Analysis and Improvement of the securing RFID systems conforming to EPC Class 1 Generation 2 standard

Amin Mohammadali, Zahra Ahmadian, and Mohammad Reza Aref
Information Systems and Security Lab, Electrical Engineering Department,
Sharif University of Technology, Tehran, Iran.
E-mail: aminmohammadali@gmail.com, ahmadian@ee.sharif.edu, aref@sharif.edu

***Abstract*— Radio Frequency IDentification (RFID) technology is a wireless identification method in which security and privacy are important parameters for public acceptance and widespread use. In order to thwart such security and privacy problems, a wide variety of authentication protocols have been proposed in the literature. In 2010, Yeh et al's proposed a new RFID authentication protocol conforming to EPC Class 1 Generation 2 standard. They claimed that this protocol is secure against DoS attack, replay attack, DATA forgery attack, and provides untraceability and forward secrecy. In 2012, Yoon showed that this protocol does not provide forward secrecy and DATA integrity. He improved the protocol and tried to eliminate the weaknesses and claimd that the improved protocol does not have the weaknesses of the primary protocol. In this paper, we show that the improved protocol has some weaknesses including DoS attack, back-end server impersonation, tag impersonation and DATA forgery attack. We also show that it can not provide forward secrecy of the reader and untraceability. We improve the protocol, which offers a high level of security and provides mutual authentication, untraceability and forward secrecy as well as resistance to DATA forgery, replay and DoS attacks, while retaining a competitive communication cost.**

***Index Terms*— RFID authentication protocol, EPC, Forward secrecy, integrity.**

I. Introduction

RFID technology has many applications such as inventory control to supply chain management, traffic control, stock control, libraries and so on.

The RFID system consists of three parts: tag, reader and back-end server. In general terms, the tag comprises a wireless microchip with a very limited computational and storage capabilities, and a coupling element, such as an antenna coil for communication that can be used to identify the objects it is attached to. The reader is composed of a radio frequency (RF) module, a control unit and a coupling element. The reader communicates to the back-end server and the tag. The reader can carry out some complex cryptographic operations instead of the tag. The back-end server is usually composed of a database and a processing logic. It receives data from the readers, stores data into a database, and provides access to the data. Furthermore, the back-end server is assumed to have an infinite computational power [1], [2], [3].

The RFID technology is a replacement for the bar-codes; the bar-codes are very low-cost and have small sizes. In order to promote the great potential of the RFID technology, the cost of RFID tags should be able to compete with bar-codes [1]. This means RFID tags have cost limitations that restrains their resources and computational capabilities. Therefore, they cannot provide significant processing capabilities necessary for primitive cryptographic functions. To compensate the hardware restrictions, special attention should be paid to the design of the security protocol [2].

Tags can be classified according to their power and memory resources. A tag's memory is classified as read-only, write−once read−many and rewritable. In terms of power supply, tags are classified into three categories: passive, semi-passive and active. Tags are called passive if they have no power supplies, they receive their computational power from the electrical field generated by the reader. Semi-passive tags use a battery, but that battery is not for communication, instead, it is used to run the internal circuitry, and the energy of communication is provided by the reader. Active tags use a battery for both communication and running the internal circuitry [2], [4].

EPC Class 1 Generation 2 (EPC-C1-GEN2) has served as the most popular standard for passive tags. It supports on-chip 16-bit Pseudo-Random Number Generator (PRNG), and a 16-bit Cyclic redundancy Code (CRC) checksum is used to detect errors in the transmitted data. Despite the great benefits that EPC-C1-GEN2 offers in terms of communication compatibility and performance between the tags, the security level of the standard is extremly weak. The previous work on designing a security protocol for RFID based on EPC-C1-GEN2 suffers from security weaknesses [5], [6]. In [5], Chein et al. propose a mutual authentication protocol for improving the security performance of EPC-C1-GEN2, but their scheme suffers from some weaknesses including tag impersonation, back-end server impersonation, traceability and DoS attack and it also does not provide forward secrecy [7]. Yeh [6] et al. proposed a new RFID authentication protocol based on EPC-C1-GEN2 that does not provide DATA integrity and forward secrecy [8]. Yoon [8] imroved the Yeh et al.'s protocol and claimed that the improved protocol provides DATA integrity, untraceability and forward secrecy and resistance to replay attack, DoS attack and impersonation. We investigate this improved protocol and show it has some important weaknesses. We also try to eliminate weaknesses and present a new improved protocol.

The rest of the paper is organized as follows. In the next Section, we submit the privacy and security threats for RFID protocols. We introduce Yoon's protocol briefly in Section III. In SectionIV, we point out its security weaknesses. The improved protocol is presented in Section V while Section VI discusses the security and privacy of it. Finally, Section VII concludes the paper.

II. RFID PROTOCOL THREATS

A successful RFID technology is highly dependent on how security/privacy issues are addressed. The widespread employment of RFID systems into various identifications, increases security threats and the privacy problems. Some common types of attacks and threats on RFID systems include the following:

- **Privacy concern:** Privacy is a major concern in RFID systems, especially if this technology is interconnected to personal devices (like mobile phones, credit cards and e-passports). An authentication protocol must protect the privacy of the tag against illegitimate tracking and an adversary who wants to obtain information from the tag [2].
- **Replay attack:** In this attack, an adversary can replay (to the tag or the back-end server) the eavesdropped message between a reader and a tag without being detected, thereby performing a successful authentication to the tag or the reader [1], [9], [10].
- **DoS attack:** An adversary can block or modify the transmitted message between a reader and a tag. Consequently, the adversary can cause the reader and the tag to lose synchronization; therefore, they cannot communicate with each other. This attack can cause DoS attack between the tag and the back-end server [1], [9], [10], [11].
- **Impersonation attack:** An adversary can impersonate a reader (tag) to a tag (reader) if the tag (reader) does not recognize the adversary and accepts it as a legitimate reader (tag)[1], [9].
- **Forward secrecy concern:** If an adversary finds an internal state of a target tag (reader) at a time t, it should not be able to identify the target tag (reader) interactions which occurred at time $t' \langle t$. In other words, the previous tag's (reader's) sessions should not be distinguishable from the current tag's (reader's) sessions [1], [9], [10], [6], [12].
- **DATA integrity concern:** In RFID protocols, integrity assures the originality of the transmitted DATA and guarantees that it is not manipulated during the transition.

