Using RFID Yoking Proof to Design A Supply-Chain Applications for Customs Check

Chin-Ling Chen*
Department of Computer Science and Information Engineering
Chaoyang University of Technology, Taiwan
clc@mail.cyut.edu.tw

Chun-Yi Wu
Department of Computer Science and Engineering
National Chung-Hsing University, Taiwan
tim22774413@yahoo.com.tw

Fang-Yie Leu and Yi-Li Huang
Department of Computer Science
Tunghai University, Taiwan
{leufy, yifung}@thu.edu.tw

Abstract

There are currently numerous practical applications to improve the effectiveness of Radio Frequency Identification (RFID) Systems. RFID provides an efficient identification for shipping and receiving goods. In warehouse management, RFID mechanisms can be used to check whether large quantities of goods are presented within a supply-chain system. In a supermarket, product information can be acquired without asking the clerks, one by one. The above applications are limited with regard to authentication between a single tag and server, while other applications focus on authenticating the relationships among the tags. In this paper, we use a yoking proof mechanism conforming to EPC-global Class 1 Generation 2 Standards to improve the customs container check. Our scheme can also defend against known attacks and greatly enhance security.

Keywords: RFID, EPCglobal, Security, Yoking proof, Supply-chain, Authentication, Attacks.

1 Introduction

In recent years, division of work has been a popular model in the product line. It is a trend of globalization around the world. For example, a product may contain various kinds of materials from different countries. That is, a commercialization product may come from various countries or companies. In order to reduce the costs and offer higher quality, increased transportation cannot be avoided among the countries around the world. Accompanying the increased transportation, custom checks may form a bottleneck in the supply chain. Thus, in this paper we are interested in designing a method to solve this problem by using an RFID mechanism and offering an effective service. Since RFID technology exhibits powerful identification characteristics, it can help reduce manpower costs and increase operational efficiency. This is especially useful in a supply chain environment [1][17][19]. As RFID can identify an object without requiring physical contact, it provides an efficient identification for shipping and receiving containers. However, an important issue for RFID systems is demonstrating where two or more RFID tags are simultaneously located. The typical RFID applications can be classified using three authentication styles: (1) object-to-object, (2) object-to-man, and (3) man-to-man. An example of (1) would be a scenario where a pharmaceutical distributor may want to prove that a bottle of medicine was
sold together with its instructions leaflet or that a number of components were delivered simultaneously. Regarding (2), a user could lend a book by associating it with another book using their RFID card in a library. Alternately, customs could quickly check boarding card information, i.e. baggage, along with one’s electronic passport. Furthermore, in a battlefield context, weaponry or equipment may be linked to a specific person. With (3), meeting organizers may want to prove that a group of people were present together at a meeting. The common thread is that these applications have to prove that the related tags simultaneously existed, and our RFID application focuses on the object-to-man style. Thus, how to prove where two or more members’ RFID tags are simultaneously has become an important issue in RFID applications.

For example, the Boeing Company is famous for building airplanes all over the world, but the Boeing Company only produces a small amount of the airplane materials, while the most material is manufactured by other companies. When Boeing wants to build an airplane from materials from different companies or countries, they must correctly identify each part for its corresponding plane. If the Company does not create an exact match, it may cause an accident because of unsafe architecture. Similarly, customs has to differentiate between the containers from the various countries, so customs is also confronted with an overload of container checks. To prevent the checks from becoming a bottleneck for the supply chain, an RFID yoking proof mechanism can solve the similar applications.

