A Lightweight Authentication Protocol based on Universal Re-encryption of RFID Tags

Youngjoon Seo, Hyunrok Lee and Kwangjo Kim*

*International Research center for Information Security, ICU

Abstract

RFID(Radio Frequency Identification) is now being used in everything for economic feasibility and convenience. In contrast, RFID tags may infringe on user’s privacy. A number of previous schemes are suitable for high-end RFID. In this paper, we propose a lightweight authentication protocol for RFID tags that require only a random number generator. Our protocol exploits a proxy and the universal re-encryption which has several advantages: (1) readers do not share the secret key with tags, (2) ownership transfer is possible, (3) desyncronization problem is completely eliminated, (4) and our protocol is scalable.

I. Introduction

RFID is recently becoming popular, and play definitely an important role in moving on ubiquitous society in the world due to deploying its convenience and economic efficiency; furthermore, RFID nowadays comes into the spotlight as a technology to substitute barcodes system since RFID can solve several problems in barcode system.

On the other hand, RFID is jeopardized from various attacks and problems as an obstacle of widespread RFID deployment; attacks are spoofing, swapping, replay and DoS(Denial of Service); problems are privacy, tracing, tag cloning, desyncronization and overhead in back-end server due to a large number of tags. Table 1 shows that a lot of countermeasures to protect these attacks and to solve these problems have been proposed, which fall into different categories. Permanent and temporary tag deactivation are analogous to power-down of personal computer owing to the dread of being cracked. In other words, it can not be an eventual solution. On-tag cryptographic primitives and on-tag access control require high-end RFID tags (On-tag means that mechanisms are located on the RFID tags themselves; in contrast, off-tag is taken care of by the external device) ; that is, they are not reasonable to implement RFID tags so far. Low-cost is important issue to proliferate RFID technology into the billion of items.

Table 1. Countermeasures for preventing attacks

Countermeasure	Example
Permanent tag deactivation	kill command, tag destruction
Temporary tag deactivation	Faraday cages, sleep/wake command
On-tag cryptographic primitives	stream ciphers, symmetric or symmetric cryptographic alogrithm
On-tag access control	hash-based, pseudonym-based, tree-based schemes
Off-tag access control	blocker and noisy tag, proxy-based schemes

In this paper, we propose a authentication protocol for low-cost tags; our protocol also can be called by off-tag access control mechanism to make and proliferate low-cost tags based on universal re-encryption which has several good security properties.

1. Notations

We use the notations for entities and operations as summarized in Table 2 throughout the paper.

Table 2. Notations

R	RFID tag reader
T	RFID tag
P	Proxy
B	Back-end Server
$\mathcal{G}$	Underlying group for ElGamal cryptosystem
q	The order of $\mathcal{G}$
g	Generator for $\mathcal{G}$
ID	Identifier
m	Message
$\in_U$	Uniform, and random selection

II. Universal Re-encryption

UR(Universal Re-encryption) is introduced by Golle *et al.*[2] C' is said to represent re-encryption of C(Ciphertext) provided that two ciphertexts are decrypted to the same plaintext.

UR has several security advantages. First, re-encryption can be done without knowledge of PK(Public Key). Second, the one time decryption is sufficient to get the plaintext no matter how many times the re-encryption is done. In most case, secret keys should be refreshed not to be traced on a regular basis. These secret keys have to be synchronized among parities who share the secret keys after key update. However, we can eliminate desyncronization entirely exploiting second property of UR. Third, UR based on ElGamal encryption algorithm is semantically secure under the re-encryption if the adversary cannot determine b for given two re-encryptions (C'_b, C'_{1-b}) with probability significantly greater than 1/2 (See more in [3]).

1. Description

UR consists of four functional components as follows.

- *Key generation* : *TA*(Trusted Authorities) take charge of generating keys. *TA* generate a secret key *SK* x and a public key *PK* $y=g^x$, where $x \in_U Z_q$.
- *Encryption* : *R* or *P* selects random encryption factor $r=(k_0,k_1) \in Z_q^2$, and then generates $C=[(\alpha_0,\beta_0);(\alpha_1,\beta_1)]=[(my^{k_0},g^{k_0});(y^{k_0},g^{k_1})]$, where *m* would be EPC or pseudo-EPC in our protocol. We recommend using pseudo-EPC rather than EPC (See more details in [7]).
- *Decryption* : *P* and *B* decrypt C to get the *m*. First *P* and *B* check $\alpha_0,\beta_0,\alpha_1,\beta_1 \in \mathcal{G}$. If this check fails, then decryption fails. Second, they compute $m_0=(\alpha_0/\beta_0^x)$ and $m_1=(\alpha_1/\beta_1^x)$ using *T*'s *SK x*. If $m_1=1$, then decrypted message is $m=m_0$. Otherwise, decryption fails.
- *Re-encryption* : Anyone, which includes *P*, *B*, dishonest *R*, honest *R*, and the adversary, can re-encrypt C to $C'=[(\alpha'_0,\beta'_0);(\alpha'_1,\beta'_1)]=[(\alpha_0\alpha_1^{k'_0},\beta_0\beta_1^{k'_0});(\alpha_1^{k'_1},\beta_1^{k'_1})]$ with random re-encryption factor $r' \in Z_q^2$, where $k'_0,k'_1 \in Z_q$.

