A Novel Mutual Authentication Scheme for RFID Conforming EPCglobal Class 1 Generation 2 Standards

Chin-Ling Chen*, 1, Yu-Cheng Huang2, Tzay-Farn Shih1

1 Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung, Taiwan, (R.O.C.), 41349
   e-mail: clc@mail.cyut.edu.tw; tfshih@mail.cyut.edu.tw

2 Department of Computer Science and Information Engineering, National Central University, No.300, Jhongda Rd., Jhongli City, Taoyuan County 32001, Taiwan (R.O.C.)
   e-mail : 985402024@cc.ncu.edu.tw

http://dx.doi.org/10.5755/j01.itc.41.3.860

Abstract. Radio-frequency identification (RFID) is an automatic identification technology. In recent years, more and more applications have been found for its use. However, there are still many security issues worth discussing. In this paper, we propose a mutual authentication scheme, which can solve problems such as privacy, replay attack, forward security, and user location privacy. In our scheme, we only use the tag for the purposes of being a storage media based on EPCglobal Class 1 Generation 2 (C1G2) standards. Analysis shows that the proposed scheme can resist known attacks and can be used in light-weight RFID systems of the current low-cost tags.

Keywords: RFID; EPC; security; mutual authentication; attack.

1. Introduction

Radio frequency identification (RFID) is a technology which is used to identify various objects. An RFID system consists of tags, readers, a host, and an antenna [9]. When a reader sends a request message to a tag, the tag responds with a message via radio frequency signals. In such an environment, there exist many latent attacks (such as Denial of Service attacks, man-in-the-middle attacks, replay attack and forge attacks etc.). Moreover, there are two notable security issues that should be considered: One is the privacy issue; another is the tracking issue [4].

(1) Privacy: If the Electronic Product Code (EPC) [8] in the tag is not encrypted, the attacker can obtain the message from the user’s RFID tag. Anyone may use a reader to obtain the EPC in the tag and query the database for the related information and the privacy of the tag owner would be violated.

(2) Tracking: For a tag, the same message is always given to a reader. If an attacker intercepts a message from the user’s RFID tag, the attacker can track the tag.

A good RFID system must avoid illegal accessing, protect a user’s privacy, and protect the RFID system.

The following security issues are often discussed for RFID systems:

(1) Resist forgery tag attack [16]

An attacker can continually listen to communication messages between the reader and the legitimate tag, for the attacker to isolate the legitimate tag, preventing it from operating. The attacker holds the communication information of the tags. The attacker should be able to calculate the next correct communication parameter from the intercepted message to forge a legitimate tag.

(2) Resist forgery server attack [16]

If an attacker intercepts the messages between a legitimate tag and a reader, then the attacker could supplant the server without knowing all its private information. The objective of this is to prevent the legitimate tag from updating its key. In the next tag reading, the server will receive next correct communication parameter. The attacker should be able to build message and “forgery server spoof” the tag to pass authentication.

(3) User location privacy [6, 13]
It means that the attacker knows the user’s location. The reason that causes this security problem is that the attacker gets a response message from the tag. Although the tag transfers the encrypted message to a legitimate reader, the attacker still can get user’s location by use of a multi-reader to query RFID tags at different locations.

(4) Resist replay attack [3]

When a reader wants to query a tag, the attacker will intercept the message between a legitimate reader and a legitimate tag, and then the attacker will be able to transmit intercepted message to spoof the tag or server to pass authentication.

(5) Forward secrecy [16]

An attacker listens to iteration between a legitimate reader and a legitimate tag and stores the messages. Then, the tag, which is not resistant to physical attacks, is compromised with the EPC being obtained by the attacker. At this point, the attacker will be able to obtain the secret keys and to generate future message.

(6) Man-in-the-middle attack [3, 6, 17]

Attacker can hold and modify the messages between tags and readers by intercepting the exchanged messages between a legitimate reader and a legitimate tag.

