On the Security of Moessner's and Khan's Authentication Scheme for Passive EPCglobal C1G2 RFID Tags

Walid Khedr

Department of Information Technology, Faculty of Computers & Informatics, Zagazig University
Zagazig University, Zagazig 44519, Egypt
(Email: wkhedr@zu.edu.eg)

(Received Feb. 2, 2013; revised and accepted June 23, 2013)

Abstract

RFID technology is one of the most promising automatic data collection technologies. It uses radio waves to identify object. Through automatic and real-time data acquisition, this technology can give a great benefit to various industries by improving the efficiency of their operations. Due to the increasing popularity of RFID applications, different authentication schemes have been proposed to provide security and privacy protection for users. Recently, Moessner's and Khan's proposed an authentication scheme for passive RFID tags that can be embeddable into the ubiquitous EPCglobal C1G2 protocol in order to offers a high level of security through the combination of a random key scheme with a strong cryptography. In this paper a tag traceability attack and a server impersonation attack are presented. An improved scheme that eliminates these two attacks and decreases the storage requirements is also presented. These improvements introduce slight modification to the scheme.

Keywords: Authentication, hash function, privacy, RFID, security

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 [4]. Using RFID, the identification is performed electromagnetically. Unfortunately, RFID also introduces problems respecting data security and privacy arises. To solve these problems, many previous studies proposed solutions in diverse aspects [1, 3, 7, 9-12]

Recently, Moessner and Khan [6] proposed an authentication protocol that is based on symmetric key cryptography. The protocol is applicable to passive tags and it can be embedded with EPCglobal C1G2 standard protocol. The authors presented an implementation of their protocol on INTEL WISP UHF RFID tag and a C1G2 compliant reader [2]. It was designed to provide mutual authentication and assures forward and backward security. It was also designed to resist tracking, replay, DoS and MitM attacks. However, a tag traceability attack and a server impersonation attack were found. Using the tag tracing attack, attackers can either identify the same tag from passively logged messages or interact actively with the tag to understand its location [8]. On the other hand the server impersonation attack allows an adversary to eavesdrop a valid set sessions. The next time the protocol is run, the adversary impersonating the server can respond to the tag's messages and the tag would accept this as valid response. This leads to the reveal of tag secret information by the impersonating server [8]. The security of Moessner's and Khan's authentication scheme [6] relies on two key tables. The key tables are generated during the manufacturing process and are written on the tag. The storage requirements of the scheme are high for both server and the tag. This is due to the key tables with each having a size of 0.5K [6]. In this paper an improved scheme that decreases the storage requirements is also proposed.

The rest of this paper is organized as follows: Section 2 briefly reviews Moessner's and Khan's authentication scheme. The security analysis of Moessner's and Khan's scheme is presented in section 3. Section 4 presents the improved scheme. Section 5 shows the analysis of the improved scheme. Finally, section 6 concludes the paper.

2 Review of Moessner's and Khan's Scheme

In this section, we review Moessner's and Khan's authentication scheme for passive RFID tags. The notations listed in Table 1 are used throughout this paper. The scheme is based on symmetric key cryptography; it employs ciphers to hide messages contents. The security of the scheme relies on key tables (A and B) that are stored at the tag level, and ciphers that keep the message content secret. It also employs monotonically increasing timestamps to authenticate a tag. The key tables are generated during the manufacturing process, and along with a primary timestamp T_t and the tag's *ID* (EPC) they are written on the tag. These two key tables satisfy the following property: For a particular value of $Key_A[i]$,

Table 1: Notation

Symbol	Meaning
EPC	Electronic Product Code
ID	The tag's ID same as EPC
$a \oplus b$	XOR operation of *a* and *b*
$h(a,b)$	Encryption of a, with the key b.
$h^{-1}(a,b)$	Decryption of a, with the key b
m	A message, exchanged between (backend-server, reader) and tag
R	Random number
T_r	Current timestamp at the reader or backend-server
T_t	Timestamp stored on the tag.
SQN	Sequence number
Inc()	*SQN* increment function
ROTL(*x*, *n*)	Left rotation of *x* by *n* bits

there is only one unique value of key $Key_B[i+1]$ i.e. the key pair for a certain index *i* must be unique. The main idea of the scheme is that a tag can authenticate the reader/server as only an authentic entity can know the unique key pairs [6]. The scheme works as follows:

1. Reader → Tag: Random challenge $m_1 = R_r$
2. The tag generates a random challenge R_t. The tag use R_t to fetch two keys ($Key_A[R_t]$ and $Key_B[R_t+1]$) from its key tables. The first key ($Key_A[R_t]$) is employed to encrypt the tag's timestamp T_t and the challenge R_r: $h_1(T_t \| R_r, Key_A[R_t])$.
3. Tag → Reader: $m_2 = R_t \| h_1$
4. The reader fetches a subset of keys, K_A, from the database that matches the search criteria: Table = *A* and Index = R_t. The reader decrypts the tag response h_1 with each key of subset K_A until the decryption yields the reader's challenge R_r. As soon as a matching key is found, the reader fetches $Key_B[R_t+1]$ that is the other half of the unique key pair. This key is used to encrypt the tag's random number and the recent reader timestamp: $h_2(T_r \| R_t, Key_B[R_t+1])$. If the tag's response does not contain R_r then tag simply replies with a random number. $m_3 = R_{r_2}$
5. Reader → Tag: $m_3 = h_2$
6. After receiving m_3, the tag decrypts the message with the key $Key_B[R_t+1]$. The reader is approved by the tag if the decryption yields R_t since only a genuine reader can find the right key pair. If the reader's response does not contain R_t then tag simply replies with a random number $m_4 = R_{t_2}$. Otherwise, the reader's timestamp T_r is further examined. If it is greater than the tag's timestamp, the tag adopts the reader's timestamp and replies with $m_4 = h_3 = h(ID \oplus T_{t=r}, Key_B[R_t+1])$. If the reader's timestamp is not greater than the tag's timestamp, the tag does not update its timestamp and replies with $m_4 = h_3 = h(ID \oplus T_t, Key_A[R_t])$ [6].
7. Tag → Reader: $m_4 = h_3$

If the reader cannot find a pair of keys that matches with the ID and R_t or if it is expected to receive a random number from the tag then the tag is not authenticated. We urge the reader to consult the original paper [6] for details.

3 Analysis of Moessner's and Khan's Scheme

Moessner and Khan claimed that their scheme is secure against tracking and replay attacks. The security of the scheme relies on key tables that are stored at the tag level, and monotonically increasing timestamps to authenticate a tag. The tag generates a random number R_t and uses it to fetch two keys form the two tables by using R_t as an index, $Key_A[R_t]$ and $Key_B[R_t+1]$. Each of these tables has a size of 0.5K [6]. This means that each table stores 32 keys; since the key size is 128 bits. So, there are 32 possible values for R_t. Even if the whole 8 K EEPROM of the INTEL WISP tag are used to store the two tables, this number is extended to 256 possible keys. It was assumed that a hash value and a tag's ID have a length of b bits each, while key has a length of 2b bits. Timestamp, random number and a query message are assumed to have a length of 0.5b. So, if the key size is 128 bits, as assumed in the scheme, then the size of a hash value is 64 bits and the size of a random number is 32 bits. Also, the tag in this scheme cannot be expected to have a clock since the scheme is designed form passive RFID tags. Thus, it is unable to distinguish between a current and a dated timestamps. So, the tag relies on readers to update its timestamp T_t, so that it can distinguish between future and past timestamps.

Based on the above discussion, an adversary can track the tag or impersonate the server as long as its timestamp is not changed. This can be only happen between each two successive authentication sessions or when the tag is out of the reader range e.g. the tag leaves the store. At this point, various readers with different levels of security are assumed to be able to access the tag [5].