In 2004, Juels [16] first defined a yoking proof which used a random number independently generated by each tag to produce a proof. In 2005, Saito and Sakurai [22] observed that Juels’ scheme is vulnerable to replay attacks and proposed a grouping proof that requested timestamps from a trusted server. In 2006, Piramuthu [21] showed that Saito and Sakurai’s scheme was still vulnerable to replay attacks and proposed existence-proofs that kept a random number in the tag memory and sent the information generated by a pre-tag as the input of the next tag. Next, Lin et al. [18] argued that Piramuthu’s scheme was still vulnerable to race conditions between tags and readers, so they proposed a coexistence proof that used a couple chain to solve the problem. In 2008, Burmester et al. [6] proposed provably secure grouping-proofs, which offer a new solution to the communication problems between tags by group, such that the tags can use a common secret within the group to verify other tags. However, the problem for Burmester et al’s scheme is that it does not explain how to find the anonymity tag. In 2009, Chien and Liu [9] proposed a tree-based RFID yoking proof that can identify the tags as \( O(1) \) and retained anonymity. In 2010, Pedro et al. [20] discussed flaws in the RFID grouping proofs that conform to the EPCglobal C1G2 standards [15], and are low cost. Notably, this scheme is not able to achieve mutual authentication between the tag and reader. Moreover, problems exist in the above schemes, such as failure to resist known RFID attacks and race conditions, lack of forward secrecy, and non-conformer to EPCglobal C1G2 standards and off-line verification. In 2010, Huang and Ku [14] proposed an RFID grouping proof protocol for the safe medication of inpatients to discuss the RFID application with the medical management. Next, Chien et al. [10] proposed two RFID-based solutions to enhance inpatient medication safety to improve Huang and Ku’s protocol. Suitably, a good yoking proof scheme should conform to EPCglobal C1G2 standards [7] or resist related attacks, such as replay attacks, man-in-the-middle attacks, impersonation tag attacks, tag tracking attacks and DoS attacks. In this paper, we use the yoking proof mechanism of RFID to enhance the efficiency of the customers’ check. The proposed novel protocol can also achieve these requirements.

The remainder of this paper is organized as follows: Section 2 introduces the proposed protocol. We discuss how the proposed scheme can defend against known attacks and introduce the protocol’s mechanism in Section 3. Section 4 provides a comparison of security and cost with other schemes, and we provide conclusions in Section 5.
2 The proposed scheme

2.1 Notation

\[ M : \text{a secret value known by tag, reader and server} \]
\[ EPC_x : \text{the EPC code of tag } X \]
\[ Key_x : \text{tag } X \text{’s secret key} \]
\[ SSK : \text{the server’s private key, based on the RSA assumption} \]
\[ SPK : \text{the server’s public key, based on the RSA assumption} \]
\[ RSK : \text{the reader’s private key, based on the RSA assumption} \]
\[ RPK : \text{the reader’s public key, based on the RSA assumption} \]
\[ R_S, R_R, R_T : \text{the random numbers generated by server, reader, tag (driver’s tag, car’s tag or container’s tag) respectively} \]
\[ PRNG() : \text{a pseudorandom number generation function} \]
\[ S_{SSK}(m) : \text{use the server’s private key, } SSK, \text{ to sign message } m \]
\[ V_{SPK}(m) : \text{use the server’s public key, } SPK, \text{ to verify message } m \]
\[ D_{RSK}(m) : \text{use the reader’s private key, } RSK, \text{ to decrypt message } m \]
\[ E_{RPK}(m) : \text{use the reader’s public key, } RPK, \text{ to encrypt message } m \]
\[ \oplus : \text{exclusive-or operation} \]
\[ A = B : \text{determine whether } A \text{ is equal to } B \text{ or not} \]
\[ \rightarrow : \text{insecure channel} \]

2.2 System framework

The proposed protocol proves the simultaneous presence of members (driver, car and container) and then checks to see if the relationship between the members is legal or not without the server. At first, assume customs can activate their readers to communicate with tags, and there exists a secure channel between the reader and the server. When a driver ships containers into a warehouse or transports containers to another place, customs verifies the identity of the members by decrypting the message on the driver’s tag and then checks to see if the relationship between the members is legal or not. Finally, customs, according to the information, checks to see if the driver is legal or not. In the proposed scheme, we use the RSA [24] assumption to improve security between reader and server. Similar applications appeared in [2] [8] [25]. We also provide a related proof to illustrate the security in Subsection 3.1.1. Here, we present the scenario as in Figure 1.

(1) Server: manages all tags and stores related information. For example: the reader’s public key, tag’s EPC code, and secret key.

(2) Reader: an RFID reader which can authenticate the tags.
Using RFID Yoking Proof to Design A Supply-Chain Applications for Customs Check

Chin-Ling Chen, Chun-Yi Wu, Fang-Yie Leu and Yi-Li Huang

Figure 1: Scenario of yoking-proof protocol

(3) Tag: a tag (\(T_{\text{driver}}\), \(T_{\text{car}}\) or \(T_{\text{container}}\)) stores the related information of a member (driver, car or container) and conforms to EPCglobal C1G2 standards.

