Mobile RFID Mutual Authentication and Ownership Transfer

Ming Hour Yang
Information Computer Science
Chung Yuan Christian University
mhyang@cycu.edu.tw

Jia-Ning Luo
Information and Telecommunication
Ming Chuan University
deer@mail.mcu.edu.tw

***Abstract* — In this paper, we propose an ownership transfer scheme that applies in mobile RFID networks. The scheme includes a mutual authentication protocol and a role-based ownership transfer protocol. A tag will decide what actions are allowed for a reader according to the reader's role class, and the back-end server will send to the reader the requested information about the tag. Keyed-hash functions are used to secure the protocols. Last, we prove that our protocol can do against the threats of replay attacks, distributed denial of service (DDoS), Man-in-the-Middle (MITM) attacks that change users' data, interception of data and location privacy, and tracking of tags' ownership transfer.**

Keywords-RFID;authentication; ownership transfer

I. INTRODUCTION

RFID features mass identification, large data size, modifiable identification and data, and effective scanning of tags by batch processing at long distance. Nowadays, mobile RFID [1][11] integrating reading chips, passive RFID tags and mobile phones enables users to access information. Mobile RFID can be applied in business transaction; through the transfer of tagged products' ownership, each transaction can be done with mobile RFID. The transfer of a tagged product's ownership suggests whoever is registered in the tag is the one entitled to the item.

To protect the privacy of both the former and current owners of a tagged item, RFID protocol designers have to make sure that when the item's ownership changes, its tag's ownership has to change accordingly and simultaneously. Former owners, therefore, will no longer be able to access the tag, whereas the current owners have no way to track the privacy history that was kept in the tag, either.

Due to the limitation of tags, there are only 2000 logic gates in a passive tag to do security functions [4][7]. In 2006, John Ayoade[5] proposed an authentication-control framework, creating a table on back-end authentication server (AS) to control the reader-tag authentication. When a reader accesses a tag, the tag will send out its identifier and encrypted messages to the reader, and the reader sends a reading request to the AS. The AS checks the reader's identity and gives a key to the reader to decrypt the message and grant the ownership of the tag.

The authorization and ownership transfer process, the delegation, should be done securely to protect the owner and the tags [3][8][9][10]. If the delegation process is incomplete, the former owner could still access the tag [2]. Fouladgar proposed a delegation protocol to deal with incomplete ownership transfer [8][9][10]. In the protocol, the delegated reader can verify the digital certificate of a current owner's reader through a certificate authority (CA) during the ownership transfer process, and the key stored in the tag is updated by the AS to ensure only the current owner can access the tag.

Although delegated readers reduce the computation load of the AS, the reader's computation resources such as CPU and memory are limited. When a reader has too many delegated tags, it can no longer afford the authentication task because it does not have enough memory to keep tags' information. Fouladgar's protocol uses counters to limit delegated readers seemed to fail to take good control of reading limits.

When malicious users sent a large number of queries to the tags, the tags will keep asking AS to update the keys. If the update message was lost or abandoned by attackers, Foudladgar's protocol will fail and the owner's reader will lost the tag. To prevent this kind of DoS attack, Osaka[6] proposed another ownership transfer scheme. In Osaka's scheme, the tag confirms the ownership transfer is completed with AS in every session. However, in Osaka's scheme, a reader should have large memory to keep the tags' keys, and it is suffer from man in the middle attack.

In this paper, we propose a protocol for ownership transfer and reader-tag mutual authentication in a mobile RFID environment. Unlike traditional RFID, mobile readers are usually put under the presupposition that they might be malicious devices and their communication with back-end server is not secured. In our protocol, the ownership of a tag is transferred to the new reader by updating the tag's key after a mutual authentication process between the read and the tag. Our protocol can not only reduce tags' computational load effectively but also allow readers to access tags without storing any shared keys. Furthermore, our protocol provides location privacy, data privacy and forward security. Our protocol can prevent replay attacks, man in the middle attack, the DoS attack, and protects the tag location and the ownership transfer history.

