For an EPC-C1 G2 RFID compliant Protocol, CRC with Concatenation : No; PRNG with Concatenation : Yes

Masoumeh Safkhani[3] and Nasour Bagheri[1]

[1] Electrical Engineering Department, Iran University of Science and Technology, Tehran, Iran,
M_Safkhani@iust.ac.ir
[2] Electrical Engineering Department, Shahid Rajaee Teacher Training University, Iran,
NBagheri@srttu.edu

Abstract. In this paper we present new constraints to EPCglobal Class 1 Generation 2 (EPC-C1 G2) standard which if they have been considered in the design of EPC-C1 G2 complaint authentication protocols, lead to prevent predecessor's protocols' weaknesses and also present the secure ones. Also in this paper as an example, we use Pang *et al.* EPC-C1 G2-friendly protocol which has been recently proposed, to show our proposed constraints in EPC-C1 G2 standard. Pang *et al.*'s protocol security analysis show how its security claim based on untraceability and resistance against de-synchronization attacks is ruined. More precisely, we present very efficient de-synchronization attack and traceability attack against the protocol. Finally, take Pang *et al.* protocol's vulnerability points, we present new conditions to design EPC-C1 G2 complaint protocols and based on it we propose a secure (EPC-C1 G2) RFID authentication scheme which is a good sample to EPC-C1 G2 complaint protocols.

keywords: RFID, Mutual Authentication, EPC-C1 G2, Cyclic Redundancy Code, Pseudo Random Number Generator, De-synchronization, Traceability Attack.

1 Introduction

One of the most important standards for RFID passive tags is EPC Class 1 Generation 2 (or in short term EPC-C1 G2) proposed by EPCglobal. This standard was adopted in 2004 and 18 months later (March-April 2006) ratified by ISO and published as an amendment to ISO-18000-6c [2] standard. EPC C1 G2 defines a platform for the interoperability of RFID protocols, by supporting an on-chip 16-bit Pseudo-random number generator (PRNG), a 16-bit cyclic redundancy code (CRC16) and lightweight operations such as XOR and $\|$. The most important properties of EPC-C1 G2 are summarized as below:

1. Tags are passive.
2. Tags operate on the UHF band (860-960 MHz).
3. Tags cannot support conventional cryptographic primitives.
4. Tags include on chip limited storage and computational resources for security purposes.

However, the original protocol is known to be insecure [5]. To improve the security of this standard, several protocols have been proposed which are compliant to this standard. Although in EPC-C1 G2 standard it has been approved the use of CRC function, $PRNG$ function and lightweight operations such as XOR and $\|$. However this standard does not

determine the conditions and constraints which are related to input operation of CRC and $PRNG$ functions until the resulting protocol is immune against all kind of active and passive attacks. In this paper, we consider such above conditions over one case which is Pang *et al.* EPC-C1 G2 - friendly protocol. Recently Pang *et al.* in [3], have analyzed two recent proposals in the context called RAP [6] and LADP [7] and discussed their vulnerabilities. In both of these protocols, on the reader's query, the tag sends $IDS_i = H(K_i)$ as a part of its response to the reader query. Although K_i is updated at the end of each query, however, as long as the tag has not updated its secret, this message can be used to trace the tag holder object. To overcome this problem, they have proposed an scheme based on a cyclic redundancy code (CRC) and a pseudo-random number generator in accordance with the EPC Class-1 Generation-2 specification. [3], which we call it $PLHAW$ that comes from the first letter of the protocol's authors name. The authors of $PLHAW$ claimed their protocol's resistance against common attacks. However, in this paper we scrutinize its security showing how an active adversary can efficiently de-synchronize the tag and the reader. So after this attack the reader and tag cannot authenticate each other in the further transactions. In addition, we show that $PLHAW$ protocol does not provide a better security against traceability attacks compared to its predecessors. Hence, $PLHAW$ proposal as an EPC-C1 G2 compliant protocol also does not provide the expected security. These results indicating use of CRC function with $\|$ as its input operation does not guarantee the security of protocol at all. Finally, by profiting $PLHAW$ vulnerability points, we present a new constraint for designing EPC-C1 G2 complaint protocols and according to it present a new secure EPC-C1 G2 authentication protocol.

