An Efficient and Private RFID Authentication Protocol Supporting Ownership Transfer

Süleyman Kardaş[1,2], Serkan Çelik[1,2], Atakan Arslan[1], and Albert Levi[2]

[1] TÜBITAK BILGEM UEKAE Gebze, Kocaeli
[2] Sabancı University, Faculty of Engineering and Natural Sciences, İstanbul, TR-34956, Turkey

Abstract. **R**adio **F**requency **ID**entification (RFID) systems are getting pervasively deployed in many daily life applications. But this increased usage of RFID systems brings some serious problems together, security and privacy. In some applications, ownership transfer of RFID labels is sine qua non need. Specifically, the owner of RFID tag might be required to change several times during its lifetime. Besides, after ownership transfer, the authentication protocol should also prevent the old owner to trace the tags and disallow the new owner to trace old transactions of the tags. On the other hand, while achieving privacy and security concerns, the computation complexity should be considered. In order to resolve these issues, numerous authentication protocols have been proposed in the literature. Many of them failed and their computation load on the server side is very high. Motivated by this need, we propose an RFID mutual authentication protocol to provide ownership transfer. In our protocol, the server needs only a constant-time complexity for identification when the tag and server are synchronized. In case of ownership transfer, our protocol preserves both old and new owners' privacy. Our protocol is backward untraceable against a strong adversary who compromise tag, and also forward untraceable under an assumption.

Keywords: RFID, Privacy, Security, Ownership Transfer Protocol.

1 Introduction

Today, ubiquitous information and communication technology has been widely accepted by everyone that aspire to reach information anytime and anywhere. Radio-frequency identification (RFID) systems are one of the ubiquitous computing in which technology provides practical services to people in their daily life. RFID technology aims to identify and track an item or a person by using radio waves. It has been pervasively deployed in several daily life applications such as contact-less credit cards, e-passports, ticketing systems, etc.

A RFID system basically consists of several tags *(transponders)*, a set of readers *(interrogator)* and a back-end receiver. A tag contains a microchip which carries data and antenna. It is interrogated by a reader via its modulated radio signals. A RFID reader that is the central part of an RFID system, acquires

G. Avoine and O. Kara (Eds.): LightSec 2013, LNCS 8162, pp. 130–141, 2013.

the data of the tag and conveys it to the back-end system for further processing. Moreover, RFID tags can be categorized into three groups by using energy source such as active, passive and semi-passive or battery assisted tags. Passive RFID tags do not have internal energy sources. Instead, they use the radio energy transmitted by the reader [10]. Furthermore, RFID systems can also be grouped into three basic ranges by their using operating frequency: Low frequency (LF, 30-300 KHz), high frequency (HF 3-30 MHz) and ultra high frequency (300 MHz - 3 GHz) / microwave (>3 GHz) [9].

Nowadays, the number of RFID applications have been proliferating because of their productivity, efficiency, reliability and so on. Many companies also prefer low-cost tags with tiny sizes. This brings some computational and memory restrictions to RFID tags. On the other hand, RFID tags and readers communicate with each other over an air interface. This insecure channel and the limited capabilities of RFID tags cause security and privacy vulnerabilities. An adversary can do tag impersonating, tracking, eavesdropping, and denial of service (DoS) attack. Besides the vulnerabilities, a tag might be distinguishable in its life-span by an attacker. If it is once recognized by an adversary, it can be easily traceable. At that situation, there might be two attacks. (i) An attacker might track the previous interactions of the tag or (ii) he may track the future ones. These two attacks are called backward traceability and forward traceability, respectively. The protocol used for RFID system should provide not only resistance against passive attacks, replay attacks, cloning attacks but also resistance against active attacks. There are public-key cryptography solutions in the literature but none of them are convenient for the low-cost tags used in lots of applications because of their limitations. It needs to find much light-weight approaches. Therefore, many light-weight authentication protocols are proposed to defeat adversaries that deceive the capacity-restricted tags. But, designing light-weight cryptographic authentication protocols with basic cryptographic primitives (xor, hash function) is a challenging task [18].