Recently, the RFID Class 1 Generation 2 (C1G2) standard was issued by EPCglobal. The main property is briefly summarized in the following:
- The RFID tag is passive, and its power is triggered by the readers.
- The RFID tag communicates at UHF band (800-960 MHz) and its communication range is from 2 m to 10 m.
- The RFID tag only supports on-chip 16-bit Pseudo-Random Number Generator (PRNG), and the 16-bit Cyclic Redundancy Code (CRC) checksum is used to detect errors in transmission data.
- The RFID’s privacy protection mechanism is to make the tag permanently unusable once it receives the kill command with a valid 32-bit kill PIN (e.g., tags can be killed at the point-of-sale).
- Read/write to the RFID tag’s memory is allowed only after it is in secure mode (i.e., after receiving the access command with a valid 32-bit access PIN).

To overcome the security threats, several protocols [1, 11, 12, 16] were proposed to enhance the RFID security. The RFID tags can only be considered as storage media. Thus, the computation ability is limited. There are about 500-5000 logic gates in current RFID tags. Thus, the traditional encryption and hash function mechanisms [10, 11, 15] are infeasible for EPCglobal C1G2 RFID tags.

In addition, plenty of literature reviews [14, 18] mentioned RFID tag related sources. Only a few proposed schemes [2, 3, 7, 12] can be implemented on EPCglobal C1G2 RFID tags. In 2009, Pedro et al. [16] proposed a cryptanalysis of a novel authentication protocol conforming to EPCglobal C1G2 RFID standard. Peris-Lopez et al. [16] showed various major security flaws in Chien et al.’s proposal. They show that none of the protocol objectives are met. It is vulnerable to attacks including unequivocal identification, tag impersonation, back-end database impersonation, back-end database auto-desynchronization, and tracking problem.

In this paper, we design a novel mutual authentication scheme for RFID EPCglobal class 1 generation 2 standards. The proposed scheme can resist known attacks. The rest of the paper is organized as follows: The preliminaries will introduce the related Cyclic Redundancy Codes in Section 2. The proposed protocol is shown in Section 3. Security analysis and discussions are presented in Section 4. Finally, some concluding remarks are drawn in Section 5.

2. Preliminaries

We will introduce the related EPCglobal C1G2 standards, Cyclic Redundancy Codes operation and properties in this section.

2.1. EPCglobal C1G2 standards

In the EPCglobal C1G2 standards, the computing resources of tags are limited. The tags only can operate CRC functions, simple logic operations, and generate random numbers; other complex operations (such as hash functions, symmetric encryption, and asymmetric encryption) cannot conform to the standards. According to the EPCglobal C1G2 standards, the RFID tag stores two keys in the tag: the kill key (Kill_key) and the access key (Access_key).

(1) Kill key (Kill_key): the key is used to verify the legitimacy of the transmission messages.

(2) Access key (Access_key): the key is used to write data to the EPCglobal C1G2 RFID tag’s memory.

2.2. Cyclic Redundancy Codes – CRCs

The Cyclic Redundancy Check (CRC) [5] is a checksum algorithm which is used to detect data errors during transmission. The CRC checksum is computed as a remainder of the division of the original data by the CRC polynomial.

2.2.1. The mathematics of CRCs

The hardware does not need strong computation power for a CRC operation. An n-bit CRC consists of an n-bit shift with some XOR gates. Computing the CRC is as follows:

\[ \text{CRC}(x) = x \mod (\text{polynomial}) \]

where \( x \) is the data to be checked, and \( \text{polynomial} \) is the CRC polynomial.
1. Load binary stream \( i(x) \) and polynomial \( d(x) \) with \( n \) degree.
2. Augment the binary stream by appending \( n \) zeros to the end of \( i(x) \).
3. Use the polynomial \( d(x) \) to divide \( i(x) \) to get the remainder \( r(x) \).

The stream should be multiplied by \( n \) (being \( n \) the degree of the CRC polynomial) prior to division to get \( r(x) \). That is, computing the CRC of a polynomial \( p(x) \) so that,

\[
0 \leq r(x) < p(x) \quad \text{and} \quad \text{deg}(r(x)) < \text{deg}(p(x)).
\]

(1)

The mechanism of computing an \( n \)-bit binary CRC is simple. Here is the first calculation for computing a 2-bit CRC:

\[
11010011 \quad \text{input} \\
\oplus 1011 \quad \text{divisor (4 bits)} \\
\hline
01100011 \quad \text{result}
\]

If the input stream of the leftmost divisor bit is 1, then the divisor is exclusive-ORed into the input stream. The divisor is then shifted one bit to the right, and the process is repeated until the divisor is equal to the right-hand end of the input stream. The following representation is the final result:

\[
00000001 \quad \text{result of penultimate calculation} \\
\oplus 00000000 \quad \text{divisor} \\
\hline
00000000 \quad \text{remainder (2 bits)}
\]

The detailed explanations of the CRC polynomial \( d(x) \) are described in Duc et al.’s scheme [7].