2. Security Analysis

Golle *et al.*[3] define the first vulnerability when re-encryption is used for RFID. Saito *et al.*[5] (SAITO) pointed out the second and third vulnerabilities, and then found two solutions on the first and second vulnerabilities that mentioned in each paper. However, SAITO fails to find a solution on the third vulnerability.

The first attack is as follows: if $C'=[(\alpha_0,\beta_0);(\alpha_1,\beta_1)]=[(\alpha'_0,\beta'_0);(1,1)]$ is written into *T*, C' is same with *C* that is, *T* is completely traceable since *T*'s response *C* is static.

The second attack is as follows: if the adversary writes C like

$$C_A=[(\alpha_0,\beta_0);(\alpha_1,\beta_1)]=[(m_A y_A,g_A);(y_A,g_A)]$$

encrypted with the adversary's *PK* $y_A = g_A^{x_A}$, and then the adversary can decrypt C'_A with the adversary's *SK* *x* no matter how many times authorized parties re-encrypt C.

The third attack is as follows: second and third attacks exploit (α_1, β_1) although the plaintext is included in α_0, β_0 $(m_0 = \alpha_0/\beta_0^x)$. The adversary can change *m* with different one using α_1, β_1 of C previously sent by *T* or an authorized *R* not to exploit the second and third attacks. For example, the decrypted plaintext can be m_A if the adversary writes C like

$$C' = [(\alpha_0, \beta_0); (\alpha_1, \beta_1)] = [(m_A, 1); (\alpha_A, \beta_A)].$$

In other words, *T* attacked by the third one is useless.

III. Our Protocol

We propose an off-tag access control mechanism using an external device or a proxy. Off-tag access control provides chance to be widespread with low-cost tags because the external device takes care of almost all of the computations instead of *T*.

T checks the first attack and second attack by itself in SAITO's first protocol; it is one of the on-tag access control schemes. Exponential computation is needed to check the second attack; however, it is big overhead for *T*. SAITO's first protocol checks only the contents written in *T* not to authenticate *R* that is, anybody can get *T*'s information from *B* upon receiving C from *T*. In contrast, we authenticate *R* let alone message contents *C*.

1. Initialization

The Proxy has *ID*, *SK* *x* of each *T*, and access control list which consists of four fields: action, source, target, boolean variable valid C. Example of access control list is described in Table 3.

Table 3. Example of Access Control List

Action	Source	Target	Valid C
pass *x*	an authorized party	an authorized party	Yes
write	any party	owner's tag	No
write	an authorized party	owner's tag	Yes or No

T has random number generator and memory storage to store C based on ElGamal encryption algorithm. Any other cryptographic primitives like hash or symmetric or asymmetric algorithm do not need.

2. Description

Our protocol checks three types of the attacks using *P* for personal use. *P* rewrites a new valid C' to *T* *w*hen one of the three attacks is found by *P* which is around *T*. The checking procedure of valid C of *T* performed by *P* is in Figure 1.

```
if $\alpha_1, \beta_1$ = 1 or -1, //for preventing the first attack
then $P$ writes valid $C'$ to $T$ and abort.

$m_1 \leftarrow (\alpha_1/\beta_1^x)$
else if $m_1 != 1$, // for preventing the second attack
then $P$ write valid $C'$ to $T$ and abort.

$m_0 \leftarrow (\alpha_0/\beta_0^x)$
else if $m_0 != m$, // for preventing the third attack
then $P$ write valid $C'$ to $T$ and abort.
```

Figure 1. Checking Algorithm in *P*

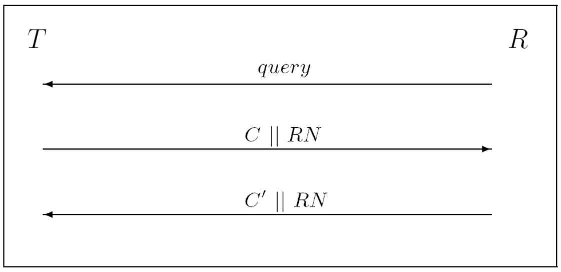

Figure 2. Our protocol between *R* and *T*

Our protocol procedure works as follows.