3.1 Tag Tracking

The adversary keeps challenging all n tags with $R_r = c$, where c is a constant value, until he gets 32 different replies form each tag. Each reply corresponds to one of the R_t 32 possible values. The adversary ends with 32 sets of messages 'm_2'. Each set contains n messages, where n is the number of tags, and each set corresponds to one of the R_t 32 possible values. The adversary constructs Table 2 that can be used to launch tag tracking attack, where $h_1[i,j] = h(T_{t_i} \| c, Key_A[R_{t_j}])$ is the second part of message m_2 send by tag i. It is clear that tag i will reply with one of the $h_1[i,j], j = 1, \cdots 32$ possible values if the tag is queried between each two successive rounds or when

Table 2: Adversary collected data for tag tracking

	R_{t_1}	……	R_{t_j}	……	R_{t_m}
Tag_1	$h_1[1,1]$	……	$h_1[1,j]$	……	$h_1[1,m]$
⋮	⋮	⋮	⋮	⋮	⋮
Tag_i	$h_1[i,1]$	……	$h_1[i,j]$	……	$h_1[i,m]$
⋮	⋮	⋮	⋮	⋮	⋮
Tag_n	$h_1[n,1]$	……	$h_1[n,j]$	……	$h_1[n,m]$

the tag is out of the reader range. So, $h_1[i,j]$ can be considered as tag$_i$'s ID when the tag replies with R_{t_j} in message m$_3$ i.e. we can assume that each tag has 32 possible IDs. Table 2 can be used to identify the tag as follows:

1. Adversary $\rightarrow Tag_i$: $R_r = c$
2. $Tag_i \rightarrow$ Adversary: $m_2 = R_{t_j} \| h_1$, where
   $h_1 = h[i,j] = R_{t_j} \| h(T_{t_i} \| c, Key_A[R_{t_j}])$
3. Using R_{t_j} received in step 2, the adversary searches column j of Table 2 for a matching Tag_i , which identifies the tag. It is clear that tag i will reply with the same time stamp T_{t_i} each query as long as it is out of the reader range e.g. the tag leaves the store.

3.2 Server Impersonation

Moessner's and Khan's scheme is also subject to server impersonation attack. The attack can be performed as follows: The adversary keeps recording R_t in step 3 of the protocol and the server reply messages m_3 in step 5 for each R_t until it collects all the possible 32 value of R_t for each tag. The adversary ends with 32 sets of m_3 messages. Each set corresponds to one of the R_t 32 possible values. The adversary constructs Table 3.

The next step the adversary must take before it is ready to impersonate the server is to determine which Tag_i received message $m_3[i,j]$ for each R_{t_j} . So, based on the assumption that the tag is queried between each two successive rounds or when the tag is out of the reader range, Tag_i will always reply with the same message $m_4 = h_3 = h(ID \oplus T_{t_i}, Key_A[R_{t_j}])$ if it receives the same $m_3[i,j]$. Based on the above discussion, the adversary determines Tag_i that received message $m_3[i,j]$ as follows:

1. Adversary $\rightarrow Tag_i$: $R_r = c$
2. $Tag_i \rightarrow$ Adversary: $m_2 = R_{t_j} \| h_1[i,j], 1 \le i \le n$
3. The adversary picks each message $m_3[i,j], 1 \le i \le n$ of column R_{t_j} of Table 3 and applies the following steps.
4. Adversary $\rightarrow Tag_i$: $m_3[i,j]$
5. When Tag_i receives $m_3[i,j]$, the tag decrypts it with the key $Key_B[R_t + 1]$.
   $$m_3[i,j] = h_2(T_r \| R_{t_j}, Key_B[R_{t_j} + 1])$$
   If the decryption yields the same R_t expected by the tag, the tag examines the timestamp T_r after the decryption of m_3. This value should be less than or equal the tag's timestamp; since it is part of a replay message. According to Moessner's and Khan's scheme, the tag does not update its timestamp and replies with $m_4 = h_3 = h(ID \oplus T_{t_i}, Key_A[R_{t_j}])$ if reader's timestamp is not greater than the tag's timestamp. If the decryption does not yield the same R_t expected by the tag, the tag replies with a random number.
6. $Tag_i \rightarrow$ Adversary: $m_4 = h_3$
7. When the adversary receives m_4 he computes its size. If the size is b (64 bits) i.e. the tag replied with h_3, the adversary associate $m_3[i,j]$ and m_4 with the tag identified by $h[i,j]$. If the size is 0.5b (32 bits) i.e. the tag replied with a random number, the adversary picks the next message $m_3[i+1,j]$ of column R_{t_j} of Table 3 and uses it in the next query.
8. The above steps are repeated until the adversary ends with Table 4.