The EPCglobal C1G2 standards proposed the use of CRC-16–CCITT (CRC-16–CCITT = \( x^{16} + x^{12} + x^5 + 1 \)) which detects all single and double errors, and errors with an odd number of bits.

2.2.2. CRC properties

On the basis of the CRC linear property, Peris-Lopez et al. [16] proposed a cryptanalysis of a novel authentication protocol to show the Chien and Chen’s [3] faults. The Peris-Lopez et al.’s cryptanalysis basis is described as follows:

**Theorem.** For any CRC (independent of its divider polynomial) and for any values \( a, b, c \) and \( d \) \( \in F[2][x] \), it holds that:

\[
\text{CRC}(a||b) \oplus \text{CRC}(c||d) = \text{CRC}(a \oplus c||b \oplus d). \quad (2)
\]

**Proof.** From the definition in Eq. (1) above, one can write:

\[
\text{CRC}(a||b) = (a \cdot x^n \oplus b) \cdot x^n \oplus d_i(x) \cdot p(x) \quad \text{(3)}
\]

\[
\text{CRC}(c||d) = (c \cdot x^n \oplus d) \cdot x^n \oplus d_j(x) \cdot p(x) \quad \text{(4)}
\]

For polynomials \( d_1(x) \) and \( d_2(x) \) \( \in F[2][x] \), substituting these values in the left-hand of Eq. (2) we obtain the following:

\[
(a \cdot x^n \oplus b) \cdot x^n \oplus d_1(x) \cdot p(x) \oplus (c \cdot x^n \oplus d_2(x)) \cdot p(x). \quad (5)
\]

Rearranging terms in this expression we get:

\[
((a \oplus c) \cdot x^n \oplus (b \oplus d)) \cdot x^n \oplus (d_1(x) \oplus d_2(x)) \cdot p(x). \quad (6)
\]

That is, the corresponding expression is \( \text{CRC}(a \oplus c||b \oplus d) \) (analogously to Eqs. (3) and 4).

**Corollary 1.** In particular, if in Eq. (2) we have \( a = c \), then

\[
\text{CRC}(a||b) \oplus \text{CRC}(a||d) = \text{CRC}(a \oplus a||b \oplus d) = \text{CRC}(0||b \oplus d) = \text{CRC}(b \oplus d) \quad \text{(7)}
\]

because \( 0 \cdot x^n = 0 \cdot p(x) \).

**Corollary 2.** If \( c = b \) in Eq. (2), then

\[
\text{CRC}(a||b) \oplus \text{CRC}(b||d) = \text{CRC}(a \oplus b||b \oplus d) = \text{CRC}(a \oplus 0||d) = \text{CRC}(a||d) \quad \text{(8)}
\]

On the basis of the above property, we will propose a provable RFID mutual authentication scheme that conforms to the EPCglobal C1G2 standards and improves security.

### 3. Our scheme

We will propose a novel scheme for RFID systems to improve the security performance based on the EPCglobal C1G2 standard. The scheme involves three entities: tag (T), reader (R) and back-end server (S). We assume the communication channel is secure between the back-end server and the reader, but it is insecure between the tag and reader. It is susceptible to all possible attacks.

