}) \bmod 2^n$$

where i denotes the current session and $i-1$ denotes previous session. Also

$$\mathrm{ID}_i = \mathrm{ID}_{i-1} \oplus \mathrm{R1}_{i-1} \oplus \mathrm{R2}_{i-1}$$

In order to find ID from H(ID||C) in (2-1) message, the database server must prepare all possible slots. If |ID| denotes the number of tags, the number of slots is $2^m \times |\mathrm{ID}|$. If all possible candidates of H(ID||C) are distributed informly and 2^n is larger then $2^m \times |\mathrm{ID}|$, there is no collision in the same slot. If the system has some collisions, it might have a few liked slots. So, it is good to system to have a number of tags smaller than $2^{(n-m)}$. If the number of tags is about 4,294,967,296(=2^{32}) in system, 42 is enough for n and 10 is enough for m for structure efficiency of the database. But it is strongly recommended to use ID and C with large length for security.

4.3 Security Analysis

Against location tracing: When the attacker wants to trace target tag, he can use message (2-1) and (3) because both message (1) and (4) do not have any information. However, if he can't find any collision pair from hashed messages, he can only use ID ⊕ R1 and ID ⊕ R2 to get clues about the tag. Because he can only get a result of operation ⊕ with these fields, he will only know (ID⊕R1)⊕(ID⊕R2)=R1⊕R2. Since these R1 and R2 are generated by the tag and the server newly at each session, R1⊕R2 also can't be a clue for tracing, even though he can observe all authentication messages.

Also, the attacker can't estimate the messages for next session. Because he knows R1⊕R2, and also knows CWD of tag for next session. However, only with CWD, an attacker cannot trace the location because CWD is masked with R1 such as H(R1||CWD). An attacker might trace the tag using all possible values of H(ID||C), but he requires ID which is not exposed to him.

Against spoofing attack: If an attacker wants to spoof the database server or tag for asynchronism between the server and tag, he must generate message (2-1) or (3) correctly. (This asynchronism enables an attacker to mount a kind of DoS attack.) An attacker can generate a valid H(R1||CWD) only with a probability of 2^{-n} because he must guess a valid ID. In the other hand, an attacker can choose R1 and ID in order to attack against unspecific tag. However, the probability of guessing CWD correctly is 2^{-n}. Since a probability that the guessed ID is found in the database is $|\mathrm{ID}| \times 2^{-n}$, he can succeed in this attack having a probability of $|\mathrm{ID}| \times 2^{-2n}$.

An attacker can spoof a tag by sending illegal message (3). However, he can generate a valid H(R1||R2) for ID$\oplus$R2 only with a probability of 2^{-n} since he can extract a valid R1 from ID$\oplus$R1 of message (2-1) if he can guess a valid ID.

Against denial of service attack: DoS attack against a tag was described in section 3.2 and 3.3. Also we explained how to solve these problems at once in section 4.2 and our protocol works according to those principles. Using random nonces from tag and threshold counter for attack trials, the protocol has immunity against the DoS attack. Assuming an attacker cannot modify messages, he has to send or insert messages to mount DoS attack. Even though the attacker tries to do preemptive locking or stealth bombing attack by sending message (1), he cannot succeed because tag answers the request of the attacker using the same response message (2). Also he cannot guess correctly ID, R1 or C from message (2).

When the database server searches ID of the tag, it does not need to hash because all possible values of H(ID||C) are pre-computed already. Using this strategy, an attacker can't make the database server compute hash value in any cases. Note that the only messages which the attacker can send to the database server are message (2-1) and (4-1). The amount of burden that the database server experiences from one attack message is just as much as one searching of ID space. Thus, even if the attacker can insert or replay message (2-1) and (4-1) in the session, the database server will ignore that message.

When the tag waits message (3), the attacker can send invalid messages to tag for making authentication fail. However, the tag increases THR_COUNT against these messages, and finally authentication will succeed if a valid message arrives before THR_COUNT expires. In the other side, the server must send message (3) in time.