1. *R* sends query to *T*.
2. *T* generates random number *RN*, and then responds *RN* || $(C = [(\alpha_0, \beta_0); (\alpha_1, \beta_1)])$ to *R*. *R* has to be within backward channel to write C'; *R* can not write C' to *T* within forward channel

since *R* does not have the way to learn *RN*.

3. When *R* tries to write C into *T*, *R* selects random encryption factor $r' = (k'_0, k'_1) \in Z_q^2$, and then re-encrypts C to $C' = [(\alpha'_0, \beta'_0); (\alpha'_1, \beta'_1)] = [(\alpha_0, \alpha_1^{k'_0}); (\beta_0, \beta_1^{k'_0}); (\alpha_1^{k'_1}, \beta_1^{k'_1})]$. Lastly, *R* emits C' to *T*.
4. *T* checks whether the *RN* value received is the same as the *RN* value sent. if two values are same, *T* writes C' into its memory. Otherwise, *T* does not.
5. *P* checks three possible invalid C'. If it is invalid, *P* produces valid C' and then *P* writes C' into *T*'s memory. As another way, *P* can write C' whenever *R* tries to write C' without validation of C', which depends on owner of *P*.
6. *R* sends its information like certificate to be authenticated by *P*.
7. *P* authenticates R using the access control list. If *R* is an authorized party, *P* sends *SK x*, *m* and nonce to *R* encrypted with corresponding *B*'s *PK* as encrypted form like $E_{PK_B}(x \| m \| nonce)$. Lastly, *R* forwards $E_{PK_B}(x \| m \| nonce)$ to the *B*. *B*'s *PK* and *m* is used to reduce *B*'s computational time; nonce is used to be expired after being used only one time.

Our main idea is to use *P* exploiting UR security properties. *P* re-writes valid C' if *P* finds out invalid C'.

3. How the proxy works

Our *P* is used for personal use like RFID Guardian(GUARDIAN) [5] *P* which can be integrated into cellular phones or PDAs (Personal Digital Assistants) manages owner's *T*; *P* also enforces privacy policy desired by its owner using access control list. In our proposed protocol, *P* should exist around his own *T*.

Our *P* has seven functional security properties which are described in Figure 3; these properties in our protocol are a little different with REP [1] and GUARDIAN. The description of each component is as follows.

Tag acquisition ⟶ Key management ⟶ Checking C ⟶ Relabeling ⟶ Access control ⟶ Authentication ⟶ Tag release

Figure 3. *P*'s process

- *Tag acquisition* : *P* gets a new *SK* corresponding to the *PK* from *TA* and *T*'s *ID* from *T*'s owner (*P* or any other devices maintain *T*). *P* produces *C*, and then writes C into acquired *T*'s memory when *P* acquires *T*.
- *Key management* : *P* manages *T*'s *ID* and *SK*. *P* updates records in a database when acquire a new *T* or release his *T*.
- *Checking C* : *P* checks C as the following checking procedure in Figure 1.
- *Relabeling* : *P* relabels *T* contents if invalid C is found.
- *Access control* : Access control considers three cases: which *R*, which *T*, which circumstances like GUARDIAN.
- *Authentication* : *P* checks whether queried *R* are an authorized party or an unauthorized party. If an authorized party sent query, then *P* passes *SK x* with encrypted form to *R* using *PK* and nonce.
- *Tag release* : When *T*'s owner does not want to keep his *T* any more, owner releases *T*.

IV. Security and Performance Analysis

In this section, we check whether our protocol guarantees security requirements as followings: ownership transfer, scalability, privacy, protection against several threats which are desyncronization, spoofing, swapping, replay, and DoS.

- *Ownership transfer.* We described the way to provide ownership transfer in tag acquisition of Section 3.3.
- *Scalability.* Scalability means that *B* is required

to find *T*'s *ID* with constant computational time regardless of the number of tags. An authorized *R* which sent query gets encrypted *x* using corresponding *PK* of *B* from *P*, and then *R* forwards the message received from *P* and C to the *B*. *B* decrypts encrypted *x* with its *SK*, and then *B* decrypts C with *x* to get *T*'s *ID*. The complexity of this process is *O(1)* that is, *B* does not need computations related to non-relevant *T*.