This paper is organized as follows: in the next section, we proposed a Mobile Access Control and

 ISBN:978-1-61208-114-4

Ownership Transfer protocol (MACOT) to deal with mutual authentication and ownership transfer in mobile RFID environment. Section 3 deals with the security analysis of our protocol. Section 4 analyzes the protocol's performance. Conclusion is drawn in the Section 5.

II. MOBILE ACCESS CONTROL AND OWNERSHIP TRANSFER

In this section, we propose a Mobile Access Control and Ownership Transfer (MACOT) protocol to deal with mutual authentication and ownership transfer in mobile RFID. Our ownership transfer scheme consists of three stages. The first one is mobile mutual authentication procedure (MMAP), the second ownership transfer procedure (OTP), and the third RC-Action Table update procedure.

The mutual authentication protocol requires the readers obtain the corresponding information of a tag from back-end authentication server according to reader's authority. The ownership transfer protocol updates the tag's key with the authorized owner after the mutual authentication process. And the RC-Action table update procedure is used by a current owner to grant control of a tag.

A. Preliminary

With the high mobility, a mobile reader has wide-range accessibility. Subsequently, tags within its access range could probably belong to a different authority. An authentication scheme is required to identify tags and locates their corresponding back-end servers. According to H. Lee and J. Kim's mobile RFID infrastructure [11], the authentication process with 7 steps is shown in Figure 1. :

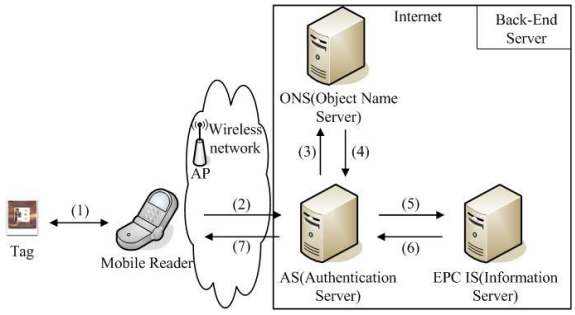

Figure 1. Mobile RFID infrastructure

Step 1. A mobile reader sends a reading request to the tag, and gets a responding message from it.
Step 2. The reader forwards the message to AS to verify the identity of the tag.
Step 3. AS verifies tag's identity and queries Object Name Server (ONS) to get the detailed information of the tag.
Step 4. ONS sends the tag's URL of EPC IS to AS.
Step 5. According to the URL, AS requests the tag's information from EPC IS, which is the back-end database of tags.
Step 6. EPC IS sends the tag's information to AS.
Step 7. AS sends the tag's information to the reader.

Because a passive tag's computation resources is limited, the packets uses a keyed-hash function $h_x()$, generated with the key x shared by the tag and the authentication server to prevent eavesdropping. The traffic between the readers and the AS is protected by traditional symmetric encryption algorithm $E_k()$. Back-end server, including AS, ONS and EPC IS, are trusted by tags and readers.

To manage a reader's authority over a specific tag, AS and the tag must store the authorizing information. Figure 2(a) stored the corresponding actions of readers of different role classes (RCs) to the tag *TID* 80 in a RC-Action table. In the table, the tag owner has the highest privileges to modify the actions of each RC. The role in an upper row has higher privileges. As shown in Figure 2(a), readers with an owner-level RC are entitled to Action 3, which means they are also authorized to do Action 1 and Action 2. In addition, the relevant information of tags is stored in different EPC ISs, as in Figure 2(b):

(1) Readers' Access Control List: each row indicates each reader's RC class. For example, the reader *RID* 312's authority over *TID* 80 is B-class RC.
(2) Action Table: the AS decides what command could be send from the reader to the tag. For example, the reader with *RID* 312 can access *TID* 80's public (general) data and private (personal) data.

