Secure and Low-cost RFID Authentication Protocols

Yong Ki Lee[1] and Ingrid Verbauwhede[1,2]
[1] University of California, Los Angeles
[2] Katholieke Universiteit Leuven
{jfirst, ingrid} @ ee.ucla.edu

Abstract

In this paper we propose two RFID (Radio Frequency Identification) authentication protocols for secure and low-cost RFID systems. The first protocol SRAC (Semi-Randomized Access Control) is designed using only a hash function as security primitives in tags. In spite of very restricted functionality, SRAC resolves not only security properties, such as the tracking problem, the forward secrecy and the denial of service attack, but also operational properties such as the scalability and the uniqueness of MetaIDs. The second protocol A-SRAC (Advanced SRAC) resolves the replay attack in the cost of a random number generator in tags. Moreover, our schemes have significantly reduced the amount of tag transmissions which is the most energy consuming task.

1 Introduction

The RFID technology has been one of the hottest issues in the wireless communication area. One of the reasons many developers are researching this topic is that the RFID is supposed to replace the bar code systems. This expectation has been accelerated since the adoption of EPCglobal Gen2 [8]. However, the application area is not restricted to product supply chains but covers livestock tracking, airline baggage, road toll management, hotel room access and so on. In order to be popular in commercial markets, the RFID system should overcome the restriction of cheap RFID tags. The limited price means limited functionalities and resources in tags. Because of the limitation, using asymmetric or symmetric key encryption algorithm or making memory secure in tags is improper [1]. To solve security problems related with low-cost RFID systems, many authentication protocols were proposed. However, those protocols could not satisfy the RFID security requirements and/or operational requirements. According to the best of our knowledge, there is no published authentication protocol that deals effectively on security and operational requirements.

In this paper we review and classify previously proposed protocols and their drawbacks, and propose new protocols which satisfy not only several major security properties such as the tracking problem, the forward secrecy, the denial of service attack and the replay attack, but also operational properties such as the scalability and the uniqueness of MetaIDs.

The remainder of this paper is organized as follows. Section 2 explains the desired properties in RFID systems. Section 3 introduces related works and points out the problems they have. In section 4 and 5, we propose new RFID authentication protocols and analyze several operational and security properties, and conclude this paper in section 6.

2 Desired Properties in RFID System

Even though the resources allowed in RFID tags are very restricted, RFID systems are supposed to satisfy some operational and security requirements. The following sub-sections explain those requirements.

2.1 Operational requirement

Considering that most applications of RFID systems require a lot of tags to be used, the scalability is a required property. For example, in order to apply RFID systems to a large library, more than 1 million RFID tags should be applicable. Another operational requirement is the uniqueness of MetaIDs. Many published protocols [2, 3, 4, 6] make MetaIDs using a hash function. One problem is that we cannot assure the uniqueness of hash outputs. In order to avoid the conflictions of hash outputs, we need to have enough length of hash outputs. Otherwise the confliction of MetaIDs can cause serious problems in the system. In another word, if we can make sure the uniqueness of MetaIDs, we can reduce the size of MetaIDs, which means the reduction of transmission and memory.

2.2 Security requirement

The most important security problems are the cloning and tracking problems. However, there are more security properties which are useful in RFID systems. We consider

the other properties when we analyze our proposing protocols.

First of all, to prevent the cloning problem in low cost tags, it is indispensable to store some secret information in tags which cannot be made arbitrarily by attackers. And the secret information should be used in authentication between a tag and a reader. There are two ways to store secret information in tags. The first way is to store common secret information among all the readers and the tags. In this way, as long as the information is secure, this method makes the systems very secure and efficient. However, if a single tag is compromised, attackers may clone other tags or may control all the tags using the secret information. The other way is to store secret information which is pertinent to only a specific tag. In this case, even if some secret information is compromised, the information will be irrelevant to the other tags so that they still remain secure.