Based on the above discussion, the adversary can impersonate the server as follows:

1. Adversary $\rightarrow Tag_i$: $R_r = c$
2. $Tag_i \rightarrow$ Adversary: $m_2 = R_{t_j} \| h_1$, where
   $h_1 = h[i,j] = R_{t_j} \| h(T_{t_i} \| c, Key_A[R_{t_j}])$
3. Using R_{t_j} received in step 2, the adversary searches column j of Table 4 for a matching tag identified by $h[i,j]$.
4. Adversary $\rightarrow Tag_i$: $m_3[i,j]$
5. When a Tag_i receives $m_3[i,j]$, the tag decrypts it with the key $Key_B[R_t + 1]$.
   $$m_3[i,j] = h_2(T_r \| R_{t_i}, Key_B[R_{t_j} + 1])$$
   The decryption should yield the same R_t expected by the tag. So, the tag examines the timestamp T_r after the decryption of m_3. This value should be less than or equal the tag's timestamp; since it is part of a replayed message. According to Moessner and Khan scheme, the tag does not update its timestamp and replies with $m_4 = h_3 = h(ID \oplus T_{t_i}, Key_A[R_{t_j}])$ if the reader's timestamp is not greater than the tag's timestamp.
6. $Tag_i \rightarrow$ Adversary: $m_4 = h_3$
7. When the adversary receives m_4, he compares it with $m_4[i,j]$. If they match, this means that the adversary is accepted by the tag as authenticated server. The

Table 3: Adversary collected data for server impersonation.

R_{t_1}	……	R_{t_j}	……	R_{t_m}
$m_3[1,1]$	……	$m_3[1,j]$	……	$m_3[1,m]$
⋮	⋮	⋮	⋮	⋮
$m_3[i,1]$	……	$m_3[i,j]$	……	$m_3[i,m]$
⋮	⋮	⋮	⋮	⋮
$m_3[n,1]$	……	$m_3[n,j]$	……	$m_3[n,m]$
⋮	⋮	⋮	⋮	⋮

Table 4: Adversary collected data for server impersonation and tag identification.

	Rt_1	……	Rt_j	……	Rt_m
Tag_1	$h_1[1,1]$, $m_3[1,1]$, $m_4[1,1]$	……	$h_1[1,j]$, $m_3[1,j]$, $m_4[1,j]$	……	$h_1[1,m]$, $m_3[1,m]$, $m_4[1,m]$
⋮	⋮	⋮	⋮	⋮	⋮
Tag_i	$h_1[i,1]$, $m_3[i,1]$, $m_4[i,1]$	……	$h_1[i,j]$, $m_3[i,j]$, $m_4[i,j]$	……	$h_1[i,m]$, $m_3[i,m]$, $m_4[i,m]$
⋮		⋮	⋮	⋮	⋮
Tag_n	$h_1[n,1]$, $m_3[n,1]$, $m_4[n,1]$	……	$h_1[n,j]$, $m_3[n,j]$, $m_4[n,j]$	……	$h_1[n,m]$, $m_3[n,m]$, $m_4[n,m]$

adversary does not need to decrypt m_4 to identify or track the tag; since the tag will reply with same message m_4 each time as along as the adversary query the tag between two successive rounds or when the tag is out of the reader range.