III. REVIEW OF YOON'S PROTOCOL

This section reviews the Yoon's protocol. For simplicity, we use notations of the original paper. These notations have been explained in TAB.I.

The information kept within respective devices:

- Tag: K_i, P_i, C_i, EPC_s
- Reader: RID
- Back-end server: $K_{old}, P_{old}, C_{old}, K_{new}, P_{new}, C_{new}, RID, EPC_s, DATA$

Yoon's protocol consists of two phases: the initialization phase, and the $(i+1)$th authentication phase (see also Fig. 1).

TABLE I
SYMBOL NOTATIONS.

Notation	Description
EPC_s	Electronic Product Code.
$DATA$	The corresponding record for the tag kept in the database.
K_i	The authentication key stored in the tag for the database. to authenticate the tag at the $(i+1)$th authentication phase
P_i	The access key stored in the tag for it to authenticate the back-end server at the $(i+1)$th authentication phase.
K_{old}	The old authentication key stored in the database.
K_{new}	The new authentication key stored in the database.
P_{old}	The old access key stored in the database.
P_{new}	The new access key stored in the database.
C_i	The database index stored in the tag to find the corresponding record of the tag in the database.
C_{old}	The old database index stored in the database.
C_{new}	The new database index stored in the database.
$A \rightarrow B$	A forwards a message to B.
N_Y	The random number generated by device Y.
$A \oplus B$	Message A is XORed with message B.
A_{adv}	Message A is producted by the adversary.
A'	Message of learning phase session.
A''	Message of attack phase session.
A_{pre}	The previous message of A.
A_{cur}	The current message of A.
RID	The reader identication number.
$H(.)$	Hash function.
$h_K(.)$	Keyed hash function.

A. Initialization phase

The manufacturer generates random values for K_0, P_0 and C_0, and sets the values for the record in the tag $(K_i = K_0, P_i = P_0, C_i = C_0)$ and the corresponding record in the database $K_{old} = K_{new} = K_0, P_{old} = P_{new} = P_0, C_{old} = C_{new} = 0)$.

B. Authentication phase

The authentication protocol operates as follows (see Fig.1).

1) **Reader $\rightarrow$Tag:** The reader generates random number N_R and sends it to the tag.
2) **Tag $\rightarrow$ Reader:** The tag generates random number N_T and computes $M_1 = PRNG(EPC_s \oplus N_R \oplus N_T) \oplus K_i$, $D = N_T \oplus K_i$ and $E = N_T \oplus PRNG(C_i \oplus K_i)$ based on the saved secret values. Then the tag sends M_1, D, E and C_i to the reader.
3) **Reader $\rightarrow$ Back-end server:** Upon receiving the messages by the reader, it computes $V = H(RID \oplus N_R)$ and forwards V as well as N_R and (M_1, D, C_i, E) to the back-end server. After receiving $(M_1, D, C_i,$ $E, V, N_R)$, the back-end server performs the following operations based on the saved information of each tag:
   a) For each RID_i in database, the back-end server computes $H(RID_i \oplus N_R)$ and compares it with the received V to identify the correct matching record and authenticate the reader.
   b) If $C_i = 0$, it stands for the first access. For each tuple $(K_{old}, P_{old}, C_{old}, K_{new}, P_{new}, C_{new}, EPC_s)$, the back-end server computes values $I_{old} = M1 \oplus K_{old}$ and $I_{new} = M_1 \oplus K_{new}$ in its database, and checks whether I_{old} or I_{new} matches $PRNG(EPC_s \oplus N_R \oplus D \oplus K)$ computed by the

Back-end server	Reader	Tag
	Generates: N_R	
	$\xrightarrow{N_R}$	Generate N_T
		$M_1 = PRNG(EPC_s \oplus N_R \oplus N_T) \oplus K_i$
		$D = N_T \oplus K_i$
		$E = N_T \oplus PRNG(C_i \oplus K_i)$
	$\xleftarrow{M_1, D, E, C_i}$	
	$V = H(N_R \oplus RID)$	
$\xleftarrow{M_1, D, E, C_i, V, N_R}$		
For each RID_i in DB		
Verifies $H(N_R \oplus RID_i) \overset{?}{=} V$		
If $C_i = 0$:		
For each tuple $(EPC_s, K_{old}, K_{new})$		
$I_{old} = M_1 \oplus K_{old}$		
$I_{new} = M_1 \oplus K_{new}$		
I_{new} or $I_{old} \overset{?}{=} PRNG(EPC_s \oplus N_R \oplus D \oplus K_X)$		
$X = old$ or new		
Else:		
C_{old} or $C_{new} \overset{?}{=} C_i$		
$X = old$ or new		
End if		
$PRNG(EPC_s \oplus N_R \oplus D \oplus K_X) \oplus K_X \overset{?}{=} M_1$		
Verifies $PRNG(C_i \oplus K_X) \oplus D \oplus K_X \overset{?}{=} E$		
$M_2 = PRNG(EPC_s \oplus N_T) \oplus P$		
$info = DATA \oplus RID$		
$MAC = H(DATA \oplus N_R)$		
if $X = new$		
$C_{old} \leftarrow C_{new} \leftarrow PRNG(N_T \oplus N_R)$		
$P_{old} \leftarrow P_{new} \leftarrow PRNG(P_{new})$		
$K_{old} \leftarrow K_{new} \leftarrow PRNG(K_{new})$		
Else:		
$C_{new} \leftarrow PRNG(N_T \oplus N_R)$		
End if		
$\xrightarrow{M_2, MAC, info}$		
	$info \oplus RID = DATA$	
	$H(N_R \oplus DATA) \overset{?}{=} MAC$	
	$\xrightarrow{M_2}$	
		Verifies $PRNG(EPC_s \oplus N_T) \oplus P_i \overset{?}{=} M_2$
		$C_{i+1} = PRNG(N_T \oplus N_R)$
		$P_{i+1} \leftarrow PRNG(P_i)$
		$K_{i+1} \leftarrow PRNG(K_i)$