4.4 Alternative Protocol

We propose another protocol, which is a slight modification of our original protocol. The database server authenticates the tag first in the original protocol. But, in the alternative protocol, the tag authenticates the database server first. After message (2-2) is sent, the server makes candidates of ID, and tries to be authenticated by sending message (3) several times until it finds a valid ID. After the tag receives a valid message that includes valid ID and R2, it sends CWD to the server in plaintext, and changes its ID by ID$\oplus$R1$\oplus$R2. If a valid CWD arrive, the database

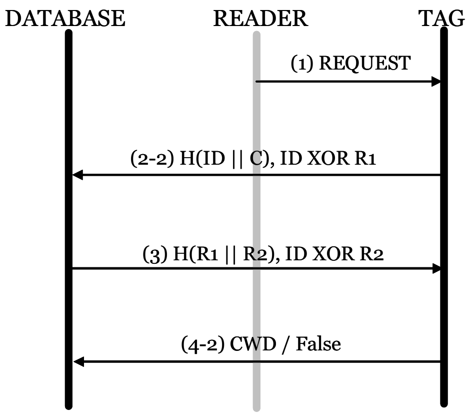

Fig. 7. Alternative Protocol

server will changes ID to ID⊕R1⊕R2 and CWD to R1+R2. Even though R1⊕R2 is obtained easily by observing previous session, it is hard to extract R1+R2 from R1⊕R2. So, the attacker can't get any clues for tracing target tag.

Even though alternative protocol might require slightly more message in average than the original one, it needs only two hash computations.

5 Conclusion

Even if there are many authentication protocols for RFID system, only a few protocols support location privacy. Because of the tag's hardware limitation, these protocols suffer from many security threats, especially from DoS attack.

We established threat model for RFID system and explained some special attack for general authentication protocol. In order to solve these problems, we suggested two strategies: keeping the nonce identical during a session, and threshold counter. With these schemes, we proposed a strong authentication protocol against DoS attack supporting location privacy.

Finally, we checked the strength of our protocol against three categories of attacks, and we concluded that our protocol has reasonable security strength. In addition, we introduced alternative protocol that reduces one more hash operation.

References

1. Dirk Henrici and Paul Müller : Hash-based Enhancement of Location Privacy for Radio-Frequency Identification Device using Verying Identifiers. University of Kaiserslautern, Germany, Workshop on Pervasive Computing and Communications Security - PerSec2004, pp. 149-153, IEEE, 2004
2. Miyako Ohkubo, Koutarou Suzuki and Shingo Kinoshita : Cryptographic Approach to 'Privacy-Friendly' Tags. NTT Laboratories, Japan, RFID Privacy Workshop MIT, 2003

3. István Vajda and Levente Buttyán : Lightweight Authentication Protocols for Low-Cost RFID tags. Budapest University of Technology and Economics, Hungary, 2003
4. Ari Juels : Minimalist Cryptography for Low-Cost RFID Tags. RSA Laboratories, USA
5. Ari Juels : Yoking-Proofs for RFID Tags. RSA Laboratories, USA
6. Ari Juels, Ronald L. Rivest and Michael Szydlo : The Blocker Tag:Selective Blocking of RFID Tag for Consumer Privacy. RSA Laboratories, USA
7. Philippe Golle, Markus Jakobsson, Ari Juels, and Paul Syverson : Universal Re-encryption for Mixnets. RSA Laboratories, USA
8. Stephen J. Engberg, Morten B. Harning and Christian Damsgaard Jensen : Zero-knowledge Device Authentication: Privacy & Security Enhanced RFID preserving Business Value and Consumer Convenience. Privacy, Security and Trust 2004 - PST2004, EU Smarttag Workshop, 2004
9. Martin Feldhofer : A Propsal for an Authentication Protocol in a Security Layer for RFID Smart Tags. Institute for Applie Information Processing and Communications (IAIK), Graz University of Technology, Austria
10. Gildas Avoine and Philippe Oechslim : RFID Traceability: A Multilayer Problem. École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, Financial Cryptography - FC'05, LNCS, Springer, 2005