RC-Action Table for TID 80

RC	Action
A	1
B	2
C	2
Owner	3

(a) Information Table in Tag

Reader's Access Control List

TID / RID	79	80	730
214	A	A	
312		B	Owner
666	A	C	A

Action Table

TID	Action	Command	Data
79	1	Read_pubilc	
	2	Ownership transfer	
80	1	Read_pubilc	
	2	Read_private	
	3	Ownership transfer	
730	1	Read_pubilc	
	2	Read_private	
	3	Ownership transfer	

(b) Information Table in Back-End Server

Figure 2. Data stored in (a) tag (b) back-end server

We assume all readers are not trusted and they do not need to store any tags' keys. In the initialization stage, the keys and secret of back-end server, readers and tags are shown in Figure 3. The

 ISBN:978-1-61208-114-4

server stored each tag's identifier *TID*, two shared keys K_x and K_y between tags and the server, a *PIN* shared with a tag and the owner's reader, and a shared secret *C*.

The tag's owner, the reader, which owns a tag, stored the *TID* of the tag, and it's *PIN* and *C* values. In each tag's memory, stored the K_x, K_y, *PIN* and *C*.

Tag's Information Table

TID	K_x	K_y	PIN_i	C
79	99	96	94	90
80	11	22	33	44
730	55	66	77	88

(a) Keys and Secret Values Stored in the Server

Tag Owner of TID 730

PIN	77
C	88

(b) Keys and Secret Values Stored in the Reader

TID	730
K_x	55
K_y	66
PIN	77
C	88

(c) Keys and Secret Values Stored in the Tag

Figure 3. Shared keys and secret values stored in (a) Server (b) Reader (c) Tag

B. *Mobile Mutual Authentication Procedure (MMAP)*

We assume the back-end server can verity the reader's identity and exchange a session key K_{dr} between them. When a reader read a tag, the Mobile Mutual Authentication Procedure (MMAP) is preforms with 8 steps:

Step 1. When a reader queries a tag, the tag generates a random number r_1 and creates a secret value *S* by XOR-ing r_1 and its own identifier *TID*, and computes $S = h_{K_x}(TID \oplus r_1)$. The tag sends *S* and r_1 to the reader.

Step 2. The reader generates another random number r_2, and sends $E_{k_{dr}}(S, r_1, r_2, RID, Command)$ to the server.

Step 3. The server decrypts the message with K_{dr} and computes $h_{K_x}(TID \oplus r_1)$ for all tags to obtain *TID*.

Step 4. The server looks up Reader's Access Control List to find out the reader's RC, access level, and generate a random number r_3. It computes $T = h_{K_y}(TID \oplus r_1 \oplus r_2 \oplus r_3, RC, Command)$ and $p = h_{K_y}(r_3 \oplus TID)$. The server encrypts *T*, *p*, r_2 and r_3 with a session key K_{dr}, i.e. $E_{K_{dr}}(T, p, r_2, r_3)$, and sends the result to the reader.

Step 5. The reader decrypts the message with the session key K_{dr} and verifies r_2. If it's correct, the reader forwards *T*, r_2 and r_3 to the tag.

Step 6. The tag verifies T by searching all the possible values of RC and commands. It computes $p = h_{K_y}(r_3 \oplus TID)$, $G = h_{K_x}(TID \oplus r_3 \oplus Act)$ and sends them to the reader.

Step 7. The reader verifies *p*, and forwards G and r3 to the server by computing $E_{K_{dr}}(G, r_3)$.

Step 8. The server verifies *G* to find a matched *Act*, and searches the action table in Figure 2. to find a matched *Command*. If the *Command* matches the *Act*, the reader is authorized, and the server sends the requested tag's data to the reader.

The complete mutual authentication protocol is illustrated in Figure 4:

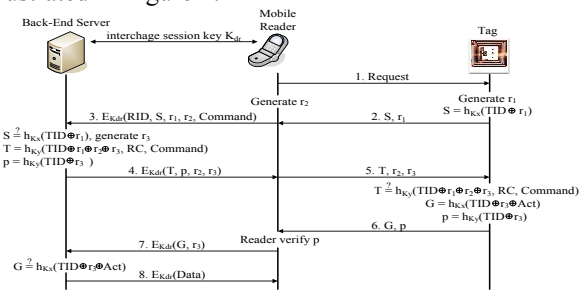

Figure 4. Mobile Mutual Authentication Procedure

C. *Ownership Transfer Procedure (OTP)*

After the server, the reader and the tag authenticate themselves to each other, the Ownership Transfer Procedure (OTP) is performed to transfer the ownership between the former owner and the current owner, as shown in Figure 5. The two owners should authenticate each other through a trust third party before perform the OTP.