1.1 Importance and Impact of paper

In this paper, we present new conditions on EPC-C1 G2 standard which leads to design secure protocols. To prove our claim, we will show important security faults on Pang *et al.* protocol (i.e.$PLHAW$), which is EPC-C1 G2 complaint protocol. In the other hand, $PLHAW$ protocol which is published in highly qualified journal [3] is the result of security analyses of two RFID protocols and hence security analysis of it has worth to work. Similar works such as what presented in this paper, introduce some conditions and constraints must be considered in the protocol design until they lead to present a secure matured protocol and hence advances research in this field. In the rest of this paper, we review $PLHAW$ protocol in Section 2 and present conditions which lead to our de-synchronization attack and traceability attack against it in Section 3 and Section 4 respectively. Section 5 presents an improved version of protocol,$PLHAW^+$ and its security analysis. Finally, we conclude the paper in Section 6.

2 Review of $PLHAW$ Protocol

$PLHAW$ protocol recently has been proposed by Pang *et al.* [3], to overcome the security pitfalls of its predecessors [6, 7]. To explain the protocol we use the notations indicated in Table 1. In this protocol each tag T_i has a secure identity SID_i and a secrete key K_i and the secret key is updated at the end of each successful session of the protocol. To overcome the

Notation	Description
IDS_i	The tag i index-pseudonym value
SID_i	The secure identity of tag i
$PRNG()$	16 bit pseudo random number generator
$H(.)$	Hash function
CRC	The cyclic redundancy check operation
K_i	The secret key shared between the tag i and the back-end server.
D_i	The detailed information of the tag i
X_{new}	The current value of X
X_{old}	The previous value of X
X_{left}	The left half of X
X_{right}	The right half of X
$\|\|$	The concatenation operation
$\oplus$	Exclusive or operation

Table 1. Notation

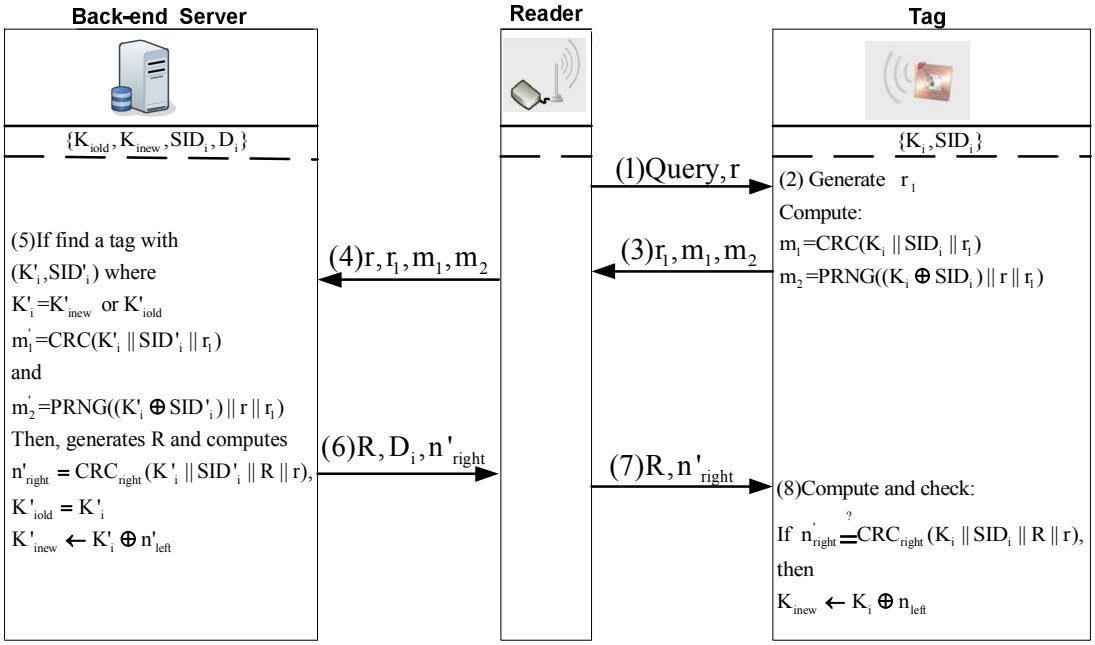

Fig. 1. *PLHAW* Mutual Authentication Protocol.