Another security property is to resolve the tracking problem. If the responses of a tag are constant or predictable by attackers, the tag can be tracked. To prevent the tracking problem, the responses of tags must appear random to the attackers.

3 Related Works

The authentications are done in two ways. By authenticating a reader to a tag, a tag is to be ready to open its information to a reader, and by authenticating a tag to a reader, the system prohibits the usage of fake tags. We can divide published authentication protocols into two types. The first type is the fixed access control in which a tag replies a reader with a fixed message. The second type is the randomized access control in which a tag replies to a reader with a pseudo-random message which varies each time of the responses.

The fixed access control is the simplest type so that tags can be implemented in a cheap price. However, this kind of protocols is under the tracking problem. [2] proposed a fixed access control using a hash based access control, where tags reply with MetaIDs, which are the hash outputs of their real IDs. Even though attackers cannot figure out the real ID, the constant responses of tags cause the tracking problem.

A solution to prevent the tracking problem is the randomized access control. In order to randomize messages, a reader and a tag need to share some secret information which is unknown to attackers so that only the entities which have the secret information can interpret the randomized messages. Again, the randomized access control can be divided into two types depending on whether all the readers and the tags share the same secret information.

Without sharing the common secret information among all the readers and the tags, making the response pseudo-random causes some drawbacks. [2, 6] described protocols which resolve tracking problems, but the systems are not scalable since the server needs to perform hashes for all the tags' ID every time of authentication protocols. One approach to resolve the un-scalability of randomized access control is proposed in [4]. This scheme used a cryptanalytic method. However, this method also causes some other problems. Since this protocol uses time-memory trade-off method [5], in order to reduce the searching time they have to increase the amount of memory in the server. Another problem is that the searching algorithm is probabilistic, i.e. there is some probability to fail in searching for a tag's ID. Even though they are saying the failure probability is small, it can cause a crucial problem in certain applications.

Protocols proposed in [3, 7] resolve the tracking problem by sharing the common secret information among all the readers and the tags. Even though these schemes are scalable and resolve the tracking problem, they have a crucial problem. By capturing and compromising only one tag, attackers can reveal the secret information. Once the secret information is revealed, the tags which share the secret information will be under attack and attackers may clone some other tags. Moreover, the protocol in [7] uses a symmetric key encryption algorithm which is unsuitable in low-cost RFID systems.

4 SRAC (Semi-Randomized Access Control)

Therefore, we need protocols which are designed considering both problems, i.e. the operational and security requirements. According to the best of our knowledge, there is no published protocol which resolves effectively both problems. In this section, we propose SRAC (Semi-Randomized Access Control) which is designed considering both requirements.

4.1 Protocol Description

We suppose that the server and the reader have sufficient resources to use strong symmetric or asymmetric key algorithms so that the communications between them are secure. Therefore, we care only about the communications between the reader and tags and we assume that the messages arrived to the reader are securely passed to the server. In the following protocols, we do not distinguish between the reader and the server. However, the computational load can be split into two parts or can be done only at the server depending on applications. Figure 1 and Table 1 illustrate SRAC. In this scheme, each tag contains its own key which is irrelevant to the other tags.

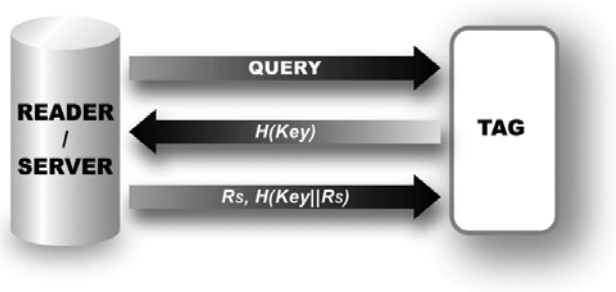

Figure 1 SRAC protocol

① Reader sends Query to Tag.

② Tag sends MetaID ($= H(Key)$) to Reader/Server.

③ Server looks up Key using MetaID, generates a random number R_S , and checks whether $H(Key \oplus R_S)$ is unique among the other MetaIDs. If it is not unique, Server regenerates R_S until $H(Key \oplus R_S)$ becomes unique.