Another significant problem is the changing ownership of an RFID tag several times during its life-cycle. For instance, tags are initially created and attached to objects by producers, then labeled objects are taken over to retailers, and finally consumers buy tagged objects from shopping malls [13]. The ownership of a labeled object may be frequently transferred from one party to another. At the moment of the transfer, both new and old owners have the same information about the tag. This might cause privacy problems. This transfer should guarantee that the old owner should no longer be able to trace the future interactions and the new owner should not be able to trace old interactions. Besides having secure authentication protocols by providing privacy, the performance of the entire system becomes an important issue. Therefore, designing authentication protocol without compromising security and privacy begets decreases the efficiency of the whole system. However, achieving both security and privacy properties, the computational complexity of the tag and the server side can vary dramatically from one protocol to another. Hence, while handling security and privacy issues, it is also important to realize it with less computational complexity.

In order to resolve these security and privacy issues, numerous RFID authentication protocols have been recently proposed [1, 4, 5, 7, 8, 11, 12, 14–17]. However, some of them are not compliant to ownership transfer. Also, none of them achieves constant-time complexity for identification while providing forward untraceability against old-owner and backward untraceability (forward secrecy) against the new owner.

Our Contributions. We propose an efficient, secure and private RFID mutual authentication protocol which needs constant-time complexity to identify a tag. Then, we utilize this protocol and achieve a secure and efficient ownership transfer. We prove that our protocol achieves forward secrecy against the new owner and forward untraceability against the old owner. Moreover, we also show that our protocol provides forward secrecy against a strong attack and forward untraceability under an assumption that the adversary misses one subsequent successful protocol between the reader and the compromised tag.

The outline of the paper is as follows. In Section 2, security and threat model, security and privacy concerns are discussed in RFID systems for ubiquitous networks. Section 3 describes our proposed protocol. In Section 4, analysis of our protocol is given in detail. In Section 5, we conclude the paper.

2 Adversarial Model

In this section we describe our adversarial model used in analyzing the proposed protocol, then define the privacy notions which are also used to be proved. Since the tags and the reader communicates over an insecure wireless channel, we consider Byzantine adversarial model [6].

- Each tag memory is not tamper resistant and vulnerable to physical attacks.
- Each tag/reader performs cryptographic hash operations.
- The reader and tags communicate over an insecure wireless channel and so an active attacker can intercept, modify and generate messages.
- The messages between server and readers are transmitted securely.
- The reader and the server are assumed to be trusted parties. They cannot be compromised.

Since the tags are not tamper resistant, we assume that a strong adversary can corrupt a tag and access to its persistent memory. In this case, the adversary should not be able link any current and past communication of the victim tags. This privacy notion is called backward untraceability. We define it more formally as follows.

Definition 1. *Backward Untraceability: An RFID scheme provides backward untraceability if $\mathcal{A}$ compromising $\mathcal{T}_i$ at time t cannot trace the past interactions of $\mathcal{T}_i$ that occurred at time $t' < t$.*

On the other hand, the strong adversary should not be able to trace the future interactions of the victim tag. This privacy notion, called forward untraceability, is described as follows.

Definition 2. *Forward Untraceability: An RFID scheme provides forward untraceability if $\mathcal{A}$ compromising $\mathcal{T}_i$ at time t cannot trace the future interactions of $\mathcal{T}_i$ that occurred at time $t' > t$.*

3 The Proposed Protocol

In this section, we propose a novel scalable RFID authentication protocol which is the enhanced version of the scheme presented in [12]. In our protocol, we achieve the constant-time complexity for the authentication of synchronized tags whereas the complexity in [12] is $\mathcal{O}(N)$ where N is the number of tag in the system.

The notations used in the protocol are defined. Then, the initialization and the authentication phases are described in detail. The protocol is summarized in Figure 1.

