Egyptian Informatics Journal (2013) **14**, 89–98

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

ORIGINAL ARTICLE

SRFID: A hash-based security scheme for low cost RFID systems

Walid I. Khedr *

Department of Information Technology, Faculty of Computers & Informatics, Zagazig University, Zagazig 44519, Egypt

Received 4 April 2012; revised 29 January 2013; accepted 7 February 2013
Available online 11 March 2013

KEYWORDS
RFID;
Auhentication;
Hash function;
Privacy;
Security

Abstract Radio Frequency Identification (RFID) technology is a promising technology. It uses radio waves to identify objects. Through automatic and real-time data acquisition, this technology can give a great benefit to various industries by improving the efficiency of their operations. However, this ubiquitous technology has inherited problems in security and privacy, due to the powerful tracking capability of the tags. This paper proposes a new simple, low cost, and scalable security scheme relying on one-way hash functions and synchronized secret information. The proposed scheme provides a two steps mutual authentication between the backend server and the tag which does not require a secure channel between the tag reader and the backend server to complete the authentication process. The proposed scheme meets the requirements for tag delegation and secure tag ownership transfer. The general idea is to change the ID of a tag on every read attempt in a secure and synchronized manner. This means that attempts like eavesdropping, replay attacks, tag cloning, tag tracing, denial of service attack, or man-in-the-middle attacks cannot compromise the scheme. Our analysis results show that the proposed scheme outperforms existing schemes in terms of security and performance.

1. Introduction

Radio Frequency Identification, abbreviated "RFID," basically provides a means to identify objects having RFID tags attached. Fundamentally, RFID tags provide the same functionality as barcodes but usually have a globally unique identifier. Using RFID, the identification is performed electromagnetically. Thus, there is, in contrast to barcodes, no line-of-sight necessary, and the identification can also be performed in contactless way. RFID also has the advantage that bulk reading is possible and that it is not susceptible to dust, dirt, or vibration like barcodes. Because of these characteristics, RFID is envisioned to be a convenient replacement for optical barcodes in the future. Unfortunately, RFID also introduces problems respecting data security and privacy arises. RFID systems have three main components [1]:

* Tel.: +20 1005371906; fax: +20 55 226 8338.
E-mail address: wkhedr@zu.edu.eg

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

- *RFID tags:* RFID tags are used as a label for item identification and communicate with a reader [1]. The reader passes tag data to the backend server for further processing, including tag identification and information retrieval [2].
- *RFID readers:* RFID readers send and receive data to and from tags. RFID readers are the connecting element between the RFID tags and the backend systems.
- *Backend server:* Readers are used for querying tags and reading and writing tag data. All the read data need to be processed, and the data to be written need to be available, so that an additional system component is required to form a complete RFID system which is the backend server.

RFID tags are classified into three types: active, semi-passive, and passive. Active tags contain batteries, so that they can actively communicate with the reader. Semi-passive tags also contain batteries, but they wait for the reader's query. As for passive tags, the power comes from the reader [2].

The key challenge in providing security mechanisms to passive RFID tags is that such tags have extremely weak computational power because they are designed to be ubiquitous low cost [3]. This means that heavy duty cryptography is not suitable for these types of tags. Our proposed scheme requires small amount of computation, since it only needs to have a one-way hash function, which makes the proposed scheme suitable for these types of tags.

1.1. RFID security problems

The characteristics of RFID systems can cause several threats which results in serious information leakage. And the adversary can engage in various illegal behaviors by using the acquired information. These threats are addressed in several studies [4–7] and summarized in [2], [8] and [9]:

- *Eavesdropping*: As communication between the tag and the reader is based on radio frequency, anyone can eavesdrop.
- *Replay attack:* This is an attack in which an adversary retransmits a message obtained by during the authentication process.
- *Cloning*: An adversary can read the tag and then clone the tag by writing all the obtained data into a blank tag.
- *Tag tracing:* Attackers can either identify the same tag from passively logged messages or interact actively with the tag to understand its location.
- *User privacy violation:* The adversary can analyze the output value of a specific RFID tag and acquire the attached object information, which can be used to violate the user's personal privacy.
- *Data forging:* Attackers can modify information stored on tags like prices which causes great loss.
- *Denial of Service* (DoS) attack: The adversary could block messages transmitted between a server and a tag. Such an attack could cause the server and the tag to lose synchronization.
- *Man-in-the-Middle (MITM) attack:* The adversary could interfere with messages sent between a server and a tag (e.g., by insertion, modification, or deletion).

1.2. RFID security requirements

To solve the security problems above, the following security requirements should be considered. These requirements are mentioned in [4,7,8,9–11] which are summarized as follows:

- *Mutual authentication:* The property permits the reader and the tag in a communication to verify the identity of the other.
- *Confidentiality*: An attacker should be able neither to analogize nor calculate a certain value through all tag messages transmitted through an insecure channel nor infer a tag's own ID.
- *Indistinguishability*: It is essential that the transmitted tag information should not be the same as, expectable, or distinguishable from the transmission information of another tag.
- *Forward security:* It is essential that the previously transmitted information cannot be traced using the present transmission tag information.
- *Desynchronization resilience*. An RFID protocol should be resilient to attacks that are targeted toward desynchronizing the tag and the backend server. With the use of shared secrets and information, it is important that the copies of any shared secret or information stored at the tag and the backend server must be consistent.
- *Tag delegation:* Delegation enables a backend server to delegate the right to identify and authenticate a tag to a specified entity for a limited number of queries.
- *Tag ownership transfer:* Tag ownership means having authorization to identity a tag and control all the related information. Tag ownership transfer implies a shift of such capabilities to a new owner [12].