trivial de-synchronization attack caused by blocking the last message sent over the protocol, the back-end server BS keeps two records of K_i denoted by K_{inew} and K_{iold} respectively. *PLHAW*, as depicted in Fig. 1, runs as below:

1. The reader R_i generates a random number r and sends $Query$ command with r to the tag T_i.
2. Upon receiving the message, T_i does as follows:
   - generates a random numbers r_1,
   - computes $m_1 = CRC(K_i \| SID_i \| r_1)$ and $m_2 = PRNG((K_i \oplus SID_i) \| r \| r_1)$,
   - and sends $\{r_1, m_1, m_2\}$ to the reader.
3. The reader receives the message sent by the tag and sends the tuple $\{r, r_1, m_1, m_2\}$ to the back-end server BS.
4. Upon receiving the message, BS does as follows:
   - It searches for a tag T_i' with K_i' and SID_i' such that $m_1' = m_1$ and $m_2' = m_2$, where $m_1' = CRC(K_i' \| SID_i' \| r_1)$ and $m_2' = PRNG((K_i' \oplus SID_i') \| r \| r_1)$, here K_i' could be either of K_{inew}' or K_{iold}'.
   - If it finds such a tag, it generates a random number R, computes $n_{right}' = CRC_{right}(K_i' \| SID_i' \| R \| r)$, updates the secret key of the tag as $K_{iold}' = K_i'$ and $K_{inew}' = K_i' \oplus n_{left}'$ and sends the tuple $\{R, D_i, n_{right}'\}$ to the reader.
5. The reader sends $\{R, n_{right}'\}$ to the tag.
6. Upon receiving the message, T_i computes $n_{right} = CRC_{right}(K_i \| SID_i \| R \| r)$ and verifies whether $n_{right} = n_{right}'$ to authenticate the reader and update its secret key as $K_{inew} = K_i \oplus n_{left}$.

The designers of *PLHAW* claim that their protocol provides optimal security against all attacks in the context include de-synchronization attack and traceability attack. However, in this paper, we show that it suffers from efficient de-synchronization attack and traceability attack which rule out the designers' claim on the security of the protocol.

3 De-synchronization Attack

If for an authentication protocol's secret parameters, that are used through the authentication process, are updated then both parties should keep the same value; otherwise they won't authenticate each other in the later sessions and we say they have been desynchronized. In a desynchronization attack, the adversary forces the tag and the back-end server to update their common values to different values. If the adversary can succeed in forcing the tag and the back-end server to do so, they will not authenticate each other in further transactions. Pang *et al.* [3] claim that their protocol is secure against desynchronization attack. More precisely, the authors state that to prevent the desynchronization attack they keep a record of old secret value (It helps to remain in sync when the adversary blocks the last message sent from the back-end database to the tag). However, we present an efficient attack where the adversary forces the tag to update its secret value such that it does not match the values that back-end database keeps in its records. Our attack is based on the following linear property of CRC function [1, 4]:

$$CRC(A \| B) = CRC(A << n) \oplus CRC(B) \tag{1}$$

where A and B represent arbitrary values, n is the bit-length of string B and $<<$ denotes the left shift operation. An active adversary ($\mathcal{A}$) can exploit the above property to desynchronize the tag and the back-end server in *PLHAW* protocol as follows:

1. At the beginning of a session the reader R_i generates a random number r and sends $Query$ command with r to the target tag T_i.
2. Upon receiving the message, T_i does as follows:
   - generates a random numbers r_1,
   - computes $m_1 = CRC(K_i \| SID_i \| r_1)$ and $m_2 = PRNG((K_i \oplus SID_i) \| r \| r_1)$,
   - and sends $\{r_1, m_1, m_2\}$ to the reader.
3. The reader receives the message sent by the tag and sends the tuple $\{r, r_1, m_1, m_2\}$ to the back-end server BS.
4. Upon receiving the message, BS does as follows:
   - It searches for a tag T_i' with K_i' and SID_i' such that $m_1' = m_1$ and $m_2' = m_2$, where $m_1' = CRC(K_i' \| SID_i' \| r_1)$ and $m_2' = PRNG((K_i' \oplus SID_i') \| r \| r_1)$, where K_i' could be either of K_{inew}' or K_{iold}'.
   - T_i is a legitimate tag and its record matches the above criteria. Hence, the server generates a random number R, computes $n_{right}' = CRC_{right}(K_i \| SID_i \| R \| r)$, updates the secret key of the tag as $K_{iold}' = K_i'$ and $K_{inew}' = K_i' \oplus n_{left}'$ and sends the tuple $\{R, D_i, n_{right}'\}$ to the reader.
5. The reader sends $\{R, n_{right}'\}$ to the tag.
6. The adversary intercepts the message sent by the reader and replaces it by $\{R', n_{right}''\}$, where R' could be any random number not equal to R and $n_{right}'' = n_{right}' \oplus CRC_{right}(R << n) \oplus CRC_{right}(R' << n)$ and n is the bit length of r.
7. Upon receiving the message, T_i computes $n_{right} = CRC_{right}(K_i \| SID_i \| R' \| r)$ and verifies whether $n_{right} = n_{right}''$ to authenticate the reader and update its secret key as $K_{inew} = K_i \oplus n_{left}$.

If at the step 7 of the above attack the tag accepts the intercepted message $\{R', n_{right}''\}$ then it will update its secret key to $K_{inew} = K_i \oplus (CRC_{left}(K_i \| SID_i \| R' \| r))$ while the back-end server records for the tag's secret key are $K_{iold} = K_i$ and $K_{inew} = K_i \oplus (CRC_{left}(K_i \| SID_i \| R \| r))$ respectively. Since $R \neq R'$ with the probability of $1 - 2^{-n+1}$ the tag's record of the secret key does not match neither of the back-end server records for this tag's secret key. On the other hand, based on equation (1) of the given observation

$$
\begin{aligned}
& n_{right}'' && (2)\\
&= n_{right}' \oplus CRC_{right}(R << n) \oplus CRC_{right}(R' << n) && (3)\\
&= CRC_{right}(K_i \| SID_i \| R \| r) \oplus CRC_{right}(R << n) \oplus CRC_{right}(R' << n) && (4)\\
&= CRC_{right}((K_i \| SID_i) << 2n) \oplus CRC_{right}(R << n) \oplus \\
& \quad CRC_{right}(r) \oplus CRC_{right}(R << n) \oplus CRC_{right}(R' << n) && (5)\\
&= CRC_{right}((K_i \| SID_i) << 2n) \oplus CRC_{right}(r) \oplus CRC_{right}(R' << n) && (6)\\
&= CRC_{right}(K_i \| SID_i \| R' \| r) && (7)\\
&= n_{right}. && (8)
\end{aligned}
$$

Hence, the tag accepts the intercepted message, authenticates the back-end server and updates its secret key based on $R' \neq R$. Therefore, the adversary was successful in her attack. As easily seen from above, the adversary's success probability to de-synchronize the tag and

the back-end server is exactly the probability that $R \neq R'$ which is $1 - 2^{-n+1}$ and is almost '1'. Although above attack can been accomplished in one run of protocol, we can say that the attack's complexity is just one run of the protocol between the tag and a legitimate reader.

4 Traceability Attack

Traceability attack is a attack which in the adversary can recognize the identity of tags or reader which is faced with her. The main reasoning of Pang *et al.* on designing *PLHAW* was to overcome the trivial traceability of a tag in RAP [6] and LADP [7] as long as the tag has not updated its secret. To overcome this problem they have suggested to randomize messages sent from the tag on response to the reader's query such that when the reader sends r to the tag it generates another random number r_1 and computes $m_1 = CRC(K_i \| SID_i \| r_1)$ and $m_2 = PRNG((K_i \oplus SID_i) \| r \| r_1)$ and sends them to the reader. They claim that on each query to the tag, the tag generates a fresh r_1 and since it is included in the computation of m_1 and m_2 it is not possible for any adversary to link two different sessions of a tag even if the tag has not updated its secretes. However, based on equation (1) we have :

$$m_1 \oplus CRC(r_1) \tag{9}$$
$$= CRC((K_i \| SID_i) << n) \oplus CRC(r_1) \oplus CRC(r_1) \tag{10}$$
$$= CRC((K_i \| SID_i) << n) \tag{11}$$

Hence, assuming that the tag has not updated its secret key K_i, $m_1 \oplus CRC(r_1) = CRC((K_i \| SID_i) << n)$ which is always fixed and can be used to trace the tag's holder. Therefore, *PLHAW* also puts the privacy of tag's holder's on risk and it does not provide any security against traceability attack compared to its predecessors, i.e., RAP and LADP.