The scenario of our scheme is divided into ten steps:

Step 1: The server sets the secret information into each tag (driver, car and container), and then generates a symmetric key with the reader for processing communication.

Step 2: The reader sends a query message to the driver’s tag.

37
Step 3: The tag randomly chooses a random number to mix the encrypted message and then sends the mixed message to the reader.

Step 4: The reader decrypts the mixed message to get the member’s information and sends a challenge message to the car’s tag.

Step 5: Upon receiving the challenge message from the reader, the car’s tag verifies the message and responds to the reader.

Step 6: The reader verifies the response message and sends a challenge message to the container’s tag.

Step 7: Upon receiving the challenge message, the container’s tag verifies the message and responds to the reader.

Step 8: The reader verifies the response message from the container’s tag and responds with a message with the driver’s challenge to the driver’s tag.

Step 9: Upon receiving the response from the reader, the driver’s tag verifies the received message to check to see if the reader is legal or not. Then, the driver’s tag responds to the reader.

Step 10: The reader verifies whether the driver is correct or not.

Our protocol includes the following two phases: initial phase and authentication phase.

2.3 Initial Phase

In this phase, the reader and tags register with the server, and the server sets the secret information into the tags and records the information. We present the scenario of the registration phase as follows:

Step 1: Reader and tags register with the server via a secure channel.

Step 2: Upon receiving the requested information from the driver’s tag, the server generates a random number, $R_S$, with the server’s private key, $SSK$, and the reader’s public key, $RPK$, to compute $(C_1, C_2)$ respectively:

$$C_1 = S_{SSK}(EPC_{driver} \oplus R_S, EPC_{car} \oplus R_S, EPC_{container} \oplus R_S)$$

$$C_2 = E_{RPK}(EPC_{driver}, EPC_{car}, EPC_{container}, Key_{driver}, Key_{car}, Key_{container}, R_S)$$

Then, the server sets the secret value, $M$, the EPC codes $(EPC_{driver}, EPC_{car}, EPC_{container})$, and the secret keys $Key_{driver}, Key_{car}, Key_{container}$, to their corresponding tag $T_{driver}, T_{car}, T_{container}$, respectively. Finally, the server sends the information $(R_S, C_1, C_2)$ to the driver’s tag via the secure channel.

Step 3: Upon receiving message $(R_S, C_1, C_2)$, the driver’s tag stores $(R_S, C_1, C_2)$. 38
Using RFID Yoking Proof to Design A Supply-Chain Applications for Customs Check
Chin-Ling Chen, Chun-Yi Wu, Fang-Yie Leu and Yi-Li Huang

2.4 Authentication phase

In this phase, the reader sends a query message to the driver’s tags and examines the simultaneous presence of member tags without the server’s help. In addition, the driver’s tags can authenticate the legal reader through a challenge-response. Notably, our protocol can support the member’s inclusion of
more than two members. Here we use the driver’s tag, $T_{\text{driver}}$; car’s tag, $T_{\text{car}}$; container’s tag, $T_{\text{container}}$ and reader to present the scenario of the authentication phase in Fig. 2 and 3.

Step 1: Before authenticating the tags, the reader has to query the driver’s tag, $T_{\text{driver}}$, to get the relevant information about the members. So, the reader sends a query message to the driver’s tag for service.

Step 2: Upon receiving the query message, the driver’s tag, $T_{\text{driver}}$, generates a random number $R_{T_1}$ and computes:

$$C_3 = C_1 \oplus R_{T_1}$$ (3)
Using RFID Yoking Proof to Design A Supply-Chain Applications for Customs Check

Chin-Ling Chen, Chun-Yi Wu, Fang-Yie Leu and Yi-Li Huang

\[ C_4 = C_2 \oplus R_{T_1} \] (4)

\[ C_5 = R_{T_1} \oplus M \] (5)

Then the driver’s tag, \( T_{driver} \), sends message \((C_3, C_4, C_5)\) to the reader.