The rest of the paper is organized as follows: Section 2 discusses related works. Section 3 presents the proposed scheme. Security analysis and performance evaluation are conducted in Section 4 and Section 5, respectively. Section 6 concludes the paper.

2. Related work

The authentication protocol proposed in this paper is a hash-based authentication method. Therefore, the related works introduced here focus on the hash-based authentication protocols for RFID tags.

The hash-based authentication protocol proposed in [7] dealt mainly with untraceability as the refreshment of the shared secret between the tag and the reader or the backend server; but this suffers severely from adversarial attacks, including counterfeiting, man-in-the-middle attacks. In other words, it will most likely fail to carry out mutual authentication later on account of desynchronization.

Hash-lock protocol [13] used $metaID = H(K)$ to hide the tag's real ID, where K is the shared secret between the tag and the backend server and H is a one-way hash function. Although this scheme offers certain level of reliability at low cost, an adversary can easily track the tag via its *metaID,* and thus, the transaction secret or privacy would be at risk. Furthermore, since the key shared between the tag and the backend server is sent in plaintext, even an inactive adversary can easily sniff the channel to spoof the tag later [14]. The protocol also requires a secure channel between the reader and the backend server during the authentication process to transfer the tag *ID* form the backend server to the reader.

Gervasi [15] proposed a mutual authentication protocol for RFID tags. The RFID reader and tag will carry out the authentication based on their synchronized secret information. The synchronized secret information will be monitored by a component of the database server. However, the protocol is subject to denial of service, and tag impersonation attacks as proved by Piramuthu [16], and backend server takes need work of $O(n)$ to identify a tag. The protocol also requires a secure channel between the reader and the backend server to complete the authentication process.

The following protocol is offered by [17]. This protocol allows for mutual authentication in two message exchanges and at the same time prevents the tag ID to be compromised during the authentication process. However, the protocol is not scalable; the backend server is not able to handle a large tag population. It is not able to identify multiple tags using the same radio channel. The server requires an exhaustive search to identify individual tags. The protocol also requires a secure channel between the reader and the backend server during the authentication process to transfer the tag *ID* between the backend server and the reader and suffers from desynchronization attack.

Shen [18] proposed a novel anonymous RFID authentication protocol, termed ARAP, which can accomplish the authentication without disclosing real IDs of the participating tags, and it offers the anonymity of tags in addition to tag unlocatability and untrackability. It assumes that each tag should use and store dynamic pseudonyms IDs. This could not be applicable to the limited memory RFID tags. The protocol is also subject to denial of service attack; an attacker can continuously query the same tag until the tag exhausts all the pseudonyms. Further, the protocol cannot satisfy the forward security since knowledge of the stored tag's pseudonyms can help to identify tag previous interactions.

Boyeon [19] proposed a protocol for low cost tags using the hash function, the keyed hash function, the pseudo-random number generator, and the XOR operator to guarantee the security and privacy of the RFID system. The protocol is found to be under an elaborate replay attack which can make the attacker get authenticated by the tag unlawfully and is also prone to denial of service attacks [12,14].

Lim et al. proposed a new protocol [4] to solve almost all of the existing RFID security problems. However, Zhou [14] showed that information in this protocol can be exposed via the brute-force attack. It is also found that this protocol is subject to denial of service attack, since the tag stores the received random number N_R until it received the last message from the reader to authenticate the backend server. An adversary can query the tag many times with different random numbers N_R each time. When the tag receives the query with N_R from the adversary, it will store it as a pseudo-random number to identify the session which consume the tag's user memory, so that it can no longer respond to the reader.

Zhou [14] proposed a lightweight anti-desynchronization privacy preserving RFID authentication protocol. In this protocol, the backend server keeps the history of the random key update to prevent the active attackers from desynchronizing the shared secret between the tag and the backend server. Although this technique prevents the replay attack, it is found that there exists a serious problem; an adversary can launch a denial of service attack. An adversary can query the tag n times with different random numbers r each time. When the tag receives the query with r from the adversary, it will calculate and store $H(T_i \oplus r)$ as a pseudo-random number to identify the session which consume the tag's user memory, so that it can no longer respond to the reader.

Finally, Cho [8] proposed a hash-based mutual authentication protocol as a solution to the privacy and forgery problems. The protocol is designed to send a random number generated by a tag to a backend server without disclosure. However, the backend server takes $O(n)$ to identify a tag. Also, an adversary can perform a desynchronization attack by intercepting the message in Step 5 [8] and replacing $R_t \oplus s_{j+1}$ by a random value v. According to the protocol, the tag will try to extract s_{j+1} from v using R_t. In this case, $R_t \oplus v \neq s_{j+1}$, this can bring system to a mess. Further, the protocol cannot satisfy the forward security since knowledge of the stored tag's secret can help to identify tag previous interactions with complexity $O(2^{48})$.

3. Proposed SRFID scheme

A security scheme to solve the RFID security and privacy issues is proposed in this section. The scheme provides mutual authentication and tag *ID* updating, which can resists most attacks between backend server and tag including tracing, cloning, eavesdropping, and other attacks described in Section 1.1. The proposed scheme also satisfies the security requirements described in Section 1.2. The proposed scheme is based on the challenge—response mechanism. The backend server stores all the relevant information of the tags. The content of tags is indexed by a unique *ID*. Therefore, the searching is efficient, and the system is scalable. A tag transmits its current tag *ID* to the reader which forwards it to the backend server as index in the database. In order to cope with the counterfeiting attack issues discussed in Section 1.1, the proposed scheme is based on the mutual authentication between the tag and the backend server. The mutual authentication is based on the sharing of two secret values between the tag and the backend server.