The proposed scheme is divided into two phases:
1. Initialization phase

#### 3.1. Notation

- \( M_{\text{reqM}} \): the request message
- \( N \): a nonce
- \( \oplus \): exclusive-or operation.
- \( \text{PID}_i \): the pseudonym identification code of the \( i \)-th tag
- \( \text{RID}_i \): the \( i \)-th reader’s identity
- \( S_{\text{SK}} \): the \( i \)-th session key shared by server and reader
- \( E_{\text{SK}}(m) \): use the session key \( S_{\text{SK}} \) to encrypt the message \( m \)
A Novel Mutual Authentication Scheme for RFID Conforming EPCglobal Class 1 Generation 2 Standards

\[ D_{SK}(m) \] use the session key \( SK_i \) to decrypt the message \( m \)

\[ EPC_i \]: 96-bit EPC (Electronic Product Code) of the \( i \)th tag

\( CRC(x) \): a Cyclic Redundancy Check (CRC) function

\( Kill\_key_i \): 32-bit kill key of the \( i \)th tag

\( Access\_key_i \): 32-bit access key of the \( i \)th tag

\( PRNG \): 32-bit pseudo-random number generator

\( A = B \): compare whether \( A \) is equal to \( B \) or not

\( \| \): the concatenation operation

\( DATA_i \): the product information of the \( i \)th tag

3.2. Initialization phase

Each tag and reader must register with the back-end server, respectively. The back-end server issues the corresponding Electronic Product Code (\( EPC_i \)), pseudonym identification (\( PID_i \)), an initial kill key (\( Kill\_key_i \)) and an initial access key (\( Access\_key_i \)) to a tag. The server also issues the reader's identification (\( RID \)) and the session key (\( SK \)) to a reader.

3.3. Mutual authentication phase

In this phase, the tags and the servers can perform the mutual authentication procedures. The tags and the servers can verify whether they are legal or not by working with each other. We illustrate the authentication scenario in Figure 1. The pseudonym and key updating procedures also must be executed for each transaction.

Step 1: When the reader wants to access a tag, it generates \( N_i \) and computes

\[ A = CRC(N_i) \]  \hspace{1cm} (9) \]

Then it sends the request message \( M_{req} \) and \( A \) to the tag.

![Figure 1. The scenario of mutual authentication phase](image-url)
Step 2: Upon receiving the request message, the tag generates a nonce \( N_2 \) and computes \( X, B \) and \( C_r \) as follows:
\[
X = N_2 \oplus \text{Kill} \_\text{key}_i
\]
(10)  
\[
B = \text{CRC}(A \parallel X)
\]
(11)  
\[
C_r = \text{CRC}(\text{EPC}_i \parallel B)
\]
(12)  
Then, it responds \((C_r, N_2, \text{PID}_i)\) to the reader.

Step 3: After receiving the tag’s response, the reader will involve \( A \) and its identity \( \text{RID}_i \), and \( (C_r, N_2, \text{PID}_i, A, \text{RID}_i) \) to forward the request to the server.

Step 4: When the server received the authentication request from the reader, it will verify whether \( \text{PID}_i \) and \( \text{RID}_i \), the tag’s pseudonym and reader’s identification existing in the database are equal to the received \( \text{PID}_i \) and \( \text{RID}_i \) respectively or not. If it is correct, the server will use \( A, N_2 \) and kill key \((\text{Kill} \_\text{key}_i)\) to calculate \( X' \) and \( B' \) as follows:
\[
X' = N_2 \oplus \text{Kill} \_\text{key}_i
\]
(13)  
\[
B' = \text{CRC}(A \parallel X')
\]
(14)  
Then the server will verify whether \( C_r \) is correct or not as follows:
\[
C_r \oplus \text{CRC}(\text{EPC}_i \parallel B') = 0
\]
(15)  
If the equality holds, the server computes \( Y, C_i \) and \( C_s \) as follows:
\[
Y = N_2 \oplus \text{Kill} \_\text{key}_i
\]
(16)  
\[
C_i = \text{E}_{sk_i}(\text{DATA}_i)
\]
(17)  
\[
C_s = \text{CRC}(\text{EPC}_i \parallel Y \parallel \text{Access} \_\text{key}_i) \oplus \text{Kill} \_\text{key}_i
\]
(18)  
Moreover, the server updates the pseudonym identification \((\text{PID}_i)\), kill key \((\text{Kill} \_\text{key}_i)\) and access key \((\text{Access} \_\text{key}_i)\) simultaneously as follows:
\[
\text{PID}_s = \text{PRNG}(\text{PID}_i)
\]
(19)  
\[
\text{Kill} \_\text{Key}_s = \text{PRNG}(\text{Kill} \_\text{Key}_i)
\]
(20)  
\[
\text{Access} \_\text{key}_s = \text{PRNG}(\text{Access} \_\text{key}_i).
\]
(21)  
The server transmits the message \((C_s, C_i)\) to the reader.