Fig. 1. The Yoon's protocol

database itself. The process is iteratively repeated for each entry until it finds a match. Once the matching record is found, set value X as old or new according to which authentication key (K_{new} or K_{old}) in the record is found matched with the one in the tag.

c) When $C_i \neq 0$, the back-end server considers C_i as an index to find the corresponding record in the database. If $C_i = C_{old}$, the back-end server considers X as old, otherwise ($C_i = C_{new}$) X is considered as new. Then the back-end server investigates whether M_1 is equal to the computed value by the database itself ($PRNG(EPC_s \oplus N_R \oplus N_T) \oplus K_X \overset{?}{=} M_1$).

d) The back-end server uses K_X and D to obtain N_T, then checks whether the received E matches $N_T \oplus PRNG(C_X \oplus K_X)$ computed by the database itself. If the two values match, the back-end server verifies the tag, otherwise terminates the session.

4) **Back-end server → Reader:** The back-end server computes $M_2 = PRNG(EPC_s \oplus N_T) \oplus P_X, info = DATA \oplus RID$ and $MAC = H(DATA \oplus N_R)$, then sends it to the reader. If $X = old$, the back-end server updates just C_{i+1} as $PRNG(N_T \oplus N_R)$. Otherwise, it updates the record by replacing K_{old} with K_{new}, P_{old} with P_{new} and $C_{old} = C_{new}$. New values for K_{new}, P_{new} and C_{i+1} will be reset as $PRNG(K_{new})$, $PRNG(P_{new})$ and $C_{new} = PRNG(N_T \oplus N_R)$ respectively.

5) **Reader → Tag:** The reader XORs $info$ to RID itself

and obtains DATA, then checks whether the MAC received is equal to $H(DATA \oplus N_R)$ that is computed by itself. If the message of MAC is verified, the reader sends M_2 to the tag. After receiving M_2, the tag computes $PRNG(EPC_s \oplus N_T) \oplus P$ based on the saved values (N_T, P, EPC_s), then compares the recieved value with the computed value, if both are equal, then the tag authenticates the back-end server and updates $P_{i+1} = PRNG(P_i), K_{i+1} = PRNG(K_i)$ and $C_{i+1} = PRNG(N_T \oplus N_R)$ for the next session.

IV. Weaknesses of Yoon's Protocol

In [13], Safkhani et al. have showed that how the complexity of retrieving the secret parameters of Yoons protocol ($EPC_s; K_i$, and P_i) can be reduced to 3×2^L (instead of 2^{3L}) where L is the size of variables. They also presented a tracking attack on this protocol.

In this section we show that Yoons protocol has some further weaknesses including DoS attack, back-end server impersonation, tag impersonation and DATA forgery attack. We also show that it can not provide forward secrecy of the reader and untraceability. Of course, the presented tag impersonation and tracking attacks differs from that in [13].

A. DATA forgery attack

In the improved protocol, Yoon claims that a spoofed reader or an adversary cannot forge the transmitted DATA between the back-end server and the reader because it is protected using $MAC = H(DATA \oplus N_R)$. However, we show that this disadvantage still remains. To be more precise consider an adversary that follows the following steps:

Learning Phase: In the learning phase the adversary plays as an eavesdropper. She eavesdrops on one successful run of protocol and stores the exchanged messages between the back-end server and the legitimate reader including N_R and $V = H(RID \oplus N_R)$. We call the variables associated to this session A where A' is its counter part in the next session.

Attack Phase: The adversary waits until the reader initiates a new session of the protocol.

1) The reader generates a random number N'_R and sends it to the tag. Let $\delta = N_R \oplus N'_R$
2) The tag computes the messages and sends them to the reader.
3) Once the reader receives the messages, computes $V' = H(RID \oplus N'_R)$ and forwards $M'_1, D', E', C'_i, V', N'_R$ to the back-end server.
4) An adversary (the spoofed reader) blocks and changes the exchanged messages as follows:
   a) $M_{adv} = M'_1$
   b) $N_{R_{adv}} = N_R$
   c) $E_{adv} = E' \oplus \delta$
   d) $D_{adv} = D' \oplus \delta$
   e) $V_{adv} = H(RID \oplus N_R)$
   f) $C_{i_{adv}} = C_i$
5) The back-end server operates as follows:
   a) For each stored RID in the database, computes $H(RID \oplus N_{R_{adv}})$ and compares it with the received V_{adv}. So the adversary manipulates the transferred message from the reader to the back-end server and uses the previous N_R and V, therefore the back-end server authenticates the reader. Because, the back-end server has authenticated the reader with N_R and V in the learning phase.
   b) The back-end server uses C_i as an index to find the records in the database. After finding C_i and the records, the back end server finds that $PRNG(EPC_s \oplus N_{R_{adv}} \oplus D_{adv} \oplus K) \oplus K$ is equal to M_{adv}:

$$
\begin{aligned}
&PRNG(EPC_s \oplus N_{R_{adv}} \oplus D_{adv} \oplus K) \oplus K = \\
&PRNG(EPC_s \oplus N_R \oplus D' \oplus \delta \oplus K) \oplus K = \\
&PRNG(EPC_s \oplus N'_R \oplus D' \oplus K) \oplus K = M' = M_{adv}
\end{aligned}
$$