After a successful mutual authentication, the tag *ID* is updated by both the tag and the backend server, which provides the forward security for the system. These secret update mechanisms may results in desynchronization attack. To prevent this attack, the backend server keeps the current records and the previous records of the update process. While the server fails to authenticate a tag because of the desynchronization attack, it recovers the old *ID* from the previous secrete values update record to complete the authentication. A secret update protocol is also proposed for tag delegation and tag ownership transfer.

3.1. Notation

The following table lists the notations used in the proposed scheme protocol.

Table 1

Table 1 Notation.

Symbol	Meaning
ID	The current tag's ID stored in the server
ID_T	The current tag's ID stored in the tag
ID_{old}	The previous tag's ID
ID_H	The current tag's hidden ID: Mutually shared secret between backend server and tag
ID_{H_old}	The previous tag's hidden ID
$h()$	One hash function available to all parties
R	Random number generated by the reader
SQN	A sequence number: Mutually shared secret between backend server and tag.
$INC()$	SQN increment function
$DEC()$	SQN decrement function

3.2. Assumptions

In the proposed scheme, the following assumptions are defined:

1. Tags have limited processing power.
2. Server has significantly greater computational ability than a tag.
3. The channel between the server and the reader is assumed to be secure if detailed tag information is going to be transferred from the server to the reader after the authentication process. Otherwise, the proposed scheme does not require a secure channel between the server and the tag during the mutual authentication process.
4. The reader and tags communicate over an insecure channel
5. Tags are passively powered and only need to have a one-way hash function $h()$, which makes the proposed protocol suitable for low cost RFID-based. Lightweight cryptographic hash functions for implementing RFID protocols have been recently proposed, for example, Keccak and Quark lightweight hash functions [20,21]. The RFID tag does not need a pseudo-random number generator PRNG. This will further reduce the cost of the RFID tags.
6. The tag and the backend server share two secrete values:
   - ID_H is a hidden tag identification number which is only known to the tag and the backend server and is updated each authentication session.
   - SQN is a sequence number which is known only to the tag and the backend server and is updated after each authentication session. The initial value of SQN is n_0. The sequence number SQN prevents third parties from using intercepted authentication messages for fake authentications later on. It also proves to both the server and the tag that the authentication messages have not been used before.

   These two values are generated by the backend server and are written in a secure manner into the tag's user-bank memory before deployment
7. The backend server database stores three more values for each tag: ID, ID_{old}, and ID_{H_old}, where:
   - ID is the tag's current ID. Its initial value is $h(ID_H||SQN)$.
   - ID_{old} is the last session tag's ID.
   - ID_{H_old} is the last session tag's hidden ID.

3.3. Authentication process

To carry out the tag identification by the authorized devices, tag, reader, and server conduct the following steps which are visualized in Fig. 1:

Step 1. A reader transmits a query and a random challenge number R to the tag.
Step 2. Upon receiving R, the tag computes its current $ID_T = h(ID_H||SQN)$ and the message $M_1 = h(ID_H||R)$ then transmits them to the reader which forwards them along with R to the server. The backend server uses ID_T to locate the tag in the database and uses M_1 to authenticate the tag
Step 3. When the backend server receives ID_T and M_1, it searches its database for ID to find the corresponding tag record.
Step 4. The backend server computers $M'_1 = h(ID_H||R)$ and compares it to the received M_1 if they do not match, the connection is terminated. If, $M'_1 = M_1$, the tag is authenticated, Step 5 is performed.
Step 5. The backend server authenticate itself to the tag by:
- Incrementing SQN: $SQN = INC(SQN)$
- Saving the current value of ID_H: $ID_{H_{old}} = ID_H$ to prevent the abnormal operation problem described later.
- Update ID_H: $ID_H = h(SQN||ID_H)$
- Compute $M_2 = h(ID_H||SQN||R)$ which is used by the tag to authenticate the reader and the server.

Step 6. The backend server sends ID_T, M_2, and R to the reader which forwards them to the tag.
Step 7. The backend server updates the current tag ID for the next authentication process by:
- Incrementing SQN one more time: $SQN = INC(SQN)$.
- Saving the current value of ID_{old}: $ID_{old} = ID$ to prevent the abnormal operation problem described later.
- Computing the next ID of the tag: $ID = h(ID_H, SQN)$

Step 8. When the tag receives M_2 and R, it authenticates the reader and the backend server as follows:
- Incrementing its SQN: $SQN = INC(SQN)$. This should match the value of SQN computed by the server in Step 5.
- Computing $ID_{tmp} = h(SQN||ID_H)$. This should match the value of ID_H computed by the server in Step 5.
- Computing $M'_2 = h(ID_{tmp}||SQN||R)$ and comparing it to the received M_2:
  - If they do not match, the tag restores its last state by decrementing SQN: $SQN = DEC(SQN)$ and deleting ID_{tmp}. The connection is then terminated.
  - If they match (i.e., the backend server is authenticated), Step 9 is performed.

Step 9. The tag updates SQN: $SQN = INC(SQN)$ and ID_H: $ID_H = ID_{tmp}$ to be able to compute $ID = h(ID_H||SQN)$ for the next authentication process.