Remark 1. The adversary can desynchronize the tag and the reader following the given attack in section 3 and then use the above property to trace the tag's holder for ever.

5 Improved protocol

We now describe the improved version of *PLHAW* protocol, $PLHAW^+$, which is secure against the attacks mentioned in the previous sections and other attacks in the context.

In this section, we revise *PLHAW* with minor changes to make the resulting protocol immune against the attacks described in the previous sections. The main observation in attacks presented in the previous sections, is the use of CRC function with $\|$ as these functions' input inner operation in the protocol. To prevent the given desynchronization and traceability attacks it may be enough to change the CRC function used to compute m_1 and n_{right} to $PRNG$ function as follows. $PRNG$ function does not have linearity properties so it can be used as a one-way function:

$m_1 = PRNG(K_i \| SID_i \| r_1)$

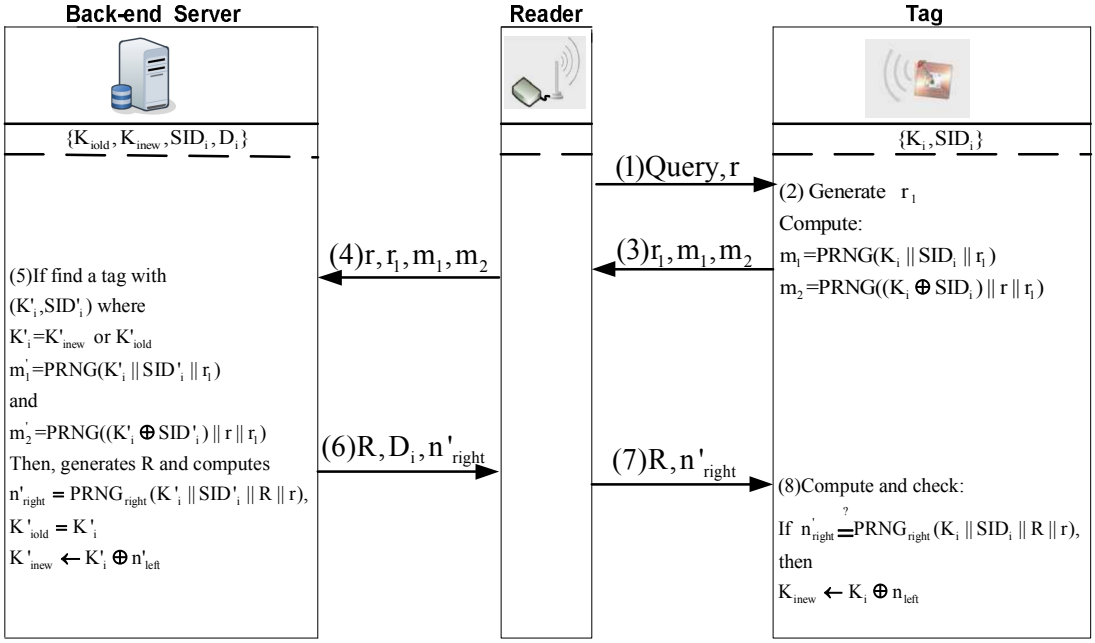

Fig. 2. $PLHAW^+$ Mutual Authentication Protocol.

$$n_{right} = PRNG_{right}(K_i \| SID_i \| R \| r)$$

Similar to $PLHAW$ protocol, $PLHAW^+$ supposes that each tag T_i has a secure identity SID_i and a secrete key K_i and the secret key is updated at the end of each successful session of the protocol. To overcome the trivial de-synchronization attack caused by blocking the last message sent over the protocol, the back-end server keeps two records of K_i denoted by K_{inew} and K_{iold} respectively.