3.1 The Notations

- $\in_R$: The random choice operator that randomly selects an element from a finite set.
- $\oplus$, $||$: XOR operator and concatenation operator, respectively.
- h, H : A hash function s.t. $h : \{0,1\}^* \rightarrow \{0,1\}^n$, $H : \{0,1\}^* \rightarrow \{0,1\}^{2n}$. Both of them are one-way and collision resistant functions.
- N : The number of tags in the database.
- N_a, N_b : n-bit nonce generated by the reader and the tag, respectively.
- K : n-bit secret shared between the tag and the reader.
- val_1, val_2 : n-bit the server validator of the tag and the reader, respectively.
- K^{old_1}, K^{old_2} : Previous n-bit secret shared between the tag and the reader.
- val_1^{old}, val_2^{old} : Previous n-bit the server validator of the tag and the reader, respectively.
- L, S : The seed value of val_1 and val_2, respectively.
- r_1, r_2 : n-bit random bit strings produced by $h(N_a)$, $h(N_b, K)$, respectively.
- v_i : n-bit random bit strings produced by $h(K, r_1, r_2)$.
- M_1, M_2 : $M_1 = v_1 \oplus L$, $M_2 = v_2 \oplus S$.
- DB : Server database.
- γ : n-bit string.
- *state* : 1-bit string is 0 or 1.

3.2 The Registration Phase

For each tag T_i, the following steps have to be performed by the registrar (e.g. the tag manufacturer) before the authentication protocol:

1. The registrar generates three n-bit random nonce (K, S, L). It also computes $val_1 = h(L, K)$, $val_2 = h(S)$. Initially, K^{old_1} and K^{old_2} are both equal to K, S^{old} is equal to S, and val_1^{old} is equal to val_1. Finally, state is set to 0 and it computes hash of the shared secret key K, $\gamma = h(K)$.

2. The registrar creates an entry in its back-end database and stores $(K, S, val_1, K^{old_1}, K^{old_2}, S^{old}, val_1^{old}, h(K))$ in the entry.
3. The registrar assigns $(K, L, val_2, state)$ to the tag T_i.

3.3 The Authentication Phase

In our protocol (see Figure 1) each tag stores its own triple values K, L, val_2, γ,and *state* . The reader stores the K, S, val_1 for that tag. The steps are described below.

Step 1. A reader randomly generates an n-bit nonce N_a and computes hash of it $r_1 = h(N_a)$. Then it sends r_1 to the tag T_i.

Step 2. The tag T_i randomly generates a n-bit N_b nonce and computes hash of it, $r_2 = h(N_b, K)$. Then, it checks the state. If its own state is 0, it computes hash of the shared secret key K. If it is not, the tag randomly generates a n-bit γ nonce. Later, the tag uses a pseudo-random function that digests r_1, r_2 messages with shared secret key K to compute $v_1||v_2 = H(K, r_1, r_2)$. The length of each v_1 and v_2 are both equal to n. After that, the tag computes message M_1 by simply XORing v_1 with secret L. Finally, the tag sends r_2, M_1 and γ messages to the reader.

Step 3. The reader transfers N_a, r_1, r_2, M_1, and γ to the server.

Step 4. The server firstly searches in DB that there exists $h(K)$ equals to γ. The server performs an exhaustive search among all tags in the database. It computes $v_1||v_2 = H(K, r_1, r_2)$ and $h(M_1 \oplus v_1, K)$. The server checks whether $h(M_1 \oplus v_1, K^{old_1})$ is equals to val_1. If one match is found, then the server computes M_2 message by XORing v_2 with S and then sends M_2 to the reader. After that, it updates $K^{old_2} = K^{old_1}$, $K^{old_1} = K$, $S^{old} = S$, $val_1^{old} = val_1$, $K = v_2$, $S = N_a$, and $val_1 = r_2$. If no match is found, then the server performs another an exhaustive search among all tags in the database. In this time, it computes $v_1||v_2 = H(K^{old_1}, r_1, r_2)$ and it checks whether $h(M_1 \oplus v_1, K^{old_2})$ is equals to val_1^{old}. If one match is found, the server computes M_2 message by XORing v_2 with S and sends M_2 to the tag. After that, it updates $K = v_2$, $S = N_a$, and $val_1 = r_1$. However, if there is no match, the server generates an n-bit random bit string and sends it to the reader. The reason behind sending random bit string is that this prevents any attacker to validate M_1 for random nonce r_1 and r_2.

Step 5. The reader forwards M_2 to the tag T_i. Upon receiving M_2 message, T_i computes $h(M_2 \oplus v_2)$ and checks whether it is equal to val_2. If equal, then it updates $K = v_2$, $L = N_b$, and $val_2 = r_1$.