Step 3: After receiving message \((C_3, C_4, C_5)\) from the driver’s tag, \( T_{driver} \), the reader first computes:

\[ R'_{T_1} = C_5 \oplus M \] (6)

then, uses the computed challenge number, \( R'_{T_1} \) and the private key, \( RSK \), to decrypt \( C_4 \) to obtain the member’s identity information and secret information.

\[ (EPC'_{driver}, EPC'_{car}, EPC'_{container}, Key'_{driver}, Key'_{car}, Key'_{container}, R'_S) = D_{RSK}(C_4 \oplus R'_{T_1}) \] (7)

Thus, the reader gets the random number, \( R'_S \), generated by the server, each member’s EPC \((EPC'_{driver}, EPC'_{car}, EPC'_{container})\) and secret keys \((Key'_{driver}, Key'_{car}, Key'_{container})\).

Next, the reader uses the random number, \( R'_S \) from the server and the member’s identity information \((EPC'_{driver}, EPC'_{car}, EPC'_{container})\) to authenticate \( C_5 \) as signed by a legal server or not:

\[ (EPC'_{driver} \oplus R'_S, EPC'_{car} \oplus R'_S, EPC'_{container} \oplus R'_S) ? = V_{SPK}(C_3 \oplus R'_{T_1}) \] (8)

If Eq. (8) holds, the message \((R_S, C_1, C_2)\) is correctly generated by the server, then the reader generates a random number, \( R_{R_1} \), and computes:

\[ C_6 = EPC'_{car} \oplus R_{R_1} \oplus M \] (9)

\[ C_7 = PRNG(EPC'_{car} \oplus Key'_{car} \oplus R_{R_1}) \] (10)

Finally, the reader sends message \((C_6, C_7)\) to the car’s tag, \( T_{car} \).

Step 4: Upon receiving the query message, the car’s tag, \( T_{car} \), uses its EPC \( EPC_{car} \) and secret value, \( M \), to compute the challenge number

\[ R_{R_1} = C_6 \oplus EPC_{car} \oplus M \] (11)

and then verifies \( C_7 \)

\[ C_7? = PRNG(EPC_{car} \oplus Key_{car} \oplus R'_{R_1}) \] (12)

If Eq. (12) holds, the reader is correct and the car’s tag, \( T_{car} \), generates a random number, \( R_{T_2} \). Then the car’s tag, \( T_{car} \), uses the challenge number, \( R'_{R_1} \), a random number, \( R_{T_2} \) and the secret key, \( Key_{car} \), to compute the message, \( C_8 \) and \( C_9 \):

\[ C_8 = EPC_{car} \oplus T_{T_2} \oplus M \] (13)
Finally, the car’s tag, $T_{car}$, sends message $(C_8, C_9)$ to the reader.

**Step 5:** Upon receiving message $(C_8, C_9)$ from the car’s tag, $T_{car}$, the reader computes the challenge number to verify as follows:

$$R'_{T_2} = C_8 \oplus \text{EPC}_{car}^t \oplus M$$

$$C_9? = \text{PRNG}(\text{EPC}_{car}^t \oplus \text{Key}_{car}^t \oplus R_{R_1}^t \oplus R'_{T_2})$$

If Eq. (16) holds, the car’s tag, $T_{car}$, is correct. Then the reader generates a random number, $R_{R_2}$, to compute message $(C_{10}, C_{11})$ as follows:

$$C_{10} = \text{EPC}_{container}^t \oplus R_{R_2} \oplus M$$

$$C_{11} = \text{PRNG}(\text{EPC}_{container}^t \oplus \text{Key}_{container}^t \oplus R_{R_2}^t)$$

Next, the reader sends message $(C_{10}, C_{11})$ to challenge the container’s tag, $T_{container}$.