Server updates Key as follows.

If $H(Key_{Curr}) = \text{MetaID}$

$Key_{Prev} \leftarrow Key_{Curr}$, $Key_{Curr} \leftarrow H(Key_{Curr} \oplus R_S)$

If $H(Key_{Prev}) = \text{MetaID}$

$Key_{Curr} \leftarrow H(Key_{Prev} \oplus R_S)$

$Key \leftarrow Key_{Prev}$

Server sends R_S and $H(Key \| R_S)$ to Tag through Reader.

④ Tag checks whether $H(Key \| R_S)$ is correct.

If it is correct, Tag updates $Key \leftarrow H(Key \oplus R_S)$.

Table 1 SRAC Protocol Flow

The reason we inserted R_S into a hash function is that the tag needs to check the integrity of R_S . The server authenticates tags by checking whether the received MetaID is on the server's database, and tags authenticate the server by checking $H(Key \| R_S)$. In order to be resilient against the denial of service attack, the key update of the server must be more sophisticated than tags. The server keeps two keys, the current key (Key_{Curr}) and the pervious key (Key_{Prev}). The reason is on the following sub-section.

4.2 Operational and Security Properties

Scalability

In the proposed protocol, the server can search out a tag's Key using MetaID ($= H(Key)$). Since the database of the server can be indexed using MetaIDs, the searching is efficient and thus the system is scalable.

Uniqueness of MetaIDs

Our proposed protocol resolves this problem by checking whether an updating MetaID is to be unique in step ③. In this protocol the server only needs to regenerate a random number R_S again until a new MetaID becomes unique. As long as the confliction probability of MetaIDs is not too high, the overhead will be reasonable. Since the uniqueness is confirmed, we do not need a large size of MetaIDs to evade the conflictions of MetaIDs. Therefore, we can significantly reduce the number of bits used in MetaIDs, which means less energy to transmit and less memory to store a MetaID.

For example, SHA-1 digests an input into 160 bits. In this case, even if we consider a large number of tag IDs, say $2^{20} \approx 10^6$, the approximate confliction probability is $2^{-140} \approx 10^{-42}$, which is negligible. However, the larger size of MetaIDs means more energy in transmission, which is one of the most energy consuming tasks of tags. If we can assure that the MetaIDs do not conflict, we can reduce the size of MetaIDs as long as the probability that a random number matches with any MetaID is not too high. If the size of a MetaID is 40 bits and the number of tags is $2^{20} \approx 10^6$, we can reduce 75% of the transmission energy of MetaIDs, where the probability that a random number matches with any MetaID is $2^{-20} \approx 10^{-6}$, which is acceptable depending on applications.

Resources required for authentication

For each time of authentication, the required cryptographic computations in tags are only three hashes, and the amount of transmission of a tag is the size of the hash output. Since our scheme confirms the uniqueness of MetaIDs, we can reduce the size of the hash outputs. There are two ways to reduce the hash outputs. We may use a hash function to produce small outputs or perform modulo operations to reduce the hash outputs. Therefore, our scheme can be implemented very efficiently in computation and transmission.

Cloning Problem

The secrete information stored on each tag is pertinent to each tag. Even if some tags are compromised, the other tags are irrelevant to the compromised information. Therefore, attacker cannot make any other fake tag except for the compromised tags.