The OTP is divided into two parts:

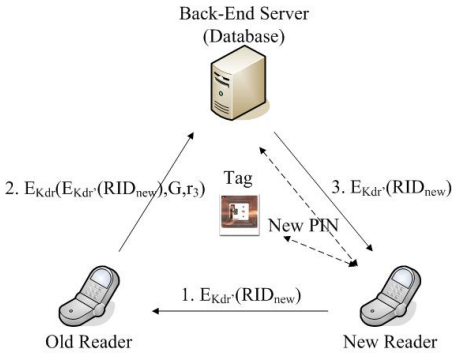

Figure 5. Diagram of Ownership Transfer

1) Part 1

In Part 1, the server authenticates the former and current owners and tag. The two owners have to exchange session keys K_{dr} and K_{dr}' with the server. Next, the current owner encrypts his identifier RID_{new} with K_{dr}' and send it to the former one. The former owner uses the MMAP protocol to authenticate him with the tag, as shown in messages 2-7 of Figure 6. . The former owner encrypts $E_{Kdr'}(RID_{new})$, r_3 and G to back-end server in message 8. The server adds the current owner RID_{new} into the Access Control List of the current owner's reader into the tag and marks RID_{new}'s RC as Owner.

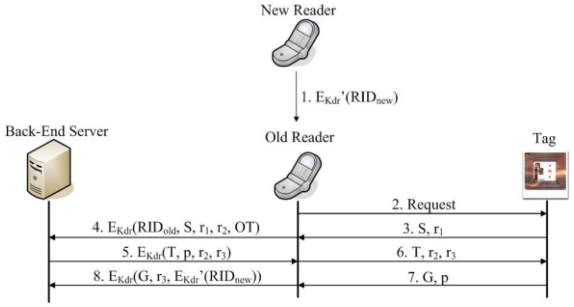

Figure 6. The First Part of OTP

2) Part 2

The second part of the OTP is shown in Figure 7. After receiving $E_{K_{dr}'}(RID_{new})$ from the server, the current owner verifies the derived RID_{new}. The current owner uses the MMAP protocol to authenticate him with the tag, as shown in messages 2-6 in Figure 7. The rest of the protocol (steps 7-13) is as follows:

Step 7. The tag generates $G' = h_{K_x}(TID \oplus r_3 \oplus Act, PIN_i)$ and sends it to the server via the reader.

Step 8. The reader encrypted G' with the random number r_3', and sends it to the server.

Step 9. The server verifies G' and updates the tag's PIN and secret value C. the server computes $h_{PIN_i}(r_1', C)$, and sends it to the reader

Step 10. The reader forward the message to the tag.

Step 11. The tag verifies r_1' and C, and generates a new $PIN_{i+1} = h_{K_x}(PIN_i \oplus K_y, r_3')$ and $C' = h_{K_x}(C \oplus K_y, r_3')$. The tag computes $h_{PIN_{i+1}}(r_3', C')$ with the new PIN and C, and sends it to the reader.

Step 12. The reader forwards the message to the server. The server uses the same function to generate the new PIN and C and verifies $h_{PIN_{i+1}}(r_3', C')$. If the comparison is the same, the server modifies the reader's Access Control List to change or delete the former owner's tag identifier and reader's RC.

Step 13. The server sends the new PIN and C to the current owner.