The steps of $PLHAW^+$ mutual authentication protocol which are depicted in Fig. 2 are as follows:

1. The reader R_i generates a random number r and sends $Query$ command with r to the tag T_i.
2. Upon receiving the message, T_i does as follows:
   - generates a random numbers r_1,
   - computes $m_1 = PRNG(K_i \| SID_i \| r_1)$ and $m_2 = PRNG((K_i \oplus SID_i) \| r \| r_1)$,
   - and sends $\{r_1, m_1, m_2\}$ to the reader.
3. The reader receives the message sent by the tag and sends the tuple $\{r, r_1, m_1, m_2\}$ to the back-end server BS.
4. Upon receiving the message, BS does as follows:
   - It searches for a tag T_i' with K_i' and SID_i' such that $m_1' = m_1$ and $m_2' = m_2$, where $m_1' = PRNG(K_i' \| SID_i' \| r_1)$ and $m_2' = PRNG((K_i' \oplus SID_i') \| r \| r_1)$, here K_i' could be either of K_{inew}' or K_{iold}'.

- If it finds such a tag, it generates a random number R, computes $n'_{right} = PRNG_{right}(K'_i \| SID'_i \| R \| r)$, updates the secret key of the tag as $K'_{iold} = K'_i$ and $K'_{inew} = K'_i \oplus n'_{left}$ and sends the tuple $\{R, D_i, n'_{right}\}$ to the reader.

5. The reader sends $\{R, n'_{right}\}$ to the tag.
6. Upon receiving the message, T_i computes $n_{right} = PRNG_{right}(K_i \| SID_i \| R \| r)$ and verifies whether $n_{right} = n'_{right}$ to authenticate the reader and update its secret key as $K_{inew} = K_i \oplus n_{left}$.

5.1 *PLHAW*$^+$ Security Analysis

In this section, we present a detailed security analysis of $PLHAW^+$ to show that the protocol meets resistance against the attacks presented in this paper and the other known active and passive attacks in the context.

5.2 Resistance against Replay Attack

In $PLHAW^+$ protocol, the authentication messages (i.e. m_1, m_2, and n_{right} are computed with the random numbers, which provides freshness and so resistance against replay attack.

5.3 Backward\Forward Security

Key update relations in $PLHAW^+$ protocol is like that $PLHAW$ protocol, which all the secret values, K_i and SID_i, are necessary for updating K_i. In $PLHAW^+$, K_i is updated as $K_{i+1} \leftarrow K_i \oplus n_{left}$ with this difference which n_{left} is computed as $PRNG_{left}(K_i \| SID_i \| R \| r)$. So if the adversary can obtain the secret value K_i, she cannot calculate the new value K_{i+1} without knowing SID_i or the previous K_{i-1} Without knowing $n_{left} = PRNG_{left}(K_{i-1} \| SID_{i-1} \| R \| r)$ and SID_{i-1}.

5.4 Resistance against Tag Impersonation Attack

The resistance of $PLHAW^+$ protocol against tag impersonation attack arises from this fact that the tag's information (i.e. D_i) is stored in the back-end server, which is assumed to be secure, and this assumption that the communication channel between the back-end server and the reader is secure. So, an adversary is not able to access the information of a tag which is stored in the back-end server.

In the other hand, the adversary, who wants to impersonate the valid tag, must be to complete the authentication steps successfully, so she needs to respond to the reader with valid messages, m_1 and m_2 which are computed on the basis of the shared secret key K_i and SID_i. So it is not possible for attacker to compute valid m_1 and m_2 without knowing secret values of K_i and SID_i. Therefore $PLHAW^+$ protocol resists against tag impersonation attack.

5.5 Resistance Against De-synchronization Attack

De-synchronization attack against$PLHAW$ is based on the linear property of CRC function (1) which is fixed by using $PRNG$ function in using m_1 and n_{right} in $PLHAW^+$. So although the above linearity does not to apply to improved protocol, $PLHAW^+$ which is based on $PRNG$ function resists against de-synchronization attack.