Step 5: After receiving the transmission messages \((C_s, C_i)\), the reader forwards \( C_s \) to the tag and obtains the product information \( \text{DATA}_i \) as follows:
\[
\text{DATA}_i = \text{D}_{sk_i}(C_i)
\]
(22)

Step 6: Upon receiving the message \( C_s \) of the reader, the tag uses the CRC function to verify the correctness as follows:
\[
C_s \oplus \text{CRC}(\text{EPC}_i \parallel Y \parallel \text{Access} \_\text{key}_i) \oplus \text{Kill} \_\text{key}_i = 0
\]
(23)  
If the equality holds, the tag also updates the pseudonym identification code \((\text{PID}_i)\), kill key \((\text{Kill} \_\text{key}_i)\) and access key \((\text{Access} \_\text{key}_i)\) simultaneously as follows:
\[
\text{PID}_s = \text{PRNG}(\text{PID}_i)
\]
(24)  
\[
\text{Kill} \_\text{Key}_s = \text{PRNG}(\text{Kill} \_\text{Key}_i)
\]
(25)  
\[
\text{Access} \_\text{key}_s = \text{PRNG}(\text{Access} \_\text{key}_i).
\]
(26)

4. Security analysis and discussions

In this section, we will examine and analyze whether the notable security requirements are satisfied or not.

4.1. Security analysis

4.1.1. Resist forge tag attack

In order to accomplish this attack, an adversary only needs to listen to iterative messages between the reader and the legitimate tag.

Each tag shares with the private messages of the reader: \( \text{EPC}_i \), the kill key \((\text{Kill} \_\text{key}_i)\) and the access key \((\text{Access} \_\text{key}_i)\), which is used to build messages \( A \) and \( C_r \). However, an attacker intercepts messages; she/he will be able to forge a legitimate tag. The following transmitted iteration messages can be intercepted by an attacker between the reader and the legitimate tag as follows:

1. \( \text{R} \rightarrow \text{T} : \text{M}_{eq} \), \( A \)
2. \( \text{T} \rightarrow \text{R} : \text{C}_r, N_2, \text{PID}_i \)

Once the attacker holds the information of \( \text{M}_{eq}, A, \text{C}_r, N_2, \) and \( \text{PID}_i \), the attacker can build message \( C_r' = \text{CRC}(\text{EPC}_i \parallel B') \) when queried by the reader.

Although the attacker does not know the private information stored in the tag \((\text{EPC}_i, \text{Kill} \_\text{key}_i\) and \(\text{Access} \_\text{key}_i\)) message \( C_r' \) can be easily computed. The different values of \( C_r \) and \( C_r' \) will be calculated by the XOR operation, and according to Corollary 1, the following expression can be derived:
\[
C_r \oplus C_r' = \text{CRC}(\text{EPC}_i \parallel B) \oplus \text{CRC}(\text{EPC}_i \parallel B') = \text{CRC}(B \oplus B').
\]
(27)  

The message \( C_r' \) is easily computed as follows:
\[
C_r' = C_r \oplus \text{CRC}(B \oplus B') = \text{CRC}(\text{EPC}_i \parallel B) \oplus \text{CRC}(B \parallel B'),
\]
(28)
In our scheme, the value \( B \), which is involved with \( C_f \), is not transmitted between the reader and the tag. Therefore, the attacker cannot calculate the next correct \( C_f \) value from the intercepted messages.

### 4.1.2. Resist forgery server attack

In this case, we will prove that our scheme can resist the forged server attacks as follows:

For the attacker, she/he can listen to iteration messages between a legitimate tag and a server. However, an attacker will be able to supplant a legitimate server.