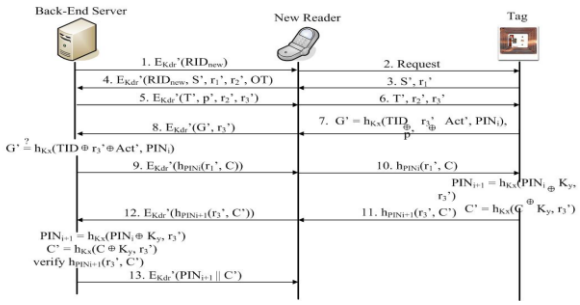

Figure 7. The Second Part of OTP

The missing of message 10 and 11 could lead to asynchronous update of data between back-end server and the tag. Our protocol is designed to tackle such asynchrony in OTP and requires that the reader re-access the tag after it sends G' to the server. Meanwhile, the server uses PIN_i for computation to generate G' and check if this is the same as the G' from the reader. If they are different, the server will compute again with other secret values to generate PIN_{i+1} and re-queries G'. If the two G's are the same, it means PIN_{i+1} is the key of the tag and it has updated its key. Therefore, the server no longer needs to update the tag, and will send PIN_{i+1} and C' to the current owner directly. If the G' that the server generates with PIN_i is identical to the one from the reader, the tag has missed message 10 in the communication and has not yet updated PIN_i. Consequently, back-end server begins to generate $h_{PINi}(r_1', C)$ and update the tag's key with it. If the tag returns $h_{PINi+1}(r_3', C')$, the update has been completed. After verification, the server will send PIN_{i+1} and C' to the current owner. The procedure is illustrated as below:

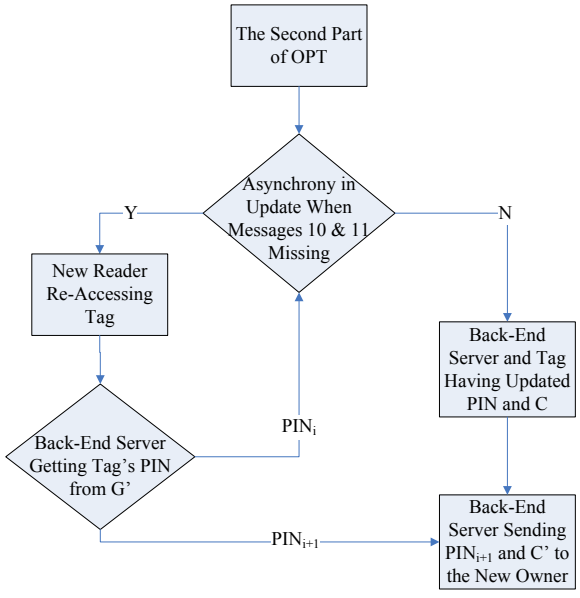

Figure 8. Diagram of OTP When Messages Missing

In mobile RFID, tag owners can transfer a tag's ownership to others through OTP. The following is an instance to exemplify OTP. Figure 2 outlines the initiation stage, the owner of reader *RID* 312 transferring his ownership over the tag *TID* 730 to a current owner of reader *RID* 666. First, mutual authentication is achieved between back-end server and *RID* 312. Then the server looks up the reader's Access Control List (ACL) and confirms *RID* 312's RC to *TID* 730 is Owner. Next, the server generates *p* and *T* before sending them to *RID* 666 and *TID* 730 respectively. For the server now, *RID* 666's RC to *TID* 730 has been updated as Owner, as the ACL in Figure 9 (b) indicates. Following these steps, the server will begin its mutual authentication with *RID* 666 and accordingly verifies it as the data receiver of *TID* 730. Subsequently, the server encrypts *T* and *p* with a key shared with *RID* 666 and then sends the encrypted *T* and *p* to *RID* 666. Now, the mutual authentication between *TID* 730 and *RID* 666 must be achieved before *TID* 730 generates *G'* with its *PIN* 77 and sends it to the server. Next, the server generates new *PIN* with 77 (see Figure 3 (b)) and a new shared secret *C* with the old one 88 (see Figure 3 (b)), and sends them to *RID* 666 and *TID* 730 respectively. As a result, both the tag and back-end server update *TID* 730's *PIN* and *C* in their own information tables, as illustrated in the highlighted cells of Figure 9. Ownership, therefore, is transferred to the reader *RID* 666.