   According to the above equation the message of M_{adv} is verified by the back-end server.
6) In the next step, the back-end server checks whether $N_T \oplus PRNG(C_{i_{adv}} \oplus K)$ is equal to E_{adv}:

$$
\begin{aligned}
&D_{adv} \oplus K \oplus PRNG(C_{i_{adv}} \oplus K) = \\
&\quad D' \oplus \delta \oplus K \oplus PRNG(C_i \oplus K) = E' \oplus \delta = E_{adv}
\end{aligned}
$$

   This message is also verified by the back-end server. As a result, the back-end server authenticates the tag using incorrect values and then computes $M'_2, info', MAC'$ with the same incorrect values and sends them to the reader.
7) When the back-end server sends $(M'_2, Info', MAC')$ to the reader, the adversary intercepts them and computes forged values and forwards them to the reader. The forged values are as follows:
   a) $M_{2_{adv}} = M'_2$
   b) $info_{adv} = info' \oplus \delta$
   c) $MAC_{adv} = MAC'$
8) Upon receiving $info_{adv}, MAC_{adv}$ and $M_{2_{adv}}$, the reader computes $info_{adv} \oplus RID$ and obtains $DATA'$ that is a corrupted $DATA$. The reader computes $MAC = H(DATA' \oplus N_{R_{adv}})$. If the computed value is equal to the recived value, then the the redear authenticates the back-end server and forwards $M_{2_{adv}}$ to the tag. We can see that the $DATA$ result is not equal to the original $DATA$ which is sent by the back-end server as follows:

$$
\begin{gathered}
info_{adv} \oplus RID = \\
info' \oplus \delta \oplus RID = \\
DATA \oplus \delta \oplus RID \oplus RID = DATA \oplus \delta
\end{gathered}
$$

   On the other hand, the reader computes MAC and compares it with the recieved value of the back-end server:

$$
\begin{aligned}
MAC &= H(DATA \oplus \delta \oplus N'_R) \\
&= H(DATA \oplus \delta \oplus N_R \oplus \delta) = MAC'
\end{aligned}
$$

Therefore, the recived message of the back-end server is verified by the reader, while the incorrect data $(DATA \oplus \delta)$ is transferred to the reader. As a result, Yoon's protocol (similar to Yeh et al.s protocol) has DATA integrity problem because it cannot prevent verification of the forged data.

B. Back-end server (server) impersonation

This attack also has two phase like the previous one. In the learning phase the spoofed reader plays as an adversary and collectes the required information. She also blocks the message of M_2 to prevent the tag from updating its state. Then in the attack phase the spoofed reader impersonates the back-end server using the required information.

Learning phase: In this phase, a spoofed reader communicates with the tag as follows:

1) The spoofed reader generates $N'_R = 0$ and sends to the tag.
2) The tag computes $M_1 = PRNG(EPC_s \oplus N_T)$, $D = N_T \oplus K$ and $E = N_T \oplus PRNG(C_i \oplus K)$, then sends them to the reader together with C_i.
3) Upon receiving M_1, D, E, C_i, the spoofed reader computes $V = H(RID)$ and sends M_1, D, E, C_i, N_R, V to the back-end server.
4) Retrieves each stored RID sequentially to compute $H(RID \oplus N_R)$ with $N_R = 0$, and compares the producted V with the received V to identify the correct matching record and authenticates the reader. So, the value of V is computed by the legitimate reader, therefore the back-end server verifies the reader. On the other hand, the values of C_i, M_1, E and D are valid, because they are generated by the legitimate tag, so the back-end server verifies them.
5) After being authenticated by the tag, the back-end server computes $M_2 = PRNG(EPC_s \oplus N_T) \oplus P$, $info = DATA \oplus RID$ and $MAC = H(DATA)$ and sends them to the spoofed reader. The back-end server updates the private values as follows:
   a) $K_{i+1} = PRNG(K_i)$
   b) $P_{i+1} = PRNG(P_i)$
   c) $C_{i+1} = PRNG(N_T)$
6) The spoofed reader obtains M_2 and terminates the session.

Attack phase: After obtaining the information (namely: M_1, M_2), the spoofed reader starts a new session with the tag as follows:

1) The spoofed reader again sends $N_R = 0$ to the tag as a random number.
2) The tag computes $M'_1 = PRNG(EPC_s \oplus N'_T) \oplus K$, $D' = N'_T \oplus K$ and $E' = N'_T \oplus PRNG(C_i \oplus K)$, then sends them to the spoofed reader as well as C_i.
3) The spoofed reader computes $M'_2 = M_1 \oplus M_2 \oplus M'_1$ and sends it to the tag.
4) The tag computes its M'_2 and compares it with the recived value. We prove that if the message of $M'_2 = M_1 \oplus M_2 \oplus M'_1$ is transfered, the tag verifies it.

$$\begin{aligned} M_1 \oplus M_2 \oplus M'_1 &= PRNG(EPC_s \oplus N_T) \oplus K \\ &\oplus PRNG(N_T \oplus EPC) \oplus P \\ &\oplus PRNG(EPC_s \oplus N'_T) \oplus K \\ &= PRNG(EPC_s \oplus N'_T) \oplus P = M'_2 \end{aligned}$$

As a result, the back-end sever is impersonated by the spoofed reader. After being authenticated by the tag, the back-end server updates its as follows:
   a) $C_{i+1} = PRNG(N'_T)$
   b) $K_{i+1} = PRNG(K_i)$
   c) $P_{i+1} = PRNG(P_i)$

C. DoS attack

It is claimed that the back-end server can be resynchronized with the tag even if the confirmation message M_2 is blocked or incorrectly received, for that, the back-end server maintains the new and the old secret values of tags. However, we show that an adversary can perform desynchronization between the back-end server and the tag and cause DoS attack.
In the back-end server impersonation attack, the tag updates the value of index $C_{i+1} = PRNG(N'_T)$, but the back-end server updates $C_{i+1} = PRNG(N_T)$. In the next session, the back-end server does not verify the tag, therefore desynchronozation attack takes place.

D. Tag impersonation

We present a simple impersonation attack against the improved protocol. We show that an adversary can impersonate a valid tag permanently from a single eavesdropping and an active query to the tag.