Tracking Problem

Tracking problem can occur when responses of a tag are constant. The proposed scheme resolves this problem by changing tags' secret information whenever the authentication is successful. Even though our proposed protocols do not resolve perfectly the tracking problem, the protocol handles the problem effectively. The fact is that a method which resolves perfectly both problems, the cloning and tracking problems, is unknown without having unreasonable operational overhead or using symmetric or

Properties \ Access Controls	Fixed Access Control [2]	Randomized Access Control		Our proposed Schemes (Semi-Randomized Access Control)	
		Without sharing common secret information [2,6]	With sharing common secret information [3,7]	SRAC	A-SRAC
Scalability	Scalable	Un-scalable	Scalable	Scalable	Scalable
Cloning Problem	Strong	Strong	Vulnerable	Strong	Strong
Tracking Problem	Vulnerable	Strong (1)	Strong (2)	Strong (3)	Strong (3)
Replay Attack(4)	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Strong

* (2) The scheme will be vulnerable after compromising a single tag.
* (3) The schemes are not strong as much as (1), but stronger than the others.
* (4) If the reader-to-tag or the tag-to-reader authentication is vulnerable, we marked "Vulnerable".

Table 2. Comparison of some properties among our schemes and others

asymmetric encryption algorithm, or without causing some other crucial problems.

Forward Secrecy

In this protocol, the revealed secret information of tags cannot affect the past secrecy. Even if all the communications between a reader and a tag were eavesdropped and recorded, using the current secret information, i.e. Key, attackers cannot infer the past secret information. This is because a reader and a tag update their secret information using a hash function each time of the protocols. Therefore, as long as a hash function is not invertible, the past secret information is secure.

Denial of Service Attack

If the server updates keys in the same way as tags, the protocol is under the denial of service attack. Suppose the server keeps only the current key per tag. Then, the attack is possible as follows. An attacker generates a jamming signal at step ③ so that the tag cannot receive the message from the reader and does not update its secret information while the server updates the tag's secret information. After this attack, the secret information will be inconsistent between the reader and the tag. Therefore, the later protocols will fail.

To resolve this problem, the server only needs to store the previous secret information for each tag. If the server fails in searching for a MetaID, the server can search out through the previous MetaIDs. Since only one more MetaID for each tag is stored in the server, we can effectively prevent the denial of service attack.

5 A-SRAC (Advanced Semi-Randomized Access Control)

One problem of SRAC (Advanced Semi-Randomized Access Control) is that it is under the replay attack. Attackers can masquerade as either a reader or a tag by replaying the past messages. The first case is to masquerade as a tag. Note that the server stores two MetaIDs, i.e. the current one and the previous one, per tag and if it is matched with either of two, the server will authenticate a tag. Attackers may eavesdrop and reuse the recently used MetaID and will succeed to be authenticated. This scenario can cause crucial problems at some applications. For example, suppose this protocol is implemented for a door access control. Just eavesdropping authentication protocols, attackers can disguise an authorized tag. Masquerading as a reader is also done by eavesdropping the reader's message, which is straightforward.

In this section we propose A-SRAC (Advanced Semi-Randomized Access Control), which prevents the replay attack in the cost of a random generator implantation and more message transmission in tags. The overhead of the server and the reader is negligible considering their sufficient resources.

5.1 Protocol Description

To prevent the replay attack, we use a challenge and response method for both directions. Figure 2 and Table 3 illustrate the protocol flow of A-SRAC, which is designed to prevent the replay attack.

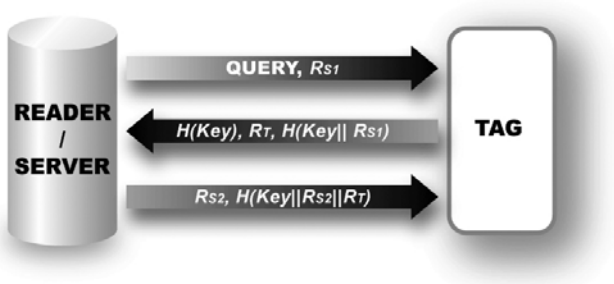

Figure 2 A-SRAC protocol

① Reader sends Query and a random number R_{S1} to Tag.

② Tag sends MetaID ($= H(Key)$), R_T and $H(Key \| R_{S1})$ to Reader/Server.

③ Server looks up Key using MetaID, generates a random number R_{S2}, and checks whether $H(Key \oplus R_{S2})$ is unique among the other MetaIDs. If it is not unique, Server regenerates R_{S2} until $H(Key \oplus R_{S2})$ becomes unique.