4 The Improved Scheme

To avoid the tag tracking and server impersonation attacks mentioned above, sequence numbers is used instead of timestamps. There are three secrete values shared between each tag and the server: K_A, K_B and SQN. Keys K_A and K_B are used instead of key tables (A and B) that are stored at the tag level. The keys K_A and K_B are chosen such that the unique key pair for each tag is generated by circularly shifting K_A and K_B i and $i+1$ bits to the left respectively. This reduces the storage space required by the original scheme to store the two key tables. The sequence number SQN is known only to the tag and the server and is updated each authentication session. 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 three values are generated by the server and are written in a secure manner into the tag's memory before deployment.

The proposed improvements introduce slight modification to the scheme. The detailed steps of the proposed improved scheme are presented as follows and illustrated in Figure 1.

1. Reader → Tag: Random challenge $m_1 = R_r$
2. The tag generates a random challenge R_t and use it to generate two ⋮ keys ($Key_A = ROTL(K_A, R_t)$ and $Key_B = ROTL(K_B, R_t + 1)$). The tag increments its sequence number ($SQN_t = Inc(SQN_t)$ and encrypt both R_t and SQN_t using the key Key_A : $h_1(SQN_t \parallel R_r, Key_A)$.
3. Tag → Reader: $m_2 = R_t \parallel h_1$
4. The reader generates a subset of keys, S_A: $ROTL(K_{A_i}, R_t)$ for each tag. The reader decrypts the tag response h_1 with each key of subset S_A until the decryption yields the reader's challenge R_r. As soon as a matching key is found, the reader checks if the increment of one of the associated sequence number $Inc(SQN_{t_i})$ is greater than or equal the received SQN_t. If this is not the case, the reader place an assumption about m_4 that DIFF > 0. Otherwise, DIFF is assumed to be equal zero. The value $DIFF$ is calculated by the tag in step 6 and is defined as the difference between the tag's sequence number SQN_t and the received reader's sequence number SQN_r. The reader update its sequence number SQN_r and generates $Key_B = ROTL(K_B, R_t + 1)$ that is the other half of the unique key pair. This key is used to encrypt the tag's random number R_t and the