- Step 1: \( R \rightarrow T \): \( M_{req}, A \)
- Step 2: \( T \rightarrow R \): \( C_f, N_2, PID_i \)
- Step 3: \( R \rightarrow S \): \( C_f, N_2, PID_i, A, RID_i \)
- Step 4: \( S \rightarrow R \): \( C_S, C_f \)
- Step 5: \( R \rightarrow T \): \( C_S \)

If an attacker intercepts the transmitted messages of \( C_f, N_2, PID_i, A, RID_i, H(PW) \), and \( C_S \) between the server and reader, the attacker can supplant the server without knowing all its private information (\( EPC_i, \text{Kill}_{key} \), \( \text{Access}_{key} \), \( \text{DATA} \)). In the next tag reading procedure, the server will receive \( C'_S, A' \) and \( N'_2 \). The fraudulent message \( C'_S \) is computed as follows. According to Corollary 1,

\[
\text{RCR}(a || b) \oplus \text{CRC}(a || d) = \text{CRC}(b || d). \tag{29}
\]

If each of \( b \) and \( d \) is the concatenation of some other variables \( (b=b_1 || b_2, d=d_1 || d_2) \), the above expression holds and can be rewritten as follows:

\[
\text{RCR}(a || b) \oplus \text{CRC}(a || d) = \text{CRC}(a || b_1) \oplus \text{CRC}(a || d_1) = \text{CRC}(b_1 || d_1) \oplus \text{CRC}(b_2 || d_2). \tag{30}
\]

According to Corollary 1, the following expression can be derived

\[
C'_S = (\text{CRC}(\text{EPC}_f || Y || \text{Access}_{key}) \oplus \text{Kill}_{key} || \text{EPC}_i, \text{Y}) \oplus \text{CRC}(\text{Y} \oplus Y') \oplus \text{CRC}(\text{Access}_{key} || \text{Access}_{key}'). \tag{31}
\]

So, the message \( C'_S \) is easily computed as follows:

\[
C'_S = (\text{CRC}(\text{EPC}_f || Y || \text{Access}_{key} || \text{Access}_{key}')) \oplus \text{Kill}_{key} \oplus \text{Kill}_{key}'. \tag{32}
\]

For the same reason, the server does not transmit \( Y \) and the access key (\( \text{Access}_{key} \)) between the server and the readers. So the attacker cannot calculate the next correct communication parameter \( C'_S \) from the intercepted message.

### 4.1.3. User location privacy

Although the attacker cannot obtain the plain text from the tag, the attacker still can trace the user’s location when tags respond to the reader’s queries with the same identifier.

The success of this attack depends on preventing tag key updating. If the attacker intercepts the messages between the reader and the legitimate tag, he or she is able to track the user’s location for the following reason:

1\(^{st} \) communication:

- Step 1: \( R \rightarrow T \): \( M_{req}, A \)
- Step 2: \( T \rightarrow R \): \( C_f = \text{CRC}(EPC_i || B), N_2, PID_i \)
- \( \vdots \)

\( n^{th} \) communication:

- Step 1: \( R \rightarrow T \): \( M_{req}, A \)
- Step 2: \( T \rightarrow R \): \( C_f = \text{CRC}(EPC_i || B'), N_2, PID_i \)

Now, the attacker intercepts the messages (\( C_f, C'_f \)) and computes the XOR of messages. According to Corollary 1

\[
C_f \oplus C'_f \equiv \text{CRC}(EPC_i || B) \oplus \text{CRC}(EPC_i || B') \tag{33}
\]

If \( C_f \) and \( C'_f \) came from the same tag, by the Eq. (34), the attacker can verify transmitted messages from the same tag as follows:

\[
A = C_f \oplus C'_f = \text{CRC}(EPC_i || B) \oplus \text{CRC}(EPC_i || B') \tag{34}
\]

Verify \( A = B \).

But the tag does not transmit \( B \) between the reader and the tag. Therefore, even the attacker intercepts the messages (\( C_f \) and \( N'_2 \)) from a legal reader, he/she cannot trace the user’s location.

### 4.1.4. Resist replay attack

Each tag shares some private information with the reader: \( EPC_i, \) the kill key (\( \text{Kill}_{key} \)) and the access key (\( \text{Access}_{key} \)). This information is used to build messages \( A \) and \( C_f \) in order to prove its authenticity.