Server checks whether $H(Key \| R_{S1})$ is correct.

Server updates Key as follows.

If $H(Key_{Curr}) = \text{MetaID}$

$Key_{Prev} \leftarrow Key_{Curr}$, $Key_{Curr} \leftarrow H(Key_{Curr} \oplus R_{S2})$

If $H(Key_{Prev}) = \text{MetaID}$

$Key_{Curr} \leftarrow H(Key_{Prev} \oplus R_{S2})$

$Key \leftarrow Key_{Prev}$

Server sends R_{S2} and $H(Key \| R_{S2} \| R_T)$ to Tag through Reader.

④ Tag checks whether $H(Key \| R_{S2} \| R_T)$ is correct.

If it is correct, Tag updates $Key \leftarrow H(Key \oplus R_{S2})$.

Table 3 A-SRAC Protocol Flow

The server authenticates tags by checking whether the received MetaID is on the server's database and checking $H(Key \| R_{S1})$, and tags authenticate the server by checking $H(Key \| R_{S2} \| R_T)$. Except for random numbers which are used as challengers, A-SRAC is similar to SRAC.

5.2 Operational and Security Properties

Basically all the operational and security properties of SRAC are inherited to A-SRAC except for the tags' required resources whose analysis is straightforward. The additional security property is the resistance against the replay attack. Since a reader and a tag both confirm the received message using hash outputs which contain internally generated random numbers, attackers cannot reuse the past messages.

Table 2 represents the comparison of some important properties among our schemes and others.

6 Conclusion

In this paper we proposed two RFID authentication protocols, SRAC and A-SRAC. The proposed schemes can be implemented efficiently, since SRAC uses only a hash function and A-SRAC uses a hash function and a random generator for security primitives. SRAC and A-SRAC both resolve the problems of the scalability and the uniqueness of MetaIDs and also resolve the tracking problem, the forward secrecy and the denial of service attack. A-SRAC prevents the replay attack which can be a crucial problem depending on applications such as a door access control. Moreover, since our schemes ensure the uniqueness of MetaIDs, the transmission message in tags is reduced. In the case of SRAC, we can reduce 75% of the transmission of tags. Therefore, SRAC and A-SRAC will be good solutions for low-cost RFID systems that require good operational and security properties.

Acknowledgement

This research has been supported by UC Micro and NSF (Grant SRC-2003-HJ-1116). We appreciate valuable discussion with Stefaan Seys.

References

[1] Sanjay E. Sarma, Stephen A. Weis, and Daniel W. Engels, "RFID Systems, Security & Privacy Implications", Auto-ID center white paper, Feb 2003. http://www.autoidlabs.org.

[2] Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," in The First International Conference on Security in Pervasive Computing (SPC 2003), March 2003.

[3] Xingxin(Grace) Gao, Zhe(Alex) Xiang, Hao Wang, Jun Shen, Jian Huang and Song Song, "An Approach to Security and Privacy of RFID System for Supply Chain," Proceedings of the IEEE International Conference on E-Commerce Technology for Dynamic E-Business (CEC-East'04), 2004.

[4] Gildas Avoine and Philippe Oechslin, "A Scalable and Provably Secure Hash-Based RFID Protocol," The 2nd IEEE International Workshop on Pervasive Computing and Communication Security Persec 2005, March 2005.

[5] Philippe Oechslin, "Making a Faster Cryptanalytic Time-Memory Trade-Off," In Advances in Cryptology - CRYPTO '03, 2003.

[6] Miyako Ohkubo, Koutarou Suzuki and Shingo Kinoshita, "Cryptographic Approach to "Privacy-Friendly" Tags," RFID Privacy Workshop @ MIT, 2003.

[7] Martin Feldhofer, "An Authentication Protocol in a Security Layer for RFID Smart Tags," IEEE MELECON 2004, May 2004.

[8] EPCglobal, "Specification for RFID Air Interface," http://www.epcglobalinc.org