3.4 The Ownership Transfer

When the owner of the tags are required to change one party to another, the tags are first synchronized with the server. The server runs at least two successful authentication protocols with tags in a secure environment where no adversary

is allowed to perform any passive/active attacks. Then, all the tags and their related information are transferred to new owner. Once the new owner receives the information and tags, he/she runs at least one successful protocol between readers and the tags in a secure environment where a malicious adversary is not allowed.

During the ownership transfer, the old owner does not need to transfer the secret values of K^{old_2} and S^{old} of the tags to the new owner because the remaining secrets are enough to communicate with the synchronized tags.

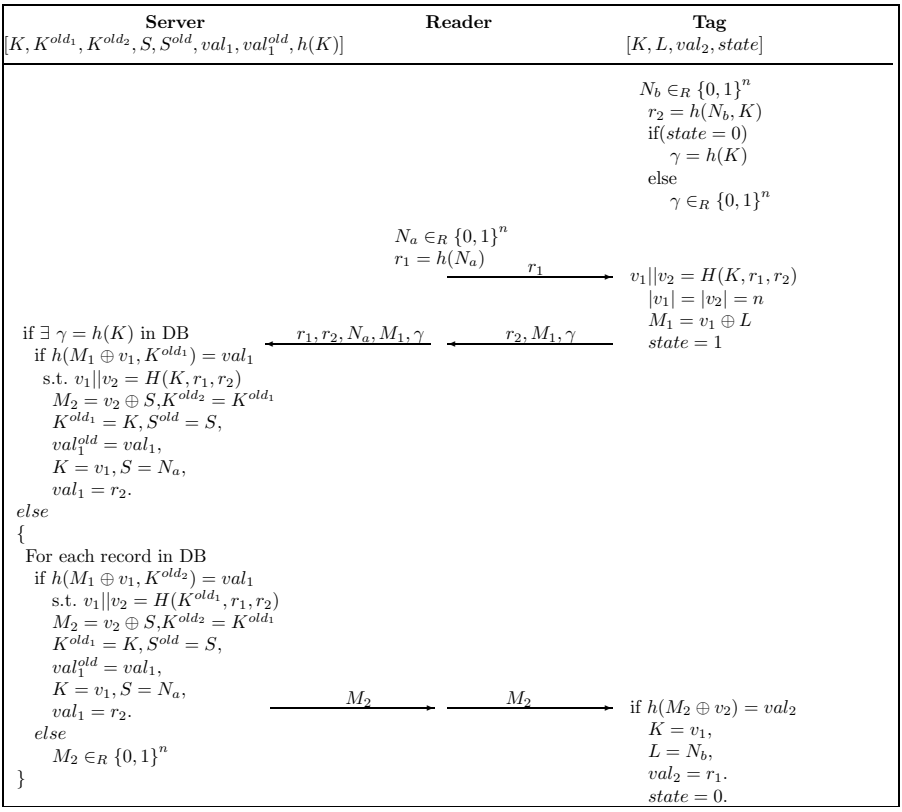

Fig. 1. The Proposed RFID Authentication Protocol

4 Security, Privacy, and Performance Analysis

In this section, we first describe the adversarial capabilities. Then, we analyze our ownership transfer protocol depicted in Figure 1 against passive and strong attacks.

In our model, we assume that each tag can perform cryptographic hash operations. The communication between server and readers are assumed to be

secure because they have no restriction on using SSL/TLS protocol. However, the reader and tags communicate over an insecure wireless channel and so an attacker can intercept, modify and generate messages. Also, each tag memory is not tamper-proof.

4.1 The Security against Timing Attacks

The proposed protocol is vulnerable to timing attacks [3]. An adversary can distinguish synchronized tags and un-synchronized tags by simply considering the response time of the server because the identification time for the latter tags requires much more than the former tags. This kind of attacks can be avoided by using distributed computation servers. Let us illustrate the solution. Assume that we have 2^{20} tags in the database and the server does only 2^{23} hash computation per second. Then, the time to identify an un-synchronized tag is $2^{20}/2^{23} = 0.125$s but for the synchronized tag is almost zero. For the solution, we can use multiple distributed server (say 16), then the identification time can be reduced to $0.125/16 = 7,8125$ms and when a synchronized tag is to be identified the server waits up to $7,8125$ms.