TID	730
K_x	55
K_y	66
PIN	$h_{55}(77 \oplus 66,15)$
C	$h_{55}(88 \oplus 66,15)$

(a) Table in Tag

Table for Tags' Information

TID	K_x	K_y	PIN_i	C
79	99	96	94	90
80	11	22	33	44
730	55	66	$h_{55}(77 \oplus 66,15)$	$h_{55}(88 \oplus 66,15)$

Reader's Access Control List

TID / RID	56	80	730
214	A	A	
312		B	A
666	A	C	Owner

(b) Tables in Back-End Server

Figure 9. After Ownership Transfer, Tables in (a) Tag (b) Back-End Server

D. RC-Action Table Update Procedure

As the ownership is transferred, the current owner, with the *PIN* and *C* from back-end server, is able to renew the tag's RC-Action Table to set the allowed actions for other readers. The steps are as follows:

Step 1. The current owner's reader generates a random number r_1, sends it to the tag and begins to update the RC-Action Table. The tag receives r_1 and generates r_2 and then XORs them. Further, it computes a hash with *PIN* and *C* before sending it to the reader.

Step 2. The reader queries the hash function. If it is valid, this message is sent by a legal tag. Next, the reader uses r_2 and *C* to compute a hash with *PIN* $h_{PIN}(r_2, C)$ before sending it to the tag.

Step 3. The tag queries $h_{PIN}(r_2, C)$. If it is not valid, the reader does not belong to the owner. If valid, then the reader does. Next, the tag puts the values of *RC* and its corresponding *Act* into a keyed-hash function for computation one row after another before sending it to the owner. Receiving the message, the owner computes the hash values one after another and therefore is able to restore the RC-Action Table. r_2 here is used to prevent MITM attacks on one hand, and for the owner to query whether the message is sent by the tag on the other.

Step 4. The owner XORs the *RC* and its corresponding *Act* in the RC-Action Table one row after another, then put each of them into a keyed-hash function with r_1 for computation, and finally sends them to the tag. In addition, the tag computes the hash values one by one and updates its RC-Action Table.

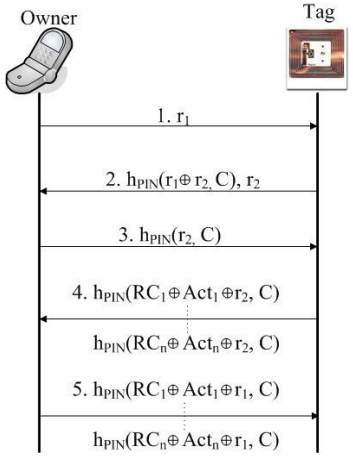

Figure 10. RC-Action Table Update Procedure

Since a tag owner can modify its RC-Action Table at will, we will take the following example to show how an owner updates a tag's RC-Action Table, enabling the RC-A users to access private data. Its initial state is shown in Figure 11 (a) and the tag's identifier *TID* is 730. After the reader-tag mutual authentication, *TID* 730 verifies this reader as Owner. Now, the reader is able to modify *TID* 730's RC-Action Table, updating RC-A's Action from 1 to 2. That is to say, users authorized as RC-A can access not only public data but also private one.

 ISBN:978-1-61208-114-4

RC-Action Table

RC	Action
A	1
B	2
C	2
Owner	3

(a) Before Update

RC-Action Table

RC	Action
A	2
B	2
C	2
Owner	3

(b) After Update

Figure 11. TID 730's RC-Action Table (a) Before Update (b) After Update

By controlling the RC-Action Table, a tag owner is also able to decide what level of data is accessible to what readers, according to their RCs. Thus, readers with lower authority is not entitled to the data that requires high authority, while readers with higher authority can fully access the tag at will.

III. SECURITY ANALYSIS

In this section, we will prove that our OTP is able to secure ownership transfer against replay attacks, DoS from asynchronous update and MITM attacks that change messages; to achieve mutual authentication; and to protect the privacy of tags' data and location, even though the valid readers have been attacked. Since we have assumed that the communication between a reader and AS is secured, we will just focus on the security of the reader and tag.