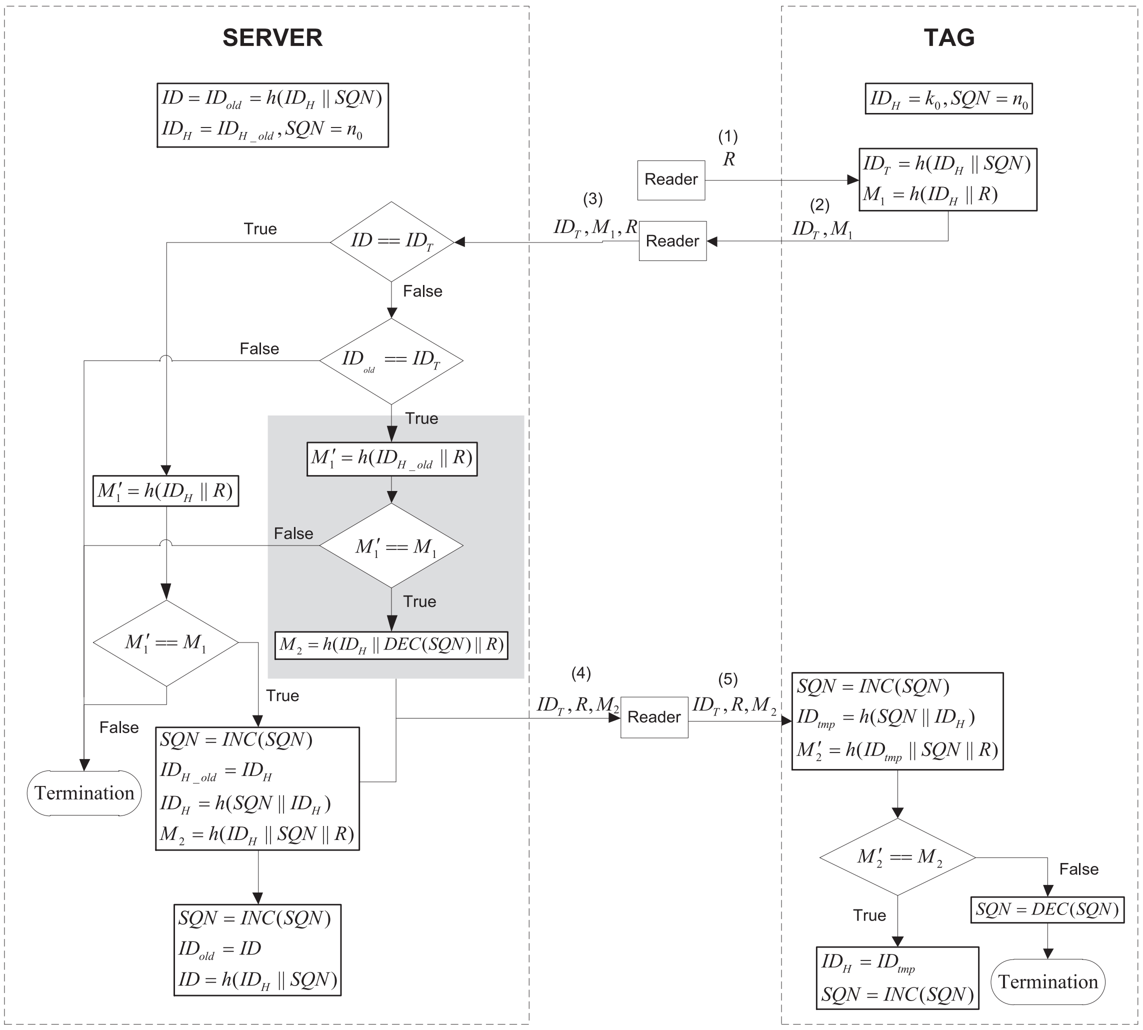

Figure 1 The proposed SRFID scheme.

This should match the value of ID and ID_H computed by the server in Step 6.

3.3.1. Abnormal operation of authentication process

Loss, interception, or blocking of messages exchanged between the tag and the reader results in abnormal operation of the proposed scheme. There are three cases:

1. Loss, interception, or blocking of R in Step 1 has no implications and will not result in any synchronization problem. The reader will resend R after a predefined time t.
2. Loss, interception, or blocking of ID and M_1 in Step 2 has no implications and will not result in any synchronization problem; since both ID_H and SQN have the same values in both the tag and the backend server. The reader will resend R after a predefined time t.
3. Loss, interception, or blocking of R and M_2 in Step 5 results in synchronization problem, since the backend server's SQN is greater than the tag's SQN by two increments, and both the backend server's ID_H and the tag's ID_H are different; because the tag's ID_H equal to ID_{H_old} stored in the server.

To solve this problem, the backend server stores ID_{H_old} which is equal to the tag's current ID_H and stores ID_{old} which equal to the tag's current ID. This problem is fixed as follows:

Step 1. The backend server sends a random number R to the tag as specified by Step 1 of the authentication process.
Step 2. When the tag receives R, it computes $ID_T = h(ID_H||SQN)$ which is equal to ID_{old} stored in the server, and computes $M_1 = h(ID_H, R)$ which is equal to $h(ID_{H_{old}}, R)$ and sends them to the reader which forwards them to the backend server.
Step 3. When the backend server receives ID_T and M_1, it searches its database for ID_T to find the corresponding tag record. In this case, a match is not fond. So, it searches the database for ID_{old}. If a match is found, the server computes $M_1' = h(ID_{H_{old}}||R)$ and compares it to the received M_1 if they do not match, the connection is terminated. If $M_1' = M_1$, the backend server recomputes M_2 which is sent before in Step 6 of the authentication process: $M_2 = h(ID_H||DEC(SQN)||R)$. Note that we used $DEC(SQN)$ instead of SQN in the calculation of the message M_2 since SQN is incremented in Step 7 of the

authentication process after sending M_2 in Step 6 of the authentication process. Also note that using $DEC(SQN)$ instead of SQN in the calculation of the message M_2 does not implies the changing of SQN value calculated in Step 7 of the authentication process. This step is shown in the gray area of Fig. 1.

3.4. Tag delegation and ownership transfer

The proposed scheme is scalable. It provides tag delegation and ownership transfer in an effective way. To prove the scalability of our scheme, we follow the approach and assumptions used in [9].

3.4.1. Tag delegation

Tag delegation enables a server to delegate the right to identify and authenticate a tag to a specified entity for a given number of times [9], as described in Section 1.2. Such a procedure could be used to reduce the computational load on a server [9].