4.2 The Security against Passive Adversary

An offline passive adversary may want to know the contents of the secrets K and L stored in the tag $\mathcal{T}_i$. Then, the adversary simply eavesdrops the channels between a legitimate reader and $\mathcal{T}_i$ in order to get r_1, r_2, M_1, M_2 and γ. With these information and publish hash function H, she cannot obtain the secret K or L because of one-wayness of the hash function.

Moreover, the protocol also resists against replay attack because a challenge-response scheme is used in the protocol. In addition, for each session of the protocol a new pair of random numbers (r_1, r_2) are used. This prevents to use the same challenge-response values in other sessions.

Furthermore, our protocol is resistant against desynchronization even if' last flow of the protocol drops. Normally, this causes desynchronization of the tag secrets and the back-end server. However, this issue is resolved by storing previous tag secrets in the database. Hence the server can resynchronize with the tags in such a condition.

4.3 The Security against Strong Adversary

In this section, we will analyze the protocol depicted at Figure 1 in terms of backward and forward untraceability [2, 15, 19] against old owner, new owner, and a strong malicious adversary who can compromise a tag. As a starting point, we assume that at time t_i, the owner of the system is changed. We test backward untraceability for the new owner, denoted by $\mathcal{A}_n$, with assumption that $\mathcal{A}_n$ has had control over communications between reader and tags made before time t_i. Note that, the number of these communications is finite. Similarly, we test

forward untraceability against the old owner, denoted by $\mathcal{A}_o$. Also, we test these two privacy properties against a strong adversary $\mathcal{A}_s$ with assumption that $\mathcal{A}_s$ has ability of corrupting a tag and captures its secrets. Throughout the analysis, in order to make proofs more understandable, without loss of generality, we assume that there are only two tags in the system, namely $\mathcal{T}_0$ and $\mathcal{T}_1$. First of all, let us give the definitions of concepts mentioned above and the oracle that we use in the proofs of theorem given below.

Definition 3. *Oracle $\mathcal{O}^k$: The oracle chooses $b \in_R \{0,1\}$. If $b = 0$, $\mathcal{O}^k$ sends to the adversary the protocol transcript which was realized between tag $\mathcal{T}_0$ and the reader at time t_k. Similarly, if $b = 1$, the protocol transcript which was realized between tag $\mathcal{T}_1$ and the reader at time t_k is sent to the adversary by the oracle. At the end, the adversary sends the bit b' by after investigating the transcript sent. If $Pr[(b' = b) = 1] = \frac{1}{2} + \epsilon$, where ϵ is non-negligible, than the adversary wins.*

One can give simplified version of the oracle defined above as follows: At time t_i, $\mathcal{A}$ gets information of server and the tag $\mathcal{T}_0$. Then at time t_k, $\mathcal{O}^k$ chooses $b \in_R \{0,1\}$. The transcript sent to the adversary according to value of b same as above. Then, $\mathcal{A}$ returns $b' = 0$ if he thinks the transcript sent by oracle realized between reader and tag $\mathcal{T}_0$. Otherwise the adversary returns $b' = 1$. If $Pr[(b' = b) = 1] = \frac{1}{2} + \epsilon$, where ϵ is non-negligible, than the adversary wins.

Throughout the proofs given to the corresponding theorem, four subsequent successful protocol transactions are enough. Thus, without loss of generality, we assume that $i = 4$ is the time where server owner changed, i.e. at time t_4. Moreover, addition to the notations given at protocol steps, we use left subscript part to denote the time that it was used.

In order to obtain traceability capability of $\mathcal{A}_n$, we start studying with more powerful adversary $\mathcal{A}_c$, who has had all secrets of the server and tags at time t_i and observed all protocol transactions realized before given time.