Learning phase: In this phase, an adversary obtains the required information as follows:

1) The adversary sends $N_R = 0$ to the tag as a random number.
2) The tag computes $M_1 = PRNG(EPC_s \oplus N_T) \oplus K$, $D = N_T \oplus K$ and $E = N_T \oplus PRNG(C_i \oplus K)$, then sends them to the adversary together with C_i and she saves these messages. Then the adversary terminates session

Attack phase: In the attack phase, the adversary plays as a blocker. It firstly blocks the recieved message of the reader, then uses the eavesdroped messages of the learning phase and tries to impersonate the tag.

1) The reader generates a random number N'_R and sends it to the tag.
2) The adversary blocks this message and computes $D_{adv} = D \oplus N'_R$ and $E_{adv} = E \oplus N'_R$, then sends $M_{1_{adv}} = M_1$, D_{adv} and E_{adv} to the reader as well as $C_{i_{adv}} = C_i$.
3) The reader sends $M_{1_{adv}}$, D_{adv}, E_{adv}, N'_R and $V = H(N'_R \oplus RID)$ to the back-end server.
4) Upon receiving the message, the back-end server proceeds as follows:
   a) For each stored RID in the database, computes $H(RID \oplus N'_R)$ and compares it with the received

V. Since the adversary has not manipulated the transferred message from the reader to the back-end server, the back-end server authenticates the reader.

b) The back-end server computes its M_1 by using EPC, K, D_{adv} and N'_R:

$$PRNG(EPC_s \oplus D_{adv} \oplus K_{old} \oplus N'_R) \oplus K_{old} =$$
$$PRNG(EPC_s \oplus D \oplus N'_R \oplus K_{old} \oplus N'_R) \oplus K_{old} =$$
$$PRNG(EPC_s \oplus D \oplus K_{old}) \oplus K_{old}$$

then the back-end server compares the computed value with the recived value $M_{1_{adv}}$, that according to the above equation are equal.

c) The back-end server also compares the recived value of E_{adv} and the computed value by the back-end server:

$$D_{adv} \oplus K_{old} \oplus PRNG(C_{i_{adv}} \oplus K_{old}) =$$
$$D \oplus N'_R \oplus K_{old} \oplus PRNG(C_{i_{adv}} \oplus K_{old}) = E_{adv}$$

that according to the above equation are equal. Therefore, the adversary authenticates itself to the back-end server. As a result, tag impersonation is occurred. This attack can perform permanently without Additional computations.

E. Traceability Attack

The success of this attack depends on preventing tag index updating. The adversary performs this attack as follows:

Learning phase:

1) An adversary sends a random number to the tag.
2) The adversary receives C_i, M_1, D and E from the tag and terminates the session without updating the tag.

Attack phase: To track the target tag, the adversary waits until the reader initiates a new session of the protocol.

1) The reader generates a random number N'_R and sends it to the tag.
2) The tag computes M'_1, D' and E', and sends them as well as C'_i to the reader. The value of C'_i is equal to C_i, because in the learning phase, the session was terminated faulty and the tag did not update itself, as a result the adversary simply tracks the tag.

F. Reader forward secrecy compromise

In [12], Doss et al. showed that Yeh et al.'s protocol does not observe reader forward secrecy. We show this weakness is maintained in Yoon's protocol. It is a requirement that if a secret value of the reader is compromised, the adversary should not be able to track the previous communications of the reader. In order to perform this attack, the adversary learns N_R by eavesdropping on the channel between the tag and the reader, then uses RID obtained by compromising the reader and checks whether $V \overset{?}{=} H(RID \oplus N_R)$. Hence, Yoon's protocol does not protect the forward secrecy of the reader.

V. Improved protocol

In this section we briefly point out the weaknesses of Yoon's protocol and then propose our revisions to eliminate them.

The most important problem of Yoon's protocol is using $N_T \oplus N_R$. Because, the adversary can change the message D and E while keeping $N_T \oplus N_R$,and consequently M_1, unchanged. Another weakness lies in updating C_i after a unsuccessful session.

In order to frustrate these attacks, we change the way of computing M_1 and E. To avoid reader forward secrecy and other mentioned attacks, we also improve the mutual authentication protocol between the reader and back-end server. Besides, our improved protocol updates C_i after all sessions whether successful or not. We use notations of Yoon's protocol. The steps of our improved protocol are as follows (see Fig. 2).

The information kept within respective devices:

- Tag: K_i, P_i, C_i, EPC_s
- Reader: RID
- Back-end server: $K_{old}, P_{old}, K_{new}, P_{new}, C_i, RID_{old}, RID_{new}, EPC_s$ and $DATA$

A. Initialization phase

The manufacturer generates random values for K_0, P_0 and C_0, and sets the values for the record in the tag ($K_i = K_0, P_i = P_0, C_i = C_0$) and the corresponding record in the database $K_{old} = K_{new} = K_0, P_{old} = P_{new} = P_0, C_i = 0$).