If an attacker intercepts: \( EPC_i, \) the kill key (\( \text{Kill}_{key} \)) and the access key (\( \text{Access}_{key} \)) between the tag and the reader, the attacker can spoof the server by transmitting previously obtained \( C_f \) and \( A \) to pass the authentication. The scenario is described as follows:

The attacker intercepts the 1\(^{st} \) communication message:

- Step 1: \( R \rightarrow T \): \( M_{req}, A \)
- Step 2: \( T \rightarrow R \): \( C_f = \text{CRC}(EPC_i || B), N_2, PID_i \)

The legitimate \( n^{th} \) communication:
Step 1: R → T : Mreq, A
Step 2: T → R : C, N1

The attacker replays the previously obtained C to pass authentication, but she/he will fail. From Corollary 1, the reason is described as follows:
\[ C_2' = CRC(EP), || B') \]
\[ C_2'' = CRC(EP), || B') \]
\[ C_2' ≠ C_2'' \]

Since B and N2 were updated for each transaction and the value B is not transmitted in plaintext. Thus, the attacker cannot spoof the server by transmitting the previously obtained C and A to pass the authentication.

4.1.5. Forward secrecy

In this subsection, we will show that an attacker cannot compromise a tag and obtain its resident data; the attacker cannot obtain any secret information of the tags.

Suppose that an attacker listens to iteration messages (A', N1', C', C2') between a legitimate reader and a legitimate tag, and stores these values. Then, the tag will suffer from the forward secrecy. Due to the EPC, being obtained by the attacker, she/he will be able to obtain the secret keys (Kill_key' and Access_key'), and to generate the correct communication C', the detail scenario of this attack is described as follows:

Step 1: R → T : Mreq, A
Step 2: T → R : C', N1', PID
Step 3: R → T : C',

From Corollary 1, the attacker obtains the transmitted messages between the server and reader, she/he will calculate C' as follows:
\[ C_s' = CRC(EP), || Y') || Access_key', || Access_key') \]
\[ C_s'' = CRC(EP), || Y') || Access_key', || Access_key') \]
\[ C_s' ≠ C_s'' \]

So, the message C is easily computed as follows:
\[ C_s = CRC(EP), || Y') || Access_key', || Access_key') \]
\[ C_s' = CRC(EP), || Y') || Access_key', || Access_key') \]
\[ C_s'' = CRC(EP), || Y') || Access_key', || Access_key') \]
\[ C_s' ≠ C_s'' \]

But in our scheme, the kill key (Kill_key') and the access key (Access_key') are updated for each transaction, and the parameters Y', Access_key', and Kill_key' are not transmitted in plaintext. Thus, if an attacker intercepts the messages between the server and the tag, the attacker cannot access the tags' secret data.

4.1.6. Resist man-in-the-middle attack

The proposed scheme can resist the man-in-the-middle attack. The reason is described as follows:

An attacker intercepts the communication messages between the tag and reader. For example, an attacker mimics a legal role when the reader wants to query a tag. The attacker will intercept the message from the reader, and then transfer the intercepted message to the tag as follows:

Step 1: R → T : Mreq, A
Step 2: T → R : C, N2, PID
Step 3: R → S : C, N2, PID, A, RID
Step 4: S → R : C, C
Step 5: R → T : C

The tag and the server can calculate C values by using EPC, Y and keys (Kill_key, and Access_key) as follows:
\[ C_s = CRC(EP), || Y', || Access_key) ⊕ Kill_key. \]

If an attacker can hold and modify the messages, the message C' is easily computed from Corollary 1 as follows:
\[ C_s' = CRC(EP), || Y', || Access_key') ⊕ Kill_key' \]

In our scheme, the correct kill key (Kill_key') and access key (Access_key) are protected by related parameters and updated for each transaction. Thus, attackers attempt to use a forged kill key (Kill_key') and access key (Access_key') to pass the tag’s authentication will fail.

The reason is described as follows:
\[ C_s = CRC(EP), || Y', || Access_key) ⊕ Kill_key \]
\[ C_s' = CRC(EP), || Y', || Access_key') ⊕ Kill_key' \]
\[ C_s = C_s' \]

The attacker cannot calculate the next correct communication parameter C' from the intercepted message to spoof the tag.