**Step 6:** Upon receiving message $(C_{10}, C_{11})$, the container’s tag, $T_{container}$, decomposes $C_{10}$ to get the challenge number, $R'_{R_2}$, of the reader with its EPC and the secret value, $M$, as follows:

$$R_{R_2}^t = C_{10} \oplus \text{EPC}_{container}^t \oplus M$$

Next, the container’s tag, $T_{container}$, verifies $C_{11}$

$$C_{11}? = \text{PRNG}(\text{EPC}_{container}^t \oplus \text{Key}_{container}^t \oplus R_{R_2}^t)$$

If Eq. (20) holds, the reader is correct and the container’s tag, $T_{container}$, generates a random number, $R_{T_3}$, to respond to the reader. Then the container’s tag, $T_{container}$, uses the challenge number, $R_{R_2}^t$; a random number, $R_{T_3}$ and its secret key, $\text{Key}_{container}$, to compute message $(C_{12}, C_{13})$:

$$C_{12} = \text{EPC}_{container}^t \oplus R_{T_3} \oplus M$$

$$C_{13} = \text{PRNG}(\text{EPC}_{container}^t \oplus \text{Key}_{container}^t \oplus R_{R_2}^t \oplus R_{T_3})$$

Finally, the container’s tag, $T_{container}$, sends message $(C_{12}, C_{13})$ to the reader.

**Step 7:** Upon receiving message $(C_{12}, C_{13})$ from the container’s tag, $T_{container}$, the first the reader must get the challenge number, $R'_{T_3}$, of the container’s tag, $T_{container}$, as follows:

$$R'_{T_3} = C_{12} \oplus \text{EPC}_{container}^t \oplus M$$
Using RFID Yoking Proof to Design A Supply-Chain Applications for Customs Check

Chin-Ling Chen, Chun-Yi Wu, Fang-Yie Leu and Yi-Li Huang

Second, the tag reader verifies

\[ C_{13} = PRNG(EPC_{\text{container}}' \oplus Key'_{\text{container}} \oplus R_{R_5} \oplus R'_{T_1}) \]  \hspace{1cm} (24)

If Eq.(24) holds, the container’s tag, \( T_{\text{container}} \), is correct. Now the reader has already checked both the car’s tag and the container’s tag. So the reader generates a random number, \( R_{R_5} \), and then computes \((C_{14}, C_{15})\) with the server’s challenge number, \( R'_{S} \), as follows:

\[ C_{14} = EPC_{\text{driver}}' \oplus R_{R_5} \oplus M \]  \hspace{1cm} (25)

\[ C_{15} = PRNG(EPC_{\text{driver}}' \oplus Key_{\text{driver}}' \oplus R'_{S} \oplus R_{R_5}) \]  \hspace{1cm} (26)

Finally, the reader sends message \((C_{14}, C_{15})\) to challenge the driver’s tag, \( T_{\text{driver}} \).

Step 8: After receiving message \((C_{14}, C_{15})\) from the reader, the driver’s tag, \( T_{\text{driver}} \), first decomposes \( C_{14} \) to get the challenge number, \( R'_{R_5} \), of the reader with its EPC and the secret value, \( M \), as follows:

\[ R'_{R_5} = C_{14} \oplus EPC_{\text{driver}}' \oplus M \]  \hspace{1cm} (27)

and then verifies \( C_{15} \)

\[ C_{15} = PRNG(EPC_{\text{driver}}' \oplus Key_{\text{driver}}' \oplus R'_{S} \oplus R_{R_5}) \]  \hspace{1cm} (28)

If Eq.(28) holds, the reader is correct, and then the driver’s tag, \( T_{\text{driver}} \), uses the server’s challenge number, \( R'_{S} \), the random number, \( R'_{T_1} \), generated by itself and the reader’s challenge number, \( R'_{R_5} \), with its secret key, \( Key_{\text{driver}} \), to compute message \( C_{16} \):

\[ C_{16} = PRNG(EPC_{\text{driver}}' \oplus Key_{\text{driver}}' \oplus R'_{S} \oplus R_{T_1} \oplus R'_{R_5}) \]  \hspace{1cm} (29)

Finally, the driver’s tag, \( T_{\text{driver}} \), sends message \( C_{16} \) to the reader.