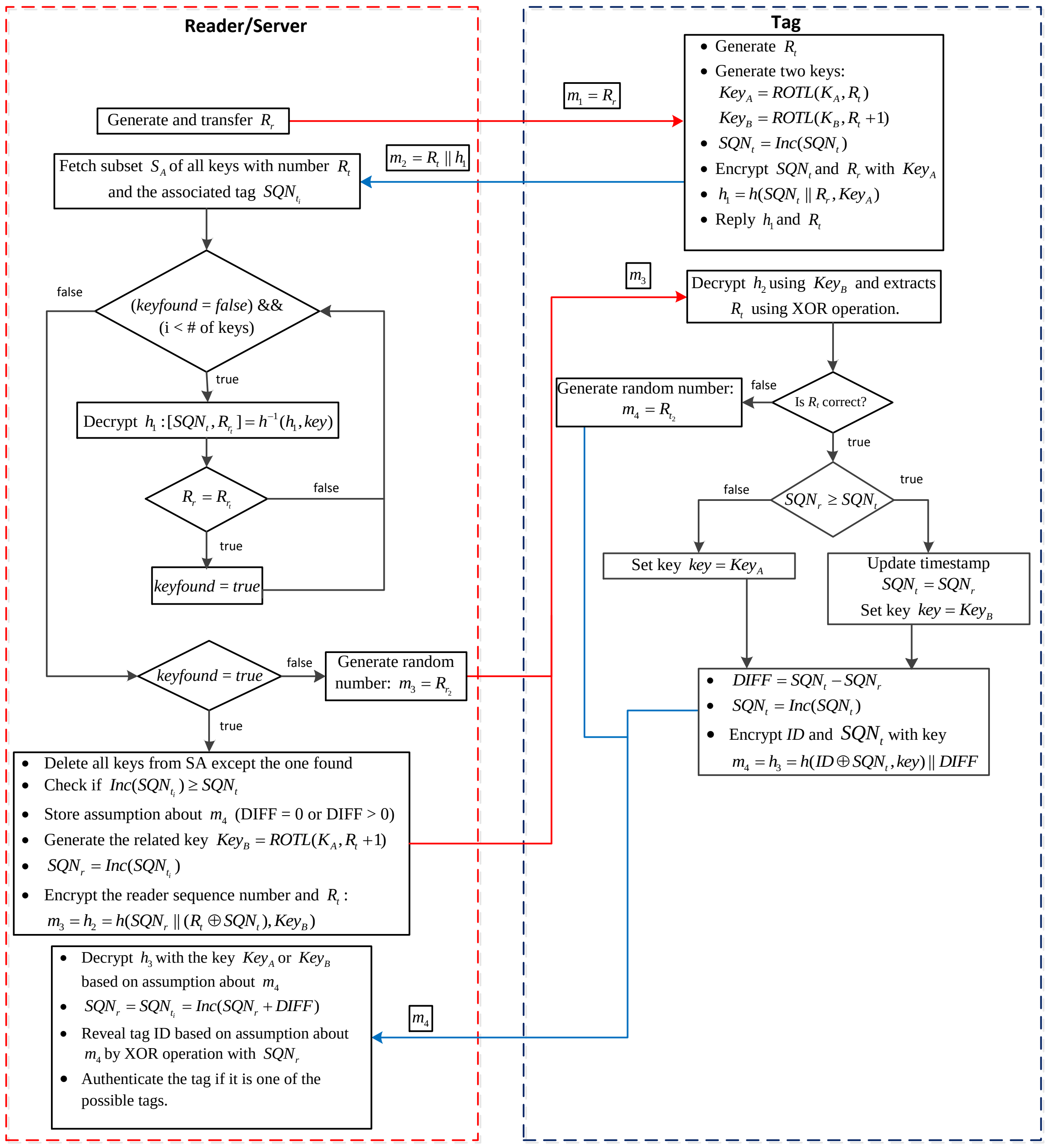
Reader/Server
Tag
Generate and transfer R_r
$m_1 = R_r$
Generate R_t
Generate two keys:
$Key_A = ROTL(K_A, R_t)$
$Key_B = ROTL(K_B, R_t + 1)$
$SQN_t = Inc(SQN_t)$
Encrypt SQN_t and R_r with Key_A
$h_1 = h(SQN_t \| R_r, Key_A)$
Reply h_1 and R_t
$m_2 = R_t \| h_1$
Fetch subset S_A of all keys with number R_t and the associated tag SQN_{t_i}
false
$(keyfound = false)$ && (i < # of keys)
true
Decrypt $h_1 : [SQN_t, R_{r_t}] = h^{-1}(h_1, key)$
$R_r = R_{r_t}$
false
true
$keyfound = true$
$keyfound = true$
false
Generate random number: $m_3 = R_{r_2}$
true
Delete all keys from SA except the one found
Check if $Inc(SQN_{t_i}) \geq SQN_t$
Store assumption about m_4 (DIFF = 0 or DIFF > 0)
Generate the related key $Key_B = ROTL(K_A, R_t + 1)$
$SQN_r = Inc(SQN_{t_i})$
Encrypt the reader sequence number and R_t :
$m_3 = h_2 = h(SQN_r \| (R_t \oplus SQN_t), Key_B)$
m_3
Decrypt h_2 using Key_B and extracts R_t using XOR operation.
Is R_t correct?
false
Generate random number: $m_4 = R_{t_2}$
true
false
$SQN_r \geq SQN_t$
true
Set key $key = Key_A$
Update timestamp $SQN_t = SQN_r$ Set key $key = Key_B$
$DIFF = SQN_t - SQN_r$
$SQN_t = Inc(SQN_t)$
Encrypt ID and SQN_t with key
$m_4 = h_3 = h(ID \oplus SQN_t, key) \| DIFF$
Decrypt h_3 with the key Key_A or Key_B based on assumption about m_4
$SQN_r = SQN_{t_i} = Inc(SQN_r + DIFF)$
Reveal tag ID based on assumption about m_4 by XOR operation with SQN_r
Authenticate the tag if it is one of the possible tags.
m_4