4.2. Discussions

We compare the time complexity of the proposed scheme with those of the previous schemes during the mutual authentication phase in Table 1.
Table 1. Comparison of the time complexity

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Time complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Karthikeyan–Nesterenko [12]</td>
</tr>
<tr>
<td></td>
<td>Duc et al. [7]</td>
</tr>
<tr>
<td></td>
<td>Chien and Chen [3]</td>
</tr>
<tr>
<td></td>
<td>Our scheme</td>
</tr>
<tr>
<td>Tag</td>
<td>$1T_{COMP} + NT_{XOR}$</td>
</tr>
<tr>
<td></td>
<td>$1T_{COMP} + 2T_{XOR} + 2T_{PRNG} + 3T_{CRC}$</td>
</tr>
<tr>
<td></td>
<td>$1T_{COMP} + 3T_{XOR} + 2T_{PRNG} + 3T_{CRC}$</td>
</tr>
<tr>
<td></td>
<td>$1T_{COMP} + 2T_{XOR} + 3T_{PRNG} + 3T_{CRC}$</td>
</tr>
<tr>
<td>Reader</td>
<td>Need not</td>
</tr>
<tr>
<td></td>
<td>$2T_{COMP} + NT_{XOR}$</td>
</tr>
<tr>
<td></td>
<td>$1T_{COMP} + NT_{XOR} + 1T_{PRNG} + 3T_{CRC}$</td>
</tr>
<tr>
<td></td>
<td>$1T_{COMP} + 2T_{XOR} + 3T_{PRNG} + 3T_{CRC}$</td>
</tr>
<tr>
<td>Server</td>
<td>$1T_{COMP} + NT_{XOR}$</td>
</tr>
<tr>
<td></td>
<td>$2T_{COMP} + NT_{XOR} + 1T_{PRNG} + 3T_{CRC}$</td>
</tr>
<tr>
<td></td>
<td>$1T_{COMP} + 3T_{XOR} + 2T_{PRNG} + 3T_{CRC}$</td>
</tr>
<tr>
<td></td>
<td>$1T_{COMP} + 3T_{XOR} + 3T_{PRNG} + 3T_{CRC} + T_{SYE}$</td>
</tr>
</tbody>
</table>

Notes:
- $N$: the number of the tags
- $T_{COMP}$: the time for comparison operation
- $T_{XOR}$: the time for executing an exclusive-or operation
- $T_{PRNG}$: the time for executing a pseudo-random number generation operation (16 bits)
- $T_{CRC}$: the time for executing a Cyclic Redundancy Check (CRC) function (16 bits)
- $T_{SYE}$: the time for executing a symmetric encryption operation (1024 bits)
- $T_{SYD}$: the time for executing a symmetric decryption operation (1024 bits)

Because of the EPCglobal C1G2 standards only support simple operations, for example: exclusive-OR, random number generation and CRC operations for tag operation, and some previous schemes often used the symmetric or asymmetric cryptosystem to implement their applications. Therefore, these schemes do not conform to the EPCglobal C1G2 standards. These operations are not suitable to current low cost tag.

Simultaneously, the proposed scheme can resist various attacks and with mutual authentication. None of the previous methods can achieve the listed requirements, but our scheme achieves all requirements. We compare the security and property of our scheme with those of the previous schemes in table 2.

Table 2. Security comparison

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Resist forgery tag attack</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Resist forgery server attack</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>User location privacy</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Resist replay attack</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Forward secrecy</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Resist man-in-the-middle attack</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

5. Conclusions

In this paper, we proposed a mutual authentication protocol based on EPCglobal C1G2 standard to resist various attacks and it can enhance the security. Although several schemes have been proposed for RFID systems, only few schemes conform to the EPC C1G2 standard. Our scheme only uses simple operator (XOR and PRNG) on the tag, hence it is suitable for low-cost RFID.

To sum up, our scheme has achieved the following:

1. Resist the forgery tag attack
2. Resist the forgery server attack
3. User location privacy
4. Resist the replay attack
5. Forward secrecy
6. Resist the man-in-the-middle attack

In summary, our scheme can be used in lightweight RFID systems that conform to EPCglobal Class 1 Generation 2 standards. The RFID system has attracted much attention and been applied to many applications, such as ownership transfer, manufacturing and inventory control can achieve a higher security in current low-cost tags.

Acknowledgment

This work is partially supported by the National Science Council, Taiwan, under contract no. 101-2622-E-324-003-CC3.
References


Received December 2010.