Step 9: Upon receiving message \( C_{16} \) from the driver’s tag, \( T_{\text{driver}} \), the reader uses the server’s challenge number, \( R'_{S} \); the challenge number, \( R'_{T_1} \), generated by the driver’s tag, \( T_{\text{driver}} \) and the challenge number, \( R_{R_5} \), with the driver’s secret key, \( Key_{\text{driver}} \), to verify \( C_{16} \) as follows:

\[ C_{16} = PRNG(EPC_{\text{driver}}' \oplus Key_{\text{driver}}' \oplus R'_{S} \oplus R'_{T_1} \oplus R_{R_5}) \]  \hspace{1cm} (30)

If Eq.(30) holds, the driver’s tag, \( T_{\text{driver}} \), is legal. Now our scheme has complete the checks for all members (reader, driver, car and container).

### 3 Security analysis

In this section, we discuss the proposed yoking proof scheme and how we can defend against various attacks.
3.1 Preliminaries of security

In the proposed yoking proof system between reader and server, security is based on the RSA assumption. As shown in Subsection 2.3 and 2.4, we utilize two functionalities that include the public key encryption, $E_X()$/decryption, $D_X()$ \[4\][11][12] and the public key signing, $S_X()$/verification, $V_X()$ \[3\][12][23], respectively.

In 1994, Bellare and Rogaway \[4\] introduced an Optimal Asymmetric Encryption (OAEP) Scheme using the RSA assumption \[24\]. The OAEP scheme was believed to achieve semantic security against adaptive chosen cipher attacks. In 2001, Fujisaki et al. \[13\] proved that the OAEP scheme, based on the RSA assumption, is semantically secure against adaptive chosen cipher attacks in the Random Oracle Model \[5\]. They mainly showed that the security of the OAEP scheme can actually be proven under the one-wayness of the RSA function. In our yoking proof system, we use Fujisaki et al.’s Asymmetric Encryption Scheme to provide the public key encryption/decryption function. The security theorem of Fujisaki et al.’s Asymmetric Encryption Scheme is presented below:

**Theorem 1.** In the Random Oracle Model, if an adversary with adaptive chosen cipher capability and a non-negligible advantage can violate the semantic security of the Optimal Asymmetric Encryption (OAEP) Scheme, then there exists a challenger to solve the one-wayness of the RSA function.

**Proof.** For the details of the proof for this theorem refer to \[11\].

For the Adopted Signature Scheme in our yoking proof system, we may use the well-known RSA Probabilistic Signature Scheme (RSA-PSS scheme) \[12\] under the RSA assumption in the Random Oracle Model or the Cramer-Shoup Signature Scheme \[3\][23] without relying on random oracles under the strong RSA assumption. In 2000, Cramer and Shoup \[11\] proposed a signature scheme and proved that the proposed scheme is existentially unforgeable under adaptive chosen-message attacks. In 2003, Fischlin \[12\] improved the Cramer-Shoup Signature Scheme to allow faster signing and verification. Here, we use the well-known RSA-PSS Scheme \[12\] in our yoking proof System. The security theorem of the RSA-PSS Signature Scheme is presented here. The following theorem shows the security of the RSA-PSS based on the security of the RSA assumption \[24\].

**Theorem 2.** In the Random Oracle Model and under the security of the RSA function, the RSA Probabilistic Signature Scheme (RSA-PSS scheme) is existentially enforceable under adaptive chosen-message attacks.

**Proof.** For the details of the proof for this theorem refer to Fischlin (Fischlin, 2003).

3.2 Prevention of attacks analysis

3.2.1 Privacy protection

In the authentication phase, an attacker can intercept messages when the tag sends messages to the reader. On the other hand, the $EPC_{car}$, $EPC_{container}$ and $EPC_{driver}$ are mixed by random numbers $R_T^2, R_T^3, R_R^4$, and the member message, $M$, respectively:

$$C_8 = EPC_{car} \oplus R_T^2 \oplus M$$

$$C_{12} = EPC_{container} \oplus R_T^3 \oplus M$$
Using RFID Yoking Proof to Design...  
Chin-Ling Chen, Chun-Yi Wu, Fang-Yie Leu and Yi-Li Huang

\[ C_{14} = EPC_{driver} \oplus R_{R_5} \oplus M \]  
\[ (25) \]

So, the attacker can’t decompose the messages to ascertain the EPC \( EPC_{driver} \) of the driver’s tag \( T_{driver} \), the EPC \( EPC_{car} \) of the car’s tag \( T_{car} \) or the EPC \( EPC_{car} \) of the container’s tag \( T_{container} \).