B. Authentication phase

1) **Reader → Tag:** The reader generates random number N_R and sends it to the tag.
2) **Tag → Reader:** The tag generates random number N_T and computes $M_1 = PRNG(EPC_s \oplus N_R) \oplus PRNG(N_T) \oplus K_i$ and $D = N_T \oplus K_i$. Then the tag checks the $flag$, if $flag = 0$ considers the saved C_i, otherwise $C_i = C_i \oplus N'_T$ and computes $E = N_T \oplus PRNG(C_i \oplus K_i) \oplus P_i$. Where, N'_T is a new random number to prevent traceability attack. After transfering messages (C_i, D, E, M_1) to the reader, the flag is set to 1.
3) **Reader → Back-end server:** Upon receiving the messages, the reader computes $V = H(RID \oplus N_R \oplus M_1)$ and forwards it as well as N_R and (M_1, D, C_i, E) to the back-end server. After receiving (M_1, D, C_i, E, V, N_R), the back-end server performs the following operations based on the saved information of each tag:
   a) The back-end server searches its look-up table for a value RID_i that satisfies $V \overset{?}{=} H(RID_i \oplus N_R \oplus M_1)$. If such a value is found, the back-end server authenticates the reader.
   b) The back-end server searches its look-up table for the value of C_i. If such a value is found, then the back-end server investigates whether M_1 is equal to the computed value by the database itself

```
Back-end server                                  Reader                                  Tag
                                                 Generates: $N_R$
                                                          $\xrightarrow{N_R}$                 Generate $N_T$
                                                                  $M_1 = PRNG(EPC_s \oplus N_R) \oplus PRNG(N_T) \oplus K_i$
                                                                                            $D = N_T \oplus K_i$
                                                                                  If $flag = 0$, considers $C_i$
                                                                                        Else generates $N'_T$
                                                                                          $C_i = C_i \oplus N'_T$
                                                                             $E = N_T \oplus PRNG(C_i \oplus K_i) \oplus P_i$
                                                                                                  $flag = 1$
                                                          $\xleftarrow{M_1, D, E, C_i}$
                                        $V = H(N_R \oplus RID \oplus M_1)$
                   $\xleftarrow{M_1, D, E, C_i, V, N_R}$
For each $RID_i$ in DB
Verifies $H(N_R \oplus RID_i \oplus M_1) \overset{?}{=} V$
If $C_i$ is verified
    $PRNG(EPC_s \oplus N_R) \oplus PRNG(D \oplus K_{new}) \oplus K_{new} \overset{?}{=} M_1$
    $X = new$
Else:
    $I_{old} = M_1 \oplus K_{old} \oplus PRNG(D \oplus K_{old})$
    $I_{old} \overset{?}{=} PRNG(EPC_s \oplus N_R)$
    $X = old$
End if
If $C_i = 0$:
    For each tuple $(EPC_s, K_{old}, K_{new})$
    $I_{old} = M_1 \oplus K_{old} \oplus PRNG(D \oplus K_{old})$
    $I_{new} = M_1 \oplus K_{new} \oplus PRNG(D \oplus K_{new})$
    $I_{new}$ or $I_{old} \overset{?}{=} PRNG(EPC_s \oplus N_R)$
    $X = old$ or $new$
End if
    Verifies $PRNG(C_i \oplus K_X) \oplus D \oplus K_X \oplus P_X \overset{?}{=} E$

If $RID_i = RID_{new}$: considers $RID_{new}$
Else: $RID_{old}$
End if
Then computes values below:
  $M_2 = PRNG(EPC_s \oplus N_T) \oplus P_X$
  $info = DATA \oplus RID \oplus h_{RID}(N_R)$
  $MAC = H(DATA \oplus N_R)$
if $X = new$:
    $C_{old} \leftarrow C_{new} \leftarrow PRNG(N_T \oplus N_R)$
    $P_{old} \leftarrow P_{new} \leftarrow PRNG(P_{new})$
    $K_{old} \leftarrow K_{new} \leftarrow PRNG(K_{new})$
Else:
    $C_{new} \leftarrow PRNG(N_T \oplus N_R)$
End if
If $RID_i = RID_{new}$
    $RID_{old} \leftarrow RID_{new} \leftarrow H(RID_{new})$
Else
    Does not update the secret value of the reader.
End if
                   $\xrightarrow{M_2, MAC, info}$
                                        $info \oplus RID \oplus h_{RID}(N_R) = DATA$
                                          $H(N_R \oplus DATA) \overset{?}{=} MAC$
                                               $RID \leftarrow H(RID)$
                                                          $\xrightarrow{M_2}$
                                                                  Verifies $PRNG(EPC_s \oplus N_T) \oplus P \overset{?}{=} M_2$
                                                                                $C_{i+1} = PRNG(N_T \oplus N_R)$
                                                                                    $P_{i+1} \leftarrow PRNG(P_i)$
                                                                                    $K_{i+1} \leftarrow PRNG(K_i)$
                                                                                                  $flag = 0$
```

Fig. 2. The improved protocol

$(PRNG(EPC_s \oplus N_R \oplus N_T) \oplus K_{new})$. If such a value is satisfied, the back-end server verifies the tag and considers X as new.

c) If after searching look-up table, the back-end server does not find match for C_i, it for each tuple $(K_{old}, P_{old}, C_i, EPC_s)$ in its database computes the value $I_{old} = M1 \oplus K_{old} \oplus PRNG(D \oplus K_{old})$ and checks whether I_{old} matches $PRNG(EPC_s \oplus N_R)$. The process is iteratively repeated for each entry until it finds a match. Once the matching record is found, it sets value X as old.

d) If $C_i = 0$, the back-end server for each tuple $(K_{old}, P_{old}, K_{new}, P_{new}, C_i, RID, EPC_s, DATA)$ in its database computes values $I_{old} = M1 \oplus K_{old} \oplus PRNG(D \oplus K_{old})$ and $I_{new} = M_1 \oplus K_{new} \oplus PRNG(D \oplus K_{new})$, and checks whether I_{old} or I_{new} matches $PRNG(EPC_s \oplus N_R)$ computed by the database itself. The process is iteratively repeated for each entry until it finds a match. Once the matching record is found, it sets value X as old or new according to which authentication key (K_{new} or K_{old}) in the record is found matched with the one in the tag.

e) The back-end server uses K_X and D to obtain N_T, then checks whether the received E matches the computed $D \oplus K_X \oplus PRNG(C_i \oplus K_X) \oplus P_X$ by the database itself. If the two values match, the back-end server verifies the tag, otherwise terminates the session.