Theorem 1. *The system has backward untraceability property for time t_k satisfying $k < i - 3$ for the adversary $\mathcal{A}_c$*

Proof. Since at time t_4, $\mathcal{A}_c$ knows the value of ${}_4val_1$ and this value equals to ${}_3r_2$, then at time t_3, $\mathcal{A}_c$ can traces $\mathcal{T}_0$. Moreover, as $\mathcal{A}_c$ knows the value of ${}_4S^{old_1}$, then she knows the value of ${}_3S$. Thus, ${}_2N_a$ value is known. Therefore, at time t_2, $\mathcal{A}_c$ can trace $\mathcal{T}_0$ as he can figure out the value of ${}_2r_1$ from $h({}_2N_a)$. Note that, after that point, $\mathcal{A}_c$ knows ${}_2r_2$ and ${}_2M_2$ and since ${}_2K = {}_4K^{old_2}$, the values of ${}_2v_1$ and ${}_2v_2$ are known. Hence, ${}_2S$ is known. So, $\mathcal{A}_c$ learns the value of ${}_1N_a$. From this knowledge, $\mathcal{A}_c$ calculates ${}_1r_1$. Therefore, $\mathcal{A}_c$ can trace $\mathcal{T}_0$ at time t_1, which means $\mathcal{A}_c$ also learns the values of ${}_1r_2$, ${}_1M_1$, ${}_1M_2$. Apart from these values, ${}_1L$ is also known. Note that, the only thing $\mathcal{A}_c$ knows about the transaction happened at time t_0 is ${}_0N_b$. Thus, the probability of $\mathcal{A}_c$'s finding the correct value of ${}_0r_2$ is $\frac{1}{2^n}$ since ${}_0K$ is not known and the range of hash function h is $\{0,1\}^n$. Similarly, finding correct values of ${}_0r_1$,${}_0M_1$,${}_0M_2$ is $\frac{1}{2^n}$. Thus, the probability that $\mathcal{A}_c$ distinguishes the transcript that the oracle sent is $\frac{1}{2} + \frac{1}{2^n}$. However, $\frac{1}{2^n}$ is negligible.

Therefore, if $\mathcal{A}_c$ has all secrets of the server and tags at time t_i, then the system has backward untraceability property for time t_k satisfying $k < i - 3$.

Remark 1. The values of K^{old_2} and S^{old} of tags are stored in server database in order to overcome synchronization problem. If the system is synchronized when ownership transfer is realized, then K^{old_2} and S^{old} values are not given to $\mathcal{A}_n$.

At the next part, we give a backward traceability result for an adversary $\mathcal{A}_{cR}$, which is like $\mathcal{A}_c$ with exception indicated at Remark 1.

Corollary 1. *The system has backward untraceability property for time t_k satisfying $k < i - 2$ for the adversary $\mathcal{A}_{cR}$.*

Remark 2. The privacy is the main aim that should be reached. Therefore, just before ownership transfer, $\mathcal{A}_o$ completes two successful protocol transactions with tags such that no part of the protocol transcripts are seen by $\mathcal{A}_n$.

Note that the adversary $\mathcal{A}_c$ with incapability explained at Remark 2 corresponds to the new owner, $\mathcal{A}_n$. Thus, we have the following corollary.

Corollary 2. *For the new owner, $\mathcal{A}_n$, the system has backward untraceability property for time t_k satisfying $k < i$.*

Theorem 2. *If $\mathcal{A}_o$ has all secrets of the server and tags at time t_i, then the system has forward untraceability property for time t_k satisfying $k > i$.*

Proof. Since ownership transfer occurs, $\mathcal{A}_o$ misses at least one of the subsequent successful protocol transactions between $\mathcal{A}_n$ and tags. We can get the best result if one subsequent successful transaction miss is assumed. In that case, $\mathcal{A}_o$ only knows values of ${}_5K^{old_1}$, ${}_4K^{old_2}$, ${}_5S^{old_1}$ and ${}_4val_1{}^{old}$. Since the attacker missed a subsequent successful transaction, the other values are unknown. Note that, $\mathcal{A}_o$ can find the value of ${}_4r_2$ with possibility of $\frac{1}{2^n}$ since the value of ${}_4N_b$ is not known. By similar arguments, $\mathcal{A}_o$ guesses the value ${}_4r_2$ with possibility of $\frac{1}{2^n}$. Although $\mathcal{A}_o$ knows the values of ${}_4S$ and ${}_4L$, as ${}_4v_1$ and ${}_4v_2$ are not known, $\mathcal{A}_o$ can figure out the values of ${}_4M_1$ and ${}_4M_2$ with possibility of $\frac{1}{2^n}$. Hence, the probability that $\mathcal{A}_n$ distinguishes the transcript that the oracle sent is at most $\frac{1}{2} + \frac{1}{2^n}$. However, $\frac{1}{2^n}$ is negligible.