In the proposed scheme, tag delegation is straightforward. When a server S wants to delegate tag T to an entity, that is, allows an entity to query tag m times, it generates the following values for tag T: $ID^i, ID^i_{old}, M^i_1, M^i_2, R^i$, where $i = 1, \ldots, m$. These values can be easily generated by the server, since the server does need to interact with the tag or the reader to generate these values. These values are transferred to the entity via a secure channel. As a result, the entity can authenticate T a maximum of m times. However, the entity receiving the delegation right cannot update the tag secrets, as it does not know ID_H or SQN.

3.4.2. Tag ownership transfer

In order to achieve ownership transfer of a tag T, S must transfer the secrets ID_H and SQN to the new owner via a secure channel. This transfer should only take place after the old owner has updated the secrets and identifiers for T, in order to protect the privacy of previously conducted transactions against possible tracking by the new owner [9]. The server of the new owner should also update the tag secrets after receiving them from the old owner, in order to protect the privacy of future transactions against possible tracking by the old owner [9]. This latter update needs to take place in an environment where there is no possibility of eavesdropping by the old owner [9]. Once this is complete, only the server of the new owner will be able to authenticate T and update the secrets for T [9]. A protocol to update tag secrets for secure tag ownership transfer is introduced and visualized in Fig. 2:

Step 1. The server transmits an update message and a random number R through the reader to the tag.
Step 2. Upon receiving R, the tag computes $ID_T = h(ID_H||SQN)$ and $M_1 = h(ID_H||R)$ then transmits them to the reader which forwards them along with R to the server.
Step 3. When the backend server receives ID_T and M_1, it compares them with the stored ID and M_1. If they do not match, the connection is terminated. If they match, the backend server generates a random number R' and performs the following computations:

- $M_2 = ID_{H/new} \oplus h(R'||ID_H)$, this securely transfer the new ID_H to the tag.
- $M_3 = SQN_{new} \oplus h(R'||SQN)$, this securely transfer the new SQN to the tag.
- $M_4 = h(ID_{H/new}||SQN_{new}||R')$
- $ID_H = ID_{H/new}$, this updates the new owner ID_H value.
- $SQN = SQN_{new}$, this updates the new owner SQN value.

Step 4. The backend server sends ID_T, R', M_2, M_3, and M_4 to the reader which forwards them to the tag.
Step 5. When the tag receives R', M_2, M_3, and M_4 it performs the following computations:

- $ID_{tmp} = M_2 \oplus h(R'||ID_H)$
- $SQN_{tmp} = M_3 \oplus h(R'||SQN)$.
- $M'_4 = h(ID_{tmp}||SQN_{tmp}||R')$.

Step 6. Compares M'_4 to the received M_4. If they match, the tag lets $ID_H = ID_{tmp}$ and $SQN = SQN_{tmp}$. If they do not match, the connection is terminated.

4. Security analysis

In this section, the security of the proposed scheme with respect to the aforementioned types of attacks, Section 1.1, is analyzed. Finally, a proof that the proposed protocol satisfies the security requirements presented in Section 1.2 is presented.

4.1. Attack analysis

- *Eavesdropping*: Throughout the SRFID protocol, the values the adversary can acquire via eavesdropping are R, ID, M_1, and M_2. Also, throughout the tag secret update protocol, the values the adversary can acquire via eavesdropping are R', R, ID, M_1, M_2, M_3, and M_4. The adversary tries to use this information to determine ID_H and SQN. As these values are protected by the hash function and XOR operation, they cannot be exposed by simple eavesdropping. Therefore, the proposed scheme is secure against eavesdropping.
- *Replay attack:* An adversary cannot reuse messages used in previous sessions because each response is a cryptographic function of a fresh random number. More specifically, M_1, M_2, M_3, and M_4 depend on R. Also, ID depends on SQN which is shared secrete between the tag and the backend server and updated each authentication session.
- *Tag cloning:* If an adversary wants to clone a genuine tag by creating a fake tag with the eavesdropping information, he needs to first query the tag and obtains a response which is ID and M_1. Then, the adversary places ID on a counterfeiting tag. The adversary will succeed if the RFID reader believes that the fake tag is a real one. However, in the proposed scheme, the real tag returns a different hashed value M_1 based on the random number R generated by the reader. Because the adversary cannot predict the random number R, the hashed value that the adversary obtains from the real tag (M_1) is not the same as the hashed value that the reader obtains. Thus, the adversary cannot clone a tag to fool the reader.