4) **Back-end server $\rightarrow$ Reader:** The back-end server computes $M_2 = PRNG(EPC_s \oplus N_T) \oplus P_X, info = DATA \oplus RID_i \oplus h_{RID}(N_R)$ and $MAC = H(DATA \oplus N_R)$, then sends them to the reader. If authenticating the reader $RID_i = RID_{new}$, the back-end server uses RID_{new} to compute $info$ and MAC, otherwise, $RID_i = RID_{old}$. If $X = old$, the back-end server updates just C_{i+1} as $PRNG(N_T \oplus N_R)$. Otherwise, it updates the record by replacing K_{old} with K_{new} and P_{old} with P_{new}. New values for K_{new}, P_{new} and C_{i+1} will be reset as $PRNG(K_{new})$, $PRNG(P_{new})$ and $C_{i+1} = PRNG(N_T \oplus N_R)$ respectively. On the other hand, to updating secret value of the reader, if in authentication of the reader $RID_i = RID_{new}$, the back-end server updates itself by replacing RID_{old} with RID_{new} and RID_{new} with $H(RID_{new})$. Otherwise, the back-end server does not update the secret value of the reader.

5) **Reader $\rightarrow$ Tag:** The reader by means of $info, RID$, and $h_{RID}(N_R)$ obtains $DATA$, then checks whether the received MAC is equal to $H(DATA \oplus N_R)$, if the message MAC is verified, the reader sends M_2 to the tag and updates secret value itself $RID = H(RID)$. After receiving M_2, the tag computes $N_T \oplus PRNG(EPC_s \oplus N_T) \oplus P$ based on the saved value (N_T, P, EPC_s), then compares the recieved value with the computed value. If both are equal, the tag authenticates the back-end server and updates $P_{i+1} = PRNG(P_i), K_{i+1} = PRNG(K_i), C_{i+1} = PRNG(N_T \oplus N_R)$ and $flag = 0$ for the next session.

VI. Analysis

In this section, we analayze how our protocol resists against privacy and security attack.

A. Tag impersonation

In order to impersonate a tag, an adversary should successfully generate M_1, D, E, and C_i. in order to compute M_1, the adversary needs EPC_s, K_i and N_T that are unkown for the adversary. On the other hand, the current value of M_1 is independent of the previous value of $M_{1_{pre}} = PRNG(EPC_s \oplus N_{R_{pre}}) \oplus PRNG(N_{T_{pre}}) \oplus K_{i_{pre}}$ and the adversary can not compute M_1 of the previous value. To compute D and E like M_1 the adversary can not use the previous values of D_{pre} and E_{pre} respectively. Therefore, the adversary can not impersonate the tag.

B. Tag forward secrecy

In order to prove forward secrecy we show that even if the tag is compromised by the adversary and its current secret values are obtained, the adversary is not be able to trace any of the previous authentication sessions. Assume that the current secret values are K_i, P_i, C_i and the previous messages of the tag are denoted by $M_{1_{pre}}, D_{pre}, E_{pre}$. Where N_T is generated by the tag that is unknown for the adversary. Further, $M_{1_{pre}}, D_{pre}$ and E_{pre} are computed from $K_{i_{pre}}, P_{i_{pre}}, N_T$. It is clear that all tag messages are independent of the current secret values and hence $M_{1_{pre}}, D_{pre}$ and E_{pre} cannot be identified by the adversary that know the current secret values.

C. Back-end server (server) impersonation

The improved protocol is protected against back-end server impersonation which can cause a tag to reveal its information to an adversary. In order to impersonate a back-end server, the adversary should successfully generate $M_2, info, MAC$. in order to comput M_2, it needs EPC_s and P_i that are secret. On the other hand, without knowing of RID and $DATA$ the adversary can not generate $info$ and MAC. Therefore, the adversary can not perform back-end server impersonation attack.

D. Replay attack

The improved protocol resists against replay attack because of the challenge-response scheme that is used in the protocol. In addition, for each session of the protocol a new pair of random number (N_R, N_T) are used. This prevents to use the same challenge-response value in other sessions.

TABLE II
COMPARE OF PROTOCOLS

	Chein [5]	Yeh [6]	Yoon [8]	This paper
Untraceable	●	●	●	$\surd$
Replay attack resistance	$\surd$	$\surd$	$\surd$	$\surd$
Tag impersonation resistance	●	$\surd$	●	$\surd$
Server impersonation resistance	●	$\surd$	●	$\surd$
DoS attack resistance	●	$\surd$	●	$\surd$
Tag forward secrecy	●	●	$\surd$	$\surd$
DATA integrity	●	●	●	$\surd$
Reader forward secrecy	N/A	●	●	$\surd$
Secret parameters disclosure	$\surd$	$\surd$	●	$\surd$

● = not satisfield, $\surd$= fully satifield and N/A= not applicable.

E. Untraceability

In Yoon and Yeh et al.'s protocols, an adversary tracks the tags, because the tag does not update C_i after an unsuccessful session too. But, in our improved protocol C_i is updated after each unsuccessful session. After the unsuccessful session, if the reader (or the adversary) starts a new session, the tag will update C_i by using N'_T that is generated by the tag and unknown even for back-end server.

Another way for tracking in Yoon's protocol is to exploit the unchanged value of $E \oplus D$ after an unsuccessful session. To prevent this attack, we improved E. After the unsuccessful session the tag uses of $C_i = C_i \oplus N'_T$ instead C_i for computing E, as a result, It is clear that the current $E \oplus D$ are independent of the prevous respons.

F. DoS attack

An adversary can cause DoS by making desynchronisation between the back-end server and the tag by either blocking or successfully forging flow.

Our improved protocol provides strong resistance against DoS attack. For example, the adversary can intercept the last flow of the protocol during authentication between the tag and the back-end server. This causes desynchronization of the tag secrets and the back-end server. But such desynchronization is resolved by storing previous secrets for each tag in the database, therefore, the tag and the back-end server are resynchronized in the next session. On the other hand, by changing each message, the back-end server and tag does not verify them. Consequently, desynchronization does not happen in our improved protocol.

G. Reader forward secrecy

In order to prove forward secrecy we show that even if the reader is compromised by the adversary and its current secret value is obtained, this cannot enable tracing of any previous communication. Assume that the current secret values is RID_{cur} and the previous message of the reader is denoted by $V_{pre} = h(N_{R_{pre}} \oplus RID_{pre} \oplus M1_{pre})$. It is clear that the reader message is independent of the current secret value and hence V_{pre} cannot be identified by the adversary that know the current secret value.