Therefore, if $\mathcal{A}_0$ has all secrets of the server and tags at time t_i, then the system has forward untraceability property for time t_k satisfying $k > i$.

Our next result is about the adversary, $\mathcal{A}_s$, who can corrupt a tag and capture all secrets of the tag at any given time and follow all steps of the each successful protocol runs before and after the time that corruption occurs.

Corollary 3. *If $\mathcal{A}_s$ corrupts a tag at time t_j with $j \neq i$, then the system has backward untraceability for time t_k satisfying $k < j - 1$ and forward untraceability for time t_k satisfying $k > j + 1$ under the assumption that $\mathcal{A}_s$ misses the transactions occurred at time $j + 1$ and $j - 1$.*

Proof. Forward secrecy part is direct result of Theorem 2. Moreover, the backward secrecy result is derived from Remark 3

Remark 3. If $\mathcal{A}_s$ does not miss the transaction at $j-1$, then by the knowledge of ${}_jval_2$, he deduces the value of ${}_{j-1}r_1$. Thus, the values of ${}_{j-1}r_2$, ${}_{j-1}M_1$, ${}_{j-1}M_2$ are known to him. Thus, in this case, $\mathcal{A}_s$ can trace the corrupted tag at time t_{j-1}. However, no more traces are possible, because $\mathcal{A}_s$ knows only the value of ${}_{j-2}N_b$ about the transaction realized at time t_{j-2} and from the similar arguments given at proof of Theorem 1, the success probability that $\mathcal{A}_s$ traces the corrupted tag at time t_{j-2} is $\frac{1}{2} + \frac{1}{2^n}$ and $\frac{1}{2^n}$ is negligible.

Remark 4. If $\mathcal{A}_s$ does not miss any transaction after corruption occurs, then $\mathcal{A}_s$ can trace the corrupted tag forever.

Theorem 3. *The proposed protocol satisfies tag authentication under the assumption specified in Corollary 3.*

Proof. First of all, let us assume that the adversary has no corrupt tag capability. In this case, the adversary has to learn the value of either K or K^{old_1} to impersonate the tag. To learn the values of these variables, the adversary has to learn the value of v_1 of previous protocol transcript. However, to learn the value of v_1, the adversary has to figure out the value of K of previous runs or the value of L. However, the value of L is the chosen random N_b value of previous run. Thus, the adversary can only guess the value of L. Therefore, the values v_1, K and K^{old_1} are dependent each other. Thus, the only remained way for the adversary to impersonate the tag is to guess the value of v_1, K or K^{old_1} correctly. Since the space of these variables are large enough, the success probability of the adversary is negligible.

Moreover, since the tag authentication is investigated under the assumption Corollary 3, the system satisfies tag authentication for the case where the adversary can corrupt the tag.

4.4 Performance Issues

Considering memory storage for tag identifiers or keys and other information, our protocol requires $3n + 1$ bit ($3n$-bit for K, L, and val_2 and 1-bit for *state*) memory in tag side. Contrary to tags, server has no limited resource so we do not consider the server-side memory usage.

Concerning computational cost, our protocol requires at most 4 hash computation overhead for the tag. If the tags and the server are synchronized, the computational complexity at the server side is $\mathcal{O}(1)$. Otherwise, the complexity is at most $\mathcal{O}(N)$.

5 Conclusions

In this paper, we first proposed a secure and efficient an RFID mutual authentication protocol which is the revised version of the scheme presented in [12].