A. Against MITM Attacks' Modification of Messages

In our protocol, messages between a reader and tag are protected by keyed-hash functions. For instance, a tag generates S and sends it to the back-end server. The server uses TID and K_x, shared with the tag, to verify S.

B. Against DoS from Asynchronous Update

We use PIN to synchronously update the keys and secret values between a tag and back-end server, that is included in G', as sent by the tag in the message 7 in Figure 7, is used for the server to verify a tag's keys. If message 11 is abandoned by malicious users, which could lead to only a tag's update of PIN and C unilaterally, the server can derive PIN_{i+1} from PIN_i found in this tag's information table and from G' sent by the tag so as to query whether the key PIN_{i+1} is exactly identical to that stored in the tag. This scheme can, therefore, prevent DoS attacks that result from asynchronous update of keys.

C. Against Replay Attacks

As every query between a tag and reader carries a random number in each session, attackers are not able to launch replay attacks by simply coping the last verified message and resending it to back-end server. Our authentication scheme will fail their attempts in this style. For example, if attackers resend to a tag a verified message that contains the value T consisting of r_1 generated by the tag, such as the message 6 in Figure 7. , the tag will query T with current r_1 in the current session. If the two are different, the authentication procedure will not go further and attackers cannot access any data from the tag, either.

D. Security of the Data Privacy of Tags

We secure the messages between a reader and tag with keyed-hash functions h_{Kx} and h_{Ky}.
If attackers launch replay attacks or try interception, they can only get hashed values sent by a tag, e.g. $h_{Kx}(TID \oplus r_1)$. They cannot obtain a tag's identifier TID from those hashed values. Thus, the privacy of tags' data is secured.

E. Security of the Location Privacy of Readers and Tags

Normally, if attackers record a couple of messages between a reader and tag, they can probably find the connection in these messages and accordingly are able to track the location of the reader and tag. In our OTP, a tag sends out three messages, i.e. S' in the message 3, G' and p' in the message 7 and $h_{PINi+1}(r_3', C')$ in the message 11, as illustrated in Figure 7. Because the three messages all contain random numbers, their results change in every session. In doing so, attackers can no longer track a tag's location from these messages and its location privacy is secured. Similarly, the messages 6 and 7 in Figure 7, which are forwarded to back-end server by a reader, also change in every session because of the random numbers that the three messages (3, 7 and 11) carry along. Consequently, attackers cannot find the connection between these messages that the reader forwards and track its location.

F. Security of Ownership Transfer

To secure ownership transfer, back-end server sends and updates a tag's secret values PIN and C via the current owner, who then encrypts them with a symmetric key K_x. Because of the keyed-encryption, owners' privacy is protected and the former owner can no longer modify a tag's RC-Action Table with the old PIN and C. Therefore, with the deprivation of former owners' access authority and the protection from the threats mentioned above, we can say the ownership transfer is secured.

IV. PERFORMANCE

The performance of our schemes will be analyzed in this section and their results will be illustrated in detail in Table 1. T_H represents the time that a hash function takes in one computation; T_{XOR}, the time that an XOR takes in one computation; T_{RNG}, the time it takes to generate a random number; N, the total tags that back-end server stores; L, the levels of a RC; M, the actions of a tag; P, the actions that a tag is entitled to.

 ISBN:978-1-61208-114-4

TABLE I. PERFORMANCE OF TAG, READER AND BACK-END SERVER IN EACH SCHEME

	Mobile Mutual Authentication	OTP	Update of RC-Action Table
Tag	1 T_{RNG} + 7 T_{XOR} + (LM+3)T_H	1 T_{RNG} + 9 T_{XOR} + (LM+7)T_H	1 T_{RNG} + (1+2LP+2L) T_{XOR} + (2+LP+L)T_H
Reader	1 T_{RNG}	1 T_{RNG}	1 T_{RNG} + (1+2L+2LP) T_{XOR} + (2+L+LP)T_H
Back-End Server	1 T_{RNG} + (N+P+5) T_{XOR} + (N+P+2)T_H	1 T_{RNG} + (N+P+7) T_{XOR} + (N+P+6)T_H	No Time

Table 1 indicates that the performance of the three items (tag, reader and back-end server) is based on the numbers that users design for L, M and P, whereas a reader does not need to store any keys to access a tag, e.g. in mobile mutual authentication and ownership transfer, except in the update of RC-Action Table.