- *Privacy.* Privacy means that no information has to leak from the *T*. We guarantees privacy since our protocol is provably secure since it is based on UR [2] . Pseudo-EPC as *T*'s *ID* should be used to provide privacy. (See more details in [7])
- *Protection against tracing and desynchronization.* Relabeling *T*'s *ID* was introduced to prevent tracing. Synchronization between parties who share secret information is needed after relabeling the shared secret. In our protocol, we eliminated entirely desynchronization problem because we get *T*'s *ID* after decrypting once regardless of the number of re-encryption times. *T*'s response is different every time when *R* requests; however, our protocol is traceable provided that the adversary collects all the messages between *P (*or *R)* and *T* within the range which *P* transmits C to *T* in.
- *Protection against cloning and spoofing.* Cloning and Spoofing *T* are meaningless since *P* maintains *SK x*.
- *Protection against swapping.* In our protocol, swapping attack is possible. Swapping attack is vulnerability on UR. Protection against swapping is one of the open problems.
- *Protection against replay.* Replay attack is meaningless since anybody can write C into *T* within backward channel.
- *Protection against DoS.* DoS attack can cause battery waste of *P*, which is the one of main problems when using the mobile device.
- *Cost.* *T* requires only one lightweight cryptographic primitive, random number generator, and re-writable memory. Consequently, our protocol can be implemented with reasonable low-cost.

V. Comparison with Related Work

Selective RFID jamming [4] makes jamming signal jam up the airwaves under lots of an unauthorized *R*'s queries while an external device just re-encrypts a new valid C in our protocol.

REP and GUARDIAN send *T*'s secret value with unencrypted form, which is insecure since REP and GUARDIAN give the adversary chance to eavesdrop while Our *P* does not reveal *T*'s secret information.

SAITO has several weaknesses: big overhead on *T*, tracking with only eavesdropping within forward channel, no *R* authentication mechanism, allowing re-encryption with a different message and swapping attack. We solve three former problems using *P*. However, vulnerability on swapping attack is effective in SAITO and our protocol.

VI. Concluding Remarks

Our protocol is different approach with previous *P* [1,5] ; *P* is compact, powerful device, used for personal usage, and around individual person in RFID-tagged environments.

In this paper, we propose one of the RFID off-tag access control mechanisms based on UR and *P* has seven security properties: tag acquisition, key management, checking C, relabeling, access control, authentication, tag release; moreover, *P* provides individually chance to enforce security policy.

Our proxy-based protocol exploiting the UR algorithm has several security properties as followings: synchronization, ownership transfer, scalability, privacy, protection against several attacks which are spoofing, replay, and cloning. In addition, *R* and *T* do not need to share secret keys, and so we sure that our protocol can contribute widespread

RFID deployment since ours can be applied to EPCglobal Class-1 Generation-2 [2] UHF tag standard which is the *de facto* worldwide specication for inexpensive RFID tags. On the other hand, our protocol can be traceable if the adversary collects all the messages; the other vulnerabilities are that swapping, DoS attack. As a future work, we are going to suggest the solution of these vulnerabilities.

References

[1] Ari Juels, Paul Syverson and Dan Bailey, "High-power Proxies for Enhancing RFID Privacy and Utility", *Workshop on Privacy Enhancing Technologies PET 2005*, May-Jun. 2005, Dubrovnik, Croatia.

[2] EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz-960 MHz Version 1.1.0 Draft 1.

[3] Philippe Golle, Markus Jakobsson, Ari Juels and Paul Syverson. "Universal Re-encryption for Mixnets", *The Cryptographers' Track at the RSA Conference CTRSA*, LNCS 2964, Feb. 2004, San Francisco, California, USA.

[4] Melanie R. Rieback, Bruno Crispo and Andrew S. Tanenbaum, "Keep on Blockin' in the Free World: Personal Access Control for Low-Cost RFID Tags", *International Workshop on Security Protocols IWSP'05*, Apr. 2006, Cambridge, England.

[5] Melanie R. Rieback, Bruno Crispo, and Andrew S. Tanenbaum, "RFID Guardian: A Battery-Powered Mobile Device for RFID Privacy Management", *Australasian Conference on Information Security and Privacy ACISP'05*, LNCS 3574, Jul. 2005, Brisbane, Australia.

[6] Junichiro Saito, Jae-Cheol Ryou and Kouichi Sakurai, "Enhancing Privacy of Universal Re-encryption Scheme for RFID Tags", *Embedded and Ubiquitous Computing EUC 2004*, LNCS 3207, Aug. 2004, Aizu-Wakamatsu City, Japan.

[7] Youngjoon Seo, Hyunrok Lee and Kwangjo Kim, "Scalable and Untraceable Authentication Protocol for RFID", *The Second International Workshop on Security Ubiquitous Computing Systems SECUBIQ 2006*, LNCS 4096, Aug. 2006, Seoul, Korea.