With the use of the authentication protocol, we achieve ownership transfer. We prove that our protocol provides forward untraceability against the old owner of the tags and backward untraceability against the new owner of the tags. Also, we show that our authentication protocol provides backward untraceability of a tag against an adversary who compromises the tag and forward untraceability under the assumption that the adversary misses at least one of the subsequent authentication protocol between the tag and the reader. Our protocol requires $\mathcal{O}(1)$ complexity to identify a synchronized tag.

References

1. Alomair, B., Clark, A., Cuellar, J., Poovendran, R.: Scalable RFID systems: a privacy-preserving protocol with constant-time identification. In: International Conference on Dependable Systems and Networks, pp. 1–10 (2010)
2. Avoine, G.: Cryptography in Radio Frequency Identification and Fair Exchange Protocols. PhD thesis, EPFL, Lausanne, Switzerland (December 2005)
3. Avoine, G., Coisel, I., Martin, T.: Time Measurement Threatens Privacy-Friendly RFID Authentication Protocols. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 138–157. Springer, Heidelberg (2010)
4. Burmester, M., de Medeiros, B., Motta, R.: Anonymous RFID authentication supporting constant-cost key-lookup against active adversaries. IJACT 1(2), 79–90 (2008)
5. Dimitriou, T.: A Lightweight RFID Protocol to protect against Traceability and Cloning attacks. In: SECURECOMM 2005: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks, pp. 59–66. IEEE Computer Society, Washington, DC (2005)
6. Dolev, D., Yao, A.C.: On the security of public key protocols. In: Proceedings of the 22nd Annual Symposium on Foundations of Computer Science, pp. 350–357. IEEE Computer Society, Washington, DC (1981)
7. Erguler, I., Anarim, E.: Practical attacks and improvements to an efficient radio frequency identification authentication protocol. Concurrency and Computation: Practice and Experience (October 2011)
8. Fernàndez-Mir, A., Trujillo-Rasua, R., Castellà-Roca, J., Domingo-Ferrer, J.: Scalable RFID Authentication Protocol Supporting Ownership Transfer and Controlled Delegation. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 147–162. Springer, Heidelberg (2012)
9. Finkenzeller, K.: RFID Handbook. John Wiley and Sons (2003)
10. Garfinkel, S., Rosenberg, B.: RFID: Applications, Security, and Privacy. Addison-Wesley (2005)
11. Ha, J., Moon, S.-J., Nieto, J.M.G., Boyd, C.: Low-Cost and Strong-Security RFID Authentication Protocol. In: EUC Workshops, pp. 795–807 (2007)
12. Kardaş, S., Levi, A., Murat, E.: Providing Resistance against Server Information Leakage in RFID Systems. In: New Technologies, Mobility and Security – NTMS 2011, Paris, France, pp. 1–7. IEEE Computer Society (February 2011)
13. Lim, C.H., Kwon, T.: Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 1–20. Springer, Heidelberg (2006)

14. Molnar, D., Wagner, D.: Privacy and security in library RFID: issues, practices, and architectures. In: CCS 2004: Proceedings of the 11th ACM Conference on Computer and Communications Security, pp. 210–219. ACM (2004)
15. Ohkubo, M., Suzuki, K., Kinoshita, S.: Cryptographic approach to 'privacy-friendly' tags. In: RFID Privacy Workshop. MIT, Massachusetts (2003)
16. Song, B., Mitchell, C.J.: RFID authentication protocol for low-cost tags. In: WiSec 2008: Proceedings of the First ACM Conference on Wireless Network Security, pp. 140–147. ACM (2008)
17. Song, B., Mitchell, C.J.: Scalable RFID Security Protocols supporting Tag Ownership Transfer. Computer Communication (March 2010)
18. Vajda, I., Buttyán, L.: Lightweight Authentication Protocols for Low-Cost RFID Tags. In: Second Workshop on Security in Ubiquitous Computing – Ubicomp 2003 (2003)
19. Van Le, T., Burmester, M., de Medeiros, B.: Universally Composable and Forward-secure RFID Authentication and Authenticated Key Exchange. In: Bao, F., Miller, S. (eds.) ACM Symposium on Information, Computer and Communications Security – ASIACCS 2007, Singapore, Republic of Singapore, pp. 242–252. ACM Press (March 2007)