5.6 Resistance Against Traceability Attack

Traceability attack against $PLHAW$ protocol is accomplished based on this fact that an adversary can link two different sessions of a tag based on equation (1) assuming that the tag has not updated its secret key K_i, $m_1 \oplus CRC(r_1) = CRC((K_i \| SID_i) << n)$ which is always fixed and can be used to trace the tag holder. However, in $PLHAW^+$ protocol, we fixed this weakness by using $PRNG$ function instead of CRC to compute m_1 and n_{right}. So the adversary cannot use the relations such that what used against $PLHAW$ to retrieve some fixed values related to identity of tags to trace given tag.

5.7 Performance Analysis of $PLHAW^+$ Protocol

In Table 2, the performance comparison of $PLHAW$ and $PLHAW^+$ protocols is provided. This table show that the proposed modifications do not increase the number of transferred bits, the number of $\oplus$ operation and the number of $\|$ operation. So we can say, $PLHAW^+$ protocol does not increase computational cost of the protocol extensively while it provides much better security. The only increased cost is some more calls of $PRNG$ function instead of use of CRC function.

	# $PRNG$	# CRC	# $\oplus$	# of $\|$	#transferred bits
BS of *PLHAW*	1	2	2n	7	3n
BS of $PLHAW^+$	3	0	2n	7	3n
Tag of *PLHAW*	1	2	2n	7	3n
Tag of $PLHAW^+$	3	0	2n	7	3n

Table 2. Performance comparison between $PLHAW$ and $PLHAW^+$ protocols. n denotes the bit length of parameters.

6 Conclusions

In this paper, we present significant points to designing secure RFID EPC-C1 G2 complaint protocols. More precisely, this paper shows use of CRC function with input of including $\|$ operation lead to insecure protocol. For an example, we show important security faults on

Pang *et al.* protocol (i.e. $PLHAW$), evidencing the necessity of a rigorous security analysis when a new protocol is proposed.

The main drawback of $PLHAW$ protocol is all due to the use of the CRC and take advantage of its linearity in computing m_1 and n_{right} values. So we used the linearity property of CRC function to mount our de-synchronization and traceability attacks against $PLHAW$. To fix this problem, one way is use of $PRNG$ function to compute m_1 and n_{right} values which use of $PRNG$ function is according to EPC-C1 G2 standard passive tags' specification. These modifications lead to burn a new EPC- C1 G2 complaint RFID mutual authentication protocol named $PLHAW^+$ which is secure against attacks mentioned in this paper and the other known active and passive attacks.

All in all, this paper shows doing similar works to what presented in this paper is essential to help to design matured secure protocols in the field of the RFID protocols design.

References

1. D. Han and D. Kwon. Vulnerability of an RFID authentication protocol conforming to EPC Class 1 Generation 2 Standards. *Computer Standards & Interfaces*, 31(4):648–652, 2009.
2. Information technology - Radio frequency identification for item management. Part 6: Parameters for air interface communications at 860 MHz to 960MHz. `http://www.iso.org/iso/catalogue_detail?csnumber=34117`. 2005.
3. L. Pang, H. Li, L. He, A. Alramadhan, and Y. Wang. Secure and efficient lightweight RFID authentication protocol based on fast tag indexing. *International Journal of Communication Systems*, doi= 10.1002/dac.2538, 2013.
4. P. Peris-Lopez, J. C. H. Castro, J. M. Estévez-Tapiador, and A. Ribagorda. Cryptanalysis of a novel authentication protocol conforming to EPC-C1 G2 standard. *Computer Standards & Interfaces*, 31(2):372–380, 2009.
5. P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda. RFID specification revisited. In *The internet of things:From RFID to The Next-Generation Pervasive Networked Systems*, pages 311–346. Taylor & Francis Group, 2008.
6. Z. Zhang, S. Zhou, and Z. Luo. Design and Analysis forRFID Authentication Protocol. In *e-Business Engineering, 2008. ICEBE '08. IEEE International Conference on*, pages 574–577, 2008.
7. S. Zhou, Z. Zhang, Z. Luo, and E. C. Wong. A lightweight anti-desynchronization RFID authentication protocol. *Information Systems Frontiers*, 12(5):521–528, 2010.