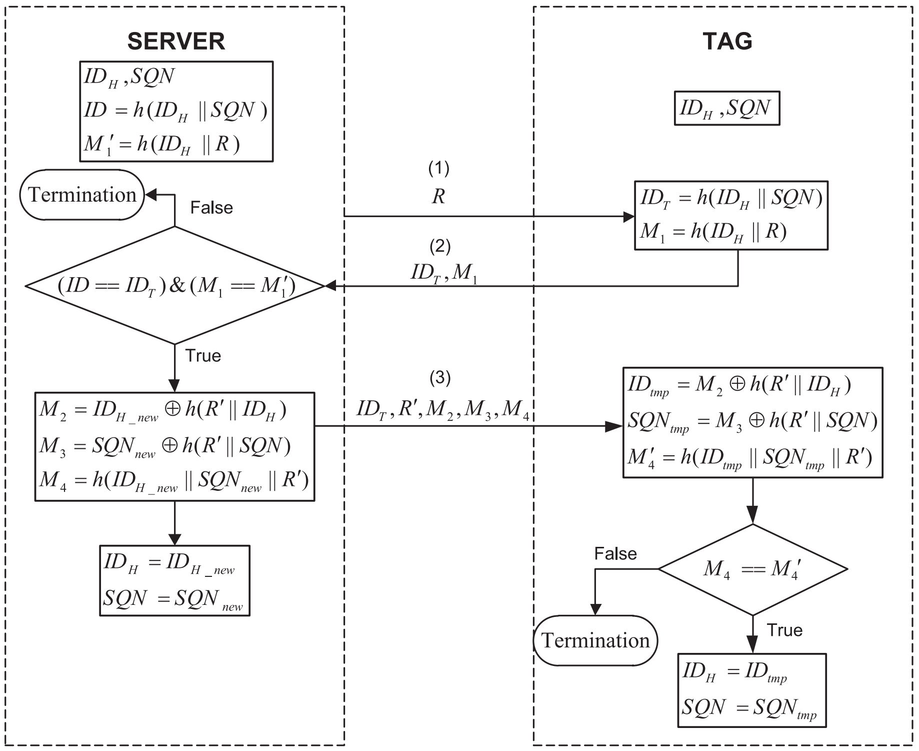
SERVER
TAG
ID_H, SQN
$ID = h(ID_H \| SQN)$
$M_1' = h(ID_H \| R)$
Termination
False
$(ID == ID_T) \& (M_1 == M_1')$
True
$M_2 = ID_{H\_new} \oplus h(R' \| ID_H)$
$M_3 = SQN_{new} \oplus h(R' \| SQN)$
$M_4 = h(ID_{H\_new} \| SQN_{new} \| R')$
$ID_H = ID_{H\_new}$
$SQN = SQN_{new}$
(1)
R
(2)
ID_T, M_1
(3)
ID_T, R', M_2, M_3, M_4
ID_H, SQN
$ID_T = h(ID_H \| SQN)$
$M_1 = h(ID_H \| R)$
$ID_{tmp} = M_2 \oplus h(R' \| ID_H)$
$SQN_{tmp} = M_3 \oplus h(R' \| SQN)$
$M_4' = h(ID_{tmp} \| SQN_{tmp} \| R')$
$M_4 == M_4'$
False
Termination
True
$ID_H = ID_{tmp}$
$SQN = SQN_{tmp}$

Figure 2 Tag secrets update protocol.

Table 2 Attacks resistance comparison against other schemes.

Attacks	Protocol					
	Shen [18]	Boyeon [19]	Chen [4]	Zhou [14]	Cho [8]	SRFID
Eavesdropping	√	√	√	√	√	√
Replay attack	√	-	√	√	√	√
Tag cloning	√	√	√	√	√	√
Tag tracing	-	√	*	*	*	*
Data forging	√	√	√	√	√	√
DoS attack	-	-	-	-	-	√
MITM attack	√	√	√	√	√	√

√: Resists such an attack.
*: Partially resists such an attack.
–: Does not protect against such an attack.

- *Tag tracing and user privacy violation:* In the proposed protocol, tags responses are random in each session. More specifically, *ID* is a hash of *SQN* which is secret number and is updated each authentication session. The identifier *ID* is also a hash of ID_H which is secret and is updated each authentication session using *SQN*. Thus, the adversary does not know which tag the response belongs to. Therefore, the location privacy is guaranteed. However, the scheme still allows a degree of tag tracking, because a tag always replies with the same hashed *ID* before the next successful authentication. An authorized reader can query the tag every time period *t* to reduce this type of tag tracing attack.
- *Data forging:* To modify the tag data, an adversary needs to authenticate himself to the tag. This is not possible, since the attacker needs to know ID_H and SQN to construct the message M_2 which is not possible. The attacker who wants to modify ID_H and SQN of a valid tag, using the tag secrets update protocol, needs to know both the current ID_H and SQN. The two secrets are hidden by the hash function and the XOR operations. In addition, the secret information stored on each tag is pertinent to itself. Unless the adversary broke the tag via physical method, he cannot know the secret information of the tag.
- *Denial of Service* (DoS) attack: The active attacker can intercept the fifth message, shown in Fig. 1, from the reader to the tag. Therefore, the backend server has refreshed the key, while the tag will not do it. Thus, the shared key between the backend server and the tag may not be the same. After a successful DoS attack which is caused by

desynchronization attack, the tag can never be legally authenticated. The desynchronization resistant mechanism discussed in Section 3.3.1 makes the protocol resistant to DoS attacks.

- *Man-in-the-Middle (MITM) attack:* An adversary cannot interfere with the exchanged messages by inserting or modifying messages, because of the use of the random number R and the secrets ID_H and SQN, which are only known to the backend server and the tag.

A simple comparison of several attacks resistance among the proposed SRFID protocol and five of the most recent protocols [4,8,14,18,19] is shown in Table 2.