V. CONCLUSION

In the foreseeable future, RFID readers will not be confined by locations anymore. The combination of reading chips and mobile devices has made mobile readers come true and paved the way for the development of mobile RFID. However, security issues remain a pain for RFID engineers, traditional and mobile alike. As for mobile RFID, the security is even at more serious stake because malicious users might take unauthorized readers to access people's tags and this could endanger the privacy of users and their data. For this reason, we propose a mutual authentication scheme for mobile RFID, using back-end server to verify readers and then find out their RCs. Besides, we require that back-end server send RCs via readers so that a tag can always obtain the current reader's RC before being accessed. Tag owners subsequently look up the information tables stored in tags and decide what actions are allowed for a reader. Eventually, following a tag's final decision, back-end server sends to a reader the requested information of this tag. Apart from these, this scheme is also capable of ownership transfer by updating tags' keys. We use keyed-hash functions in the messages between tags and readers and therefore secure the tag-reader mutual authentication and ownership transfer against replay attacks, DoS from asynchronous update, MITM attacks' modification of messages and malicious users' tracking of tags' location and ownership transfer history, and, last but not least, enhance the privacy of tags' data and location.

ACKNOWLEDGEMENT

This work was supported by the National Science Council (NSC 99-2219-E-033-001 and NSC 99-2221-E-130-007), Republic of China.

REFERENCE

[1] M. H. Yang, "Lightweight authentication protocol for mobile RFID networks," *International Journal of Security and Networks*, vol.5, no.1, pp. 53-62, 2010.

[2] B. Toiruul and K. Lee, "An advanced mutual-authentication algorithm using AES for RFID systems," *International Journal of Computer Science and Network Security,* vol.6, no.9, pp. 156-162, September 2006.

[3] D. Molnar, A. Soppera, and D. Wagner, "A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags," *Selected Areas in Cryptography,* pp. 276-290, 2006.

[4] H. Y. Chien, "Secure access control schemes for RFID systems with anonymity," *Mobile Data Management, 2006. MDM 2006. 7th International Conference on,* pp. 96-96, 2006.

[5] J. Ayoade, "Security implications in RFID and authentication processing framework," *Computers & Security,* vol. 25, no.3, pp. 207-212, 2006.

[6] K. Osaka, T. Takagi, K. Yamazaki, and O. Takahashi, "An efficient and secure RFID security method with ownership transfer," *Proceedings of the 2006 International Conference on Computational Intelligence and Security,* vol. 2, pp. 1090-1095, 2006.

[7] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, "Security and privacy aspects of low-cost radio frequency identification systems," *The First International Conference on Security in Pervasive Computing,* pp. 201–212, March 2003, Revised Papers, 2004.

[8] S. Fouladgar, F. Evry, and H. Afifi, "An efficient delegation and transfer of ownership protocol for RFID tags," *Proceedings of the First International EURASIP Workshop on RFID Technology*, September 2007.

[9] S. Fouladgar and H. Afifi, "A simple delegation scheme for RFID systems (SiDeS)," *RFID, 2007. IEEE International Conference on,* pp. 1-6, 2007.

[10] S. Fouladgar and H. Afifi, "A simple privacy protecting scheme enabling delegation and ownership transfer for RFID tags," *Journal of Communications,* vol. 2, no. 6, pp. 6-13, 2007.

[11] N. Park, H. Lee, H. Kim and D. Won "A security and privacy enhanced protection scheme for secure 900MHz UHF RFID reader on mobile phone," *IEEE International Symposium on Consumer Electronics,* pp. 692–696, 2006.

 ISBN:978-1-61208-114-4