Figure 1: The improved scheme.

reader sequence number $SQN_r = Inc(SQN_{t_i})$: $h_2(SQN_r \| (R_t \oplus SQN_t), Key_B)$. If the tag's response does not contain R_r then reader simply replies with a random number, $m_3 = R_{r_2}$. The XOR operation is necessary to prevent the replay of the message m_3 which can be used to perform server impersonation attack as discussed in section 3.

5. Reader → Tag: $m_3 = h_2$

6. After receiving m_3, the tag decrypts the message with the key Key_B and extracts R_t using XOR operation. The reader is approved by the tag if the decryption yields R_t since only a genuine reader can find the right key pair and the current sequence number SQN_t. If the reader's response does not contain R_t then tag simply replies with a random number $m_4 = R_{t_2}$. Otherwise, the reader's sequence number SQN_r is further examined. If it is greater than or equal the tag's sequence number, the tag adopts the reader's sequence number and uses $Key = Key_B$ to generate m_4. If the reader's sequence number is less than the tag's sequence number, the tag does not update its sequence number and uses

$Key = Key_A$ to generate m_4. The tag computes $DIFF = SQN_t - SQN_r$ which equals to zero, if $SQN_r \geq SQN_t$, and is greater than zero otherwise. The tag encrypts its *ID* and its sequence number, after incrementing it, $SQN_t = Inc(SQN_t)$ with the key Key and replies with m_4.

$$m_4 = h_3 = h(ID \oplus SQN_t, Key) \| DIFF$$

The difference *DIFF* is used to enable the reader to calculate the tag's current sequence number. Since it is only a difference, no information that could be used by an attacker for unwanted recognition and tracking is revealed.

7. Tag → Reader: $m_4 = h_3$

8. After receiving m_4, the reader decrypt h_3 with the key Key_A or Key_B based on assumption about m_4 i.e. if $DIFF = 0$ then the reader decrypt h_3 using Key_B and if $DIFF > 0$ then the reader decrypt h_3 using Key_A. The reader is then increments its sequence number SQN_r after the addition of *DIFF* and updates the associated tag sequence number in its database: $SQN_r = SQN_{t_i} = Inc(SQN_r + DIFF)$. Finally the reader reveals the tag's ID based on assumption about m_4 by XOR operation with SQN_r and authenticates the tag if it is one of the possible tags.

5 Analysis of the Improved Scheme

In this section, the security of the proposed improved scheme with respect to the tag tracking and server impersonation attacks is analyzed.

5.1 Tag Tracking

In the original scheme, the response coming out from a tag *i*, after challenging it with $R_r = c$, belongs to one of the 32 sets of m_2 messages. Each set corresponds to one of the R_t 32 possible values. This response can be used to track the tag or the person holding the tag. This can be happen between each two successive rounds or when the tag is out of the reader range e.g. the tag leaves the store as discussed in section 3. In the proposed improved scheme each time a tag is queried, even using a constant value *c*, it replies with different m_2 message. This is true because h_1 depends on SQN_t which is incremented and encrypted each time the reader query the tag. To prevent tag-server desynchronization that could happen due to the fake challenge of tags or the change of the *DIFF* value, the improved scheme maintains the copies of the shared secrets ((K_A, K_B, SQN)) stored at the tag and the server in a consistent and synchronized state. This happens in step 6 and step 8 of the improved scheme. It should not be possible for an adversary to induce changes to the shared secrets that lead to an inconsistent or desynchronized state.