4.2. Security requirements analysis

- *Mutual authentication:* In the proposed scheme, the backend server can authenticate a tag through a synchronized ID value, which is updated at every session and created by the secret values ID_H and SQN shared by both the backend server and the tag. The backend server also authenticates a tag through a synchronized M_1 value in the third message of Fig. 1, which is created by ID_H and a random fresh number R. Thus, an attacker without knowing ID_H and SQN cannot calculate it. The tag authenticates the backend server by computing $M'_2 = h(ID_{tmp}||SQN||R)$ and comparing it to the received M_2 value in the fifth message of Fig. 1 as described in Step 8 in Section 3.3.
- *Confidentiality*: The proposed protocol protects the information necessary for tag authentication by using the hash function, and guarantees that only the authenticated object knowing the tag ID_H and SQN pair can verify the information. Furthermore, as mentioned earlier, the proposed protocol is secure against eavesdropping by an adversary, and guarantees confidentiality by demanding that the complexity of a brute-force attack is high through the use of hash functions.
- *Indistinguishability*: The proposed scheme uses ID to identify tags. This ID is a hash of SQN which is secret number and is updated each authentication session. The identifier ID is also a hash of ID_H which is secret and updated each authentication session using SQN. So, it is impossible to anticipate the response message of the tag each session which guarantees indistinguishability.
- *Forward security:* The proposed protocol updates ID and ID_H every session using a one-way hash function. Even when an attacker has obtained the last ID_H and SQN values of a tag through capturing a tag in a physical way, he cannot calculate ID_H values of previous sessions due to the one-way property of hash functions ($ID_H = h(SQN||ID_H)$) and therefore cannot restore a message of a previous session (ID and M_1) to know or trace a user's past behaviors. Note that the tag stores only the last ID_H and SQN values.
- *Desynchronization resilience*. The desynchronization resistant mechanism discussed previously in Section 3.3.1 and Section 4.1 makes the protocol meet this requirement.
- *Tag delegation:* The tag delegation mechanism discussed in Section 3.4.1 makes the protocol meet this requirement. However, the entity receiving the delegation right cannot update the tag secrets, as it does not know ID_H or SQN. It also cannot calculate ID_H values of previous sessions due to the one-way property of hash functions.
- *Tag ownership transfer:* The tag ownership transfer mechanism discussed in Section 3.4.2 makes the protocol meet this requirement. In order to achieve ownership transfer of a tag T, server S must transfer the secrets ID_H and SQN to the new owner via a secure channel. Then, the new owner generates the new values ID_{H_new} and SQN_{new} and transfers them to the tag after encrypting them:

$$M_2 = ID_{H\_new} \oplus h(R'||ID_H)$$

$$M_3 = SQN_{new} \oplus h(R'||SQN)$$

This transfer needs to take place in an environment where there is no possibility of eavesdropping by the old owner [9]. The message M_4 in Fig. 2 proves to the tag that M_2 and M_3 are generated by the new owner, since only the tag owner and the tag can generate $h(R'||ID_H)$ and $h(R'||SQN)$.

A comparison of the security level among the proposed SRFID protocol and five of the most recent protocols [4,8,14,18,19] is shown in Table 3.

5. Performance evaluation

This section evaluates the performance of the proposed scheme in terms of computational cost, communication cost, and storage requirement. First, the computational cost for the

Table 3 Security level comparison against other schemes.

Security requirements	Protocol					
	Shen [18]	Boyeon [19]	Chen [4]	Zhou [14]	Cho [8]	SRFID
Mutual authentication	√	√	√	√	√	√
Confidentiality	√	√	√	√	√	√
Indistinguishability	√	√	√	√	√	√
Forward security	-	√	-	√	-	√
Anti-desynchronization	√	-	-	-	-	√
Tag delegation	-	-	-	-	-	√
Tag ownership transfer	-	-	-	-	-	√

√: Satisfies the requirement.
-: Does not satisfy the requirement.

Table 4 Overhead comparison against other schemes.

Protocol	Computation cost			Communication		Storage		
	T	R	S	$R \rightarrow T$	$T \rightarrow R$	T	R	S
Shen protocol [18]	$3H + RNG$	RNG	$O(\log m \log n) + 3H$	$l_R + l_H$	$2l_H + l_R$	$(m + 1)\, l_S + l_H$	l_R	$n((m + 1)\, l_S + l_H)$
Boyeon protocol [19]	$3H + RNG$	RNG	$O(n) + (n + 1)H$	$l_R + l_H$	$2l_H$	l_H	l_R	$2n(l_S + l_H)$
Chen protocol [4]	$4H$	RNG	$O(n) + (n + 3)H$	$l_R + l_H/2$	$5l_H/2$	$3l_H$	l_R	$3nl_H$
Zhou protocol [14]	$5H$	RNG	$O(\log n) + 3H + RNG$	$2l_R + l_H/2$	$5l_H/2$	$l_S + 2l_H$	l_R	$n(2l_S + 2l_H)$
Cho protocol [8]	$2H + 2MOD + RNG$	RNG	$O(n) + 2n(H + MOD)$	$2l_R + l_H$	$l_H + l_R$	$l_R + 2l_S$	l_R	$2nl_S$
SRFID protocol	$4H$	RNG	$O(1) + 4H$	$2l_R + l_H$	$2l_H$	$l_S + l_H$	l_R	$n(l_S + 3l_H)$

H: hash operation: RNG: random number generator operation: MOD: modular operation: l_H: length of the hash function output: l_R: length of RNG output: l_S: length of any stored value.

proposed scheme is examined. The tag T performs only four hash operations. These operations are low cost and can be effectively implemented on low cost RFIDs; lightweight cryptographic hash functions for implementing RFID protocols have been recently proposed, for example, Keccak and Quark lightweight hash functions [20,21].The server S requires $O(1)$ work to identify a tag in addition to four hash operation for mutual authentication. The reader R performs one random number generation operation.

Regarding the communication cost, only communication between the reader and the tag is considered. Table 4 shows the size of messages exchanged between the tag and the reader, which in total demands $2l_R + 3l_H$. See Table 4 for the meaning of l_R and l_H.

Regarding the storage requirement, each tag stores $l_S + l_H$ bits, the backend server stores $n(l_S + 3l_H)$ bits, where n is the number of tags and the reader stores l_R bits.

To analyze efficiency of the proposed *scheme*, the proposed scheme is compared with five of the most recent protocols [4,8,14,18,19] in Table 4. The comparison shows that the performance of the proposed protocol compares favorably with existing schemes.