H. DATA integrity

In the improved protocol, the adversary can not perform DATA forgery attack, because if it changes any of the messages (M_1, D, E, V, N_R), the back-end server does not verify them and authentication process is remained incompelet, as a result the back-end server does not transfer no messages. On the other hand, by changing $info = DATA \oplus RID \oplus h_{RID}(N_R)$ the reader does not verify $MAC = h(N_R \oplus DATA)$, therefor, DATA forgery can not be occured.

I. Secret parameters disclosure

In [13], Safkhani et al. showed that the adversary can compromise K_i and N_T as follows:

1) The adversary eavesdrops one session of protocol and stores all transferred messages include: $N_R, C_i, M_1 = PRNG(EPC_s \oplus N_R \oplus N_T) \oplus K_i$, $D = N_T \oplus K_i$, $E = N_T \oplus PRNG(C_i \oplus K_i)$, $M_2 = PRNG(EPC_s \oplus N_T) \oplus P_X$.
2) $\forall i = 0...N_d$ does as follows:
   - $K_i \longleftarrow i$,
   - $N_T \longleftarrow D \oplus K_i$,
   - If $E = N_T \oplus PRNG(C_i \oplus K_i)$ then returns K_i and N_T.

   The adversary also uses M_1, M_2 and like the previous state computes EPC_s and P_i respectively.
   But in the improved protocol, we add P_i to E, therefore the adversary by having K_i can not check E, because of it needs to P_i for checking E. On the other hand, without knowing K_i and N_T the adversary can not obtain other secret values, consequently it can not compromise secret values.

We also compare our protocol with the existing protocol conforming EPC-C1-GEN2 in terms of security and privacy. The result are depicted in TAB.II.

VII. Conclusion

Due to the importance of the security and privacy for EPC-C1-GEN2, many protocol have been proposed trying to solve security and privacy problems. Yoon tried to improve the Yeh et al.'s protocol and presented an improved protocol. In this paper, after breifly presenting the Yoon's protocol, the security of his protocol was analysed, showing some important security failures: DoS attack, server impersonation, tag impersonation, DATA forgery, traceability. However, we improved Yoon's protocol resolving the security and privacy problems efficiently. The improved protocol resists against replay, impersonation, DATA forgery, DoS attacks and provides forward secrecy and untraceability. Our protocol was compared with the existing EPC-C1-GEN2-based RFID authentication protocol in terms of all security and privacy features.

References

[1] B. Song and C. J. Mitchell, "Scalable rfid security protocols supporting tag ownership transfer," *Comput. Commun.*, vol. 34, pp. 556–566, April 2011. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2010.02.027

[2] K. Ouafi and R. C.-W. Phan, "Privacy of Recent RFID Authentication Protocols," in *4th International Conference on Information Security Practice and Experience – ISPEC 2008*, ser. Lecture Notes in Computer Science, L. Chen, Y. Mu, and W. Susilo, Eds., vol. 4991. Sydney, Australia: Springer, April 2008, pp. 263–277.

[3] T. Dimitriou, "A lightweight rfid protocol to protect against traceability and cloning attacks," in *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, ser. SECURECOMM '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 59–66. [Online]. Available: http://dx.doi.org/10.1109/SECURECOMM.2005.4

[4] D. Henrici and P. Müller, "Hash-Based Enhancement of Location Privacy for Radio-Frequency Identification Devices Using Varying Identifiers," in *International Workshop on Pervasive Computing and Communication Security – PerSec 2004*, R. Sandhu and R. Thomas, Eds., IEEE. Orlando, Florida, USA: IEEE Computer Society, March 2004, pp. 149–153.

[5] H.-Y. Chien and C.-H. Chen, "Mutual authentication protocol for rfid conforming to epc class 1 generation 2 standards," *Comput. Stand. Interfaces*, vol. 29, pp. 254–259, February 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1222669.1222985

[6] T.-C. Yeh, Y.-J. Wang, T.-C. Kuo, and S.-S. Wang, "Securing rfid systems conforming to epc class 1 generation 2 standard," *Expert Syst. Appl.*, vol. 37, no. 12, pp. 7678–7683, Dec. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2010.04.074

[7] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "Cryptanalysis of a Novel Authentication Protocol Conforming to EPC-C1G2 Standard," in *Workshop on RFID Security – RFIDSec'07*, Malaga, Spain, July 2007.

[8] E.-J. Yoon, "Improvement of the securing rfid systems conforming to epc class 1 generation 2 standard," *Expert Syst. Appl.*, vol. 39, no. 12, pp. 1589–1594, Dec. 2012.

[9] S. Piramuthu, "Lightweight Cryptographic Authentication in Passive RFID-Tagged Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 38, no. 3, pp. 360–376, 2008.

[10] L. Kulseng, Z. Yu, Y. Wei, and Y. Guan, "Lightweight mutual authentication and ownership transfer for rfid systems," in *IEEE INFOCOM*, 2010, pp. 251–255.

[11] S.-Y. Kang, D.-G. Lee, and I.-Y. Lee, "A study on secure rfid mutual authentication scheme in pervasive computing environment," *Comput. Commun.*, vol. 31, no. 18, pp. 4248–4254, Dec. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2008.05.006

[12] R. Doss, W. Zhou, S. Sundaresan, S. Yu, and L. Gao, "A minimum disclosure approach to authentication and privacy in rfid systems," *Comput. Netw.*, vol. 56, no. 15, pp. 3401–3416, Oct. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2012.06.018

[13] M. Safkhani, N. Bagheri, S. K. Sanadhya, and M. Naderi, "Cryptanalysis of improved yeh *et al.*'s authentication protocol: An epc class-1 generation-2 standard compliant protocol," *IACR Cryptology ePrint Archive*, vol. 2011, p. 426, 2011.