5.2 Server Impersonation

To launch a server impersonation attack, the adversary should construct Table 4. This could not happen in the improved scheme; since a tag replies with different m_2 message for each query, even if the tag generates the same random number R_{t_j}. If we assume that the adversary could construct Table 4, the replay of message m_3 does not allow the adversary to impersonate the server. After receiving $m_3 = h_2(SQN_r \| (R_t \oplus SQN_t), Key_B)$, the tag decrypts the message using the key Key_B and extracts R_t using XOR operation. If m_3 is a replayed message, the XOR operation does not yield the same R_t expected by the tag since the sequence number is changed. In this case the tag considers the response as a replayed message. So, the tag simply replies with a random number.

6 Conclusions

Recently, Moessner's and Khan's proposed an authentication scheme for passive RFID tags that can be embeddable into the ubiquitous EPCglobal C1G2 protocol in order to offers a high level of security through the combination of a random key scheme with a strong cryptography. However, their scheme is vulnerable to tag tracking attack and server impersonation attack. An improved scheme is proposed to avoid these two attacks and, thus, can be applied in environments requiring a high level of security. The improved scheme reduces storage requirements and maintains the same number of messages exchanged between the reader and the tag. The improved scheme also maintain the same messages size of m_2 and m_3, however there is a minor increase in message m_4 due to the concatenation of *DIFF*.

References

[1] T. Cao and P. Shen, "Cryptanalysis of two RFID authentication protocols," I*nternational journal of network security*, vol. 9, pp. 95-100, 2009.

[2] H. Chae, D. J. Yaeger, J. R. Smith, and K. Fu, "Maximalist cryptography and computation on the WISP UHF RFID Tag," in *Proceedings of the International Conference on RFID Security*, 2007.

[3] C. L. Chen, Y. L. Lai, C. C. Chen, Y. Y. Deng, and Y. C. Hwang, "RFID ownership transfer authorization systems conforming EPCglobal class-1 generation-2 standards," *International Journal of Network Security*, vol. 13, pp. 41-48, 2011.

[4] D. Henrici, *RFID Security and Privacy : Concepts, Protocols, and Architectures*, 1st ed. New York: Springer, 2008.

[5] A. X. Liu and L. A. Bailey, "PAP: A privacy and authentication protocol for passive RFID tags,"

Computer Communications, vol. 32, pp. 1194-1199, 2009.

[6] M. Moessner and G. N. Khan, “Secure authentication scheme for passive C1G2 RFID tags,” *Computer Networks*, vol. 56, pp. 273-286, 2012.

[7] M. Naveed, W. Habib, U. Masud, U. Ullah, and G. Ahmad, “Reliable and low cost RFID based authentication system for large scale deployment,” *International Journal of Network Security*, vol. 14, pp. 173-179, 2012.

[8] H. M. Sun and W. C. Ting, “A Gen2-based RFID authentication protocol for security and privacy,” *IEEE Transactions on Mobile Computing*, vol. 8, pp. 1052-1062, Aug. 2009 2009.

[9] C. H. Wei, M. S. Hwang, and A. Y. H. Chin, “An authentication protocol for low-cost RFID tags,” *International Journal of Mobile Communications*, vol. 9, pp. 208-223, 2011.

[10] C. H. Wei, M. S. Hwang, and A. Y. h. Chin, “A mutual authentication protocol for RFID,” *IT Professional*, vol. 13, pp. 20-24, 2011.

[11] C. H. Wei, M. S. Hwang, and A. Y. H. Chin, “An improved authentication protocol for mobile agent device in RFID environment,” *International Journal of Mobile Communications*, vol. 10, pp. 508-520, 2012.

[12] X. Zhang and B. King, “Security requirements for RFID computing systems,” *International Journal of Network Security*, vol. 6, pp. 214-226, 2008.

Walid Khedr received his Ph.D. degree in computer science from Ain Shams University in January 2009. He is currently working as assistant professor of computer science at Faculty of Computers and Informatics, Zagazig University. His current research interests are primarily in network security protocols, key management protocols, and RFID security. Another field of interest is quantum cryptography.