6. Conclusion

In this paper, a hash-based security scheme for low cost RFID systems (SRFID) is propose. The proposed scheme seeks to mitigate the performance and security weaknesses of previous schemes. A security analysis of the proposed schemes is performed by comparing its ability to meet security requirements against several attacks and find that the schemes perform well against other previously proposed schemes in term of forward security, desynchronization resilience, secure tag delegation, and secure tag ownership transfer as shown in Tables 2 and 3. While the added security comes with some costs, it is found that the costs are reasonable and can be comparable or even lower than previously proposed schemes. The computation cost of the proposed scheme outperforms that of other schemes. A tag T requires only four hash operations. Lightweight cryptographic hash functions for implementing RFID protocols have been recently proposed [20,21]. The server S requires $O(1)$ work to identify a tag in addition to four hash operation for mutual authentication. The reader R performs one random number generation operation. The total size of messages exchanged between the tag and the reader in the proposed scheme ($2l_R + 3l_H$) which is the same or less than those of the schemes proposed in [4,8,14,18]. However, the total size of messages of the proposed scheme is less than that of [19] which is $l_R + 3l_H$, that is, the difference is just 32-bits, which is the size of the random number. The storage cost of the proposed scheme is $l_S + l_H$ bits for the tag, $n(l_S + 3l_H)$ bits for the backend server, and l_R bits for the reader. This outperforms the other schemes except the scheme presented in [19] in which the tag stores only l_S, that is, the difference is just the size of the digest.

The proposed scheme has two unique features supporting scalability; a backend server takes only $O(1)$ work for tag identification, and tag delegation and ownership transfer is straightforward. A protocol to update tag secrets for secure tag ownership transfer is also introduced.

References

[1] Henrici D. RFID security and privacy: concepts, protocols, and architectures. 1st ed. New York: Springer; 2008.

[2] Sun H-M, Ting W-C. A Gen2-based RFID authentication protocol for security and privacy. IEEE Trans Mob Comput 2009;8(August):1052–62.

[3] Liu AX, Bailey LA. PAP: a privacy and authentication protocol for passive RFID tags. Comput Commun 2009;32:1194–9.

[4] Chen L, Mu Y, Susilo W, Lim J, Oh H, Kim S. A new hash-based RFID mutual authentication protocol providing enhanced user privacy protection. In: Information security practice and experience, vol. 4991. Springer Berlin/Heidelberg; 2008. p. 278–89.

[5] Gavrilova M, Gervasi O, Kumar V, Tan C, Taniar D, Laganà A, et al. Hash-based RFID security protocol using randomly key-changed identification procedure. In: Computational science and its applications – ICCSA 2006, vol. 3983. Springer Berlin/Heidelberg; 2006. p. 296–305.

[6] Irfan S, Tharam D, Elizabeth C, Song H. A survey of RFID authentication protocols based on hash-chain method. In: Presented at the proceedings of the 2008 third international conference on convergence and hybrid information technology, vol. 02; 2008.

[7] Ohkubo M, Suzuki K, Kinoshita S. Cryptographic approach to "privacy-friendly" tags. In: Radio Frequency Identification (RFID) privacy, workshop; 2003.

[8] Cho JS, Yeo SS, Kim SK. Securing against brute-force attack: a hash-based RFID mutual authentication protocol using a secret value. Comput Commun 2011;34:391–7.

[9] Song B, Mitchell CJ. Scalable RFID security protocols supporting tag ownership transfer. Comput Commun 2011;34(April):556–66.

[10] Shao M-H. An approach to security and privacy of RFID systems in anti-desynchronization. IJCSNS Int J Comput Sci Netw Security 2010;10:40–4.

[11] Tong-Lee L. Secure RFID identification and authentication with triggered hash chain variants. In: 14th IEEE international conference on parallel and distributed systems; 2008. p. 583–90.

[12] Van Deursen T, Mauw S, Radomirović S, Vullers P. Secure ownership and ownership transfer in RFID systems. In: Presented at the proceedings of the 14th European conference on research in computer security, Saint-Malo, France; 2009.

[13] Weis S, Sarma S, Rivest R, Engels D. Security and privacy aspects of low-cost radio frequency identification systems. In: Security in pervasive computing, vol. 2802. Springer Berlin/Heidelberg; 2004. p. 50–9.

[14] Zhou S, Zhang Z, Luo Z, Wong E. A lightweight anti-desynchronization RFID authentication protocol. Inf Syst Front 2010;12:521–8.

[15] Gervasi O, Gavrilova M, Han S, Potdar V, Chang E. Mutual authentication protocol for RFID tags based on synchronized secret information with monitor. In: Computational science and its applications – ICCSA 2007, vol. 4707. Springer Berlin/Heidelberg; 2007. p. 227–38.

[16] Piramuthu S. RFID mutual authentication protocols. Decision Supp Syst 2011;50:387–93.

[17] Kaleb L. A two-step mutual authentication protocol based on randomized hash-lock for small RFID networks. In: 2010 Fourth international conference on network and system security; 2010. p. 527–33.

[18] Shen J, Moh DCS, Chung I. A novel anonymous RFID authentication protocol providing strong privacy and security. In: 2010 International conference on multimedia information networking and security; 2010. p. 584–8.

[19] Boyeon S, Chris JM. RFID authentication protocol for low-cost tags. In: Presented at the proceedings of the first ACM conference on wireless network security, Alexandria, VA, USA; 2008.

[20] Ors Yalcin S, Kavun E, Yalcin T. A lightweight implementation of Keccak hash function for radio-frequency identification applications. In: Radio frequency identification: security and privacy issues, vol. 6370. Springer Berlin/Heidelberg; 2010. p. 258–69.

[21] Aumasson J-P, Henzen L, Meier W, Naya-Plasencia M. Quark: a lightweight hash. In: Cryptographic hardware and embedded systems, CHES 2010, vol. 6225. Springer Berlin/Heidelberg; 2010. p. 1–15.