### 3.2.2 Resist relay attacks

In the authentication phase, when the reader queries the tags, each tag responds with the corresponding messages to the request. An attacker can eavesdrop on the transmission messages via an insecure channel and store the messages in their memory. Next, the attacker uses the intercepted message to pass the reader’s authentication. However, the replay attack will fail. Here we use the driver’s tag, \( T_{driver} \), to present the scenario as an attacker eavesdropping on the communication messages.

The first legitimate communication:

1. **Step 1:** Reader \( \rightarrow \) Tag \( T_{driver} \) : \((C_{14}, C_{15})\)
   - Tag \( T_{driver} \) \( \rightarrow \) Reader : \( C_{16} \)
   
   Where:

   \[ C_{14} = EPC_{driver} \oplus R_{R_5} \oplus M \]  
   \[ (31) \]

   \[ C_{15} = PRNG(EPC_{driver} \oplus Key_{driver} \oplus R_{S} \oplus R_{R_5}) \]  
   \[ (32) \]

   \[ C_{16} = PRNG(EPC_{driver} \oplus Key_{driver} \oplus R_{S} \oplus R_{T_1} \oplus R_{R_5}) \]  
   \[ (33) \]

When the reader wants to query some tag in the n-th communication, the attacker replays message \( C_{14} \), to spoof the reader, but the attack will fail. The reason is described as follows:

The n-th legitimate communication:

1. **Step 1:** Reader \( \rightarrow \) Tag \( T_{driver} \) : \((C'_{14}, C'_{15})\)
   - Tag \( T_{driver} \) \( \rightarrow \) Reader : \( C'_{16} \)
   
   Where:

   \[ C_{14} = EPC_{driver} \oplus R'_{R_5} \oplus M \]  
   \[ (34) \]

   \[ C_{15}' = PRNG(EPC_{driver} \oplus Key_{driver} \oplus R_{S} \oplus R'_{R_5}) \]  
   \[ (35) \]

   \[ C_{16}' = PRNG(EPC_{driver} \oplus Key_{driver} \oplus R_{S} \oplus R'_{T_1} \oplus R'_{R_5}) \]  
   \[ (36) \]

Since

\[ R_{R_2} \neq R'_{R_2} \]  
\[ (37) \]
We can see

$$C_{16} \neq C_{16}^*$$ (38)

Because the reader uses random number, $R_{R_2}'$ to challenge the driver’s tag, the reader can verify whether the received messages are legal or not. Thus, the attacker fails to pass the reader’s authentication.

### 3.2.3 Resist man-in-the-middle attack

In the authentication phase, an attacker can eavesdrop on transmission messages between the tag and the reader, and then modify the message to counterfeit a legitimate role. The scenario is described as follows:

Step 3: Reader $\rightarrow$ Tag $T_{car}$ : $(C_6, C_7)$

Step 4: Tag $T_{car} \rightarrow$ Reader : $(C_8, C_9)$

Step 5: Reader $\rightarrow$ Tag $T_{container}$ : $(C_{10}, C_{11})$

Step 6: Tag $T_{container} \rightarrow$ Reader : $(C_{12}, C_{13})$

Step 7: Reader $\rightarrow$ Tag $T_{driver}$ : $(C_{14}, C_{15})$

Step 8: Tag $T_{driver} \rightarrow$ Reader : $C_{16}$

The reader computes message $(C_8, C_9)$ as follows:

$$C_8 = EPC_{car} \oplus R_{T_2} \oplus M$$ (13)

$$C_9 = PRNG(EPC_{car} \oplus Key_{car} \oplus R_{R_1}' \oplus R_{T_2})$$ (14)

The attacker can intercept and modify message $(C_8, C_9)$ to counterfeit a reader with $C_8'$ and $C_9'$.

$$C_8 \oplus C_8' = EPC_{car} \oplus R_{T_2} \oplus M \oplus EPC_{car} \oplus R_{T_2}' \oplus M = R_{T_2} \oplus R_{T_2}'$$ (39)

$$C_8' = C_8 \oplus R_{T_2} \oplus R_{T_2}'$$ (40)

Since $C_9$ and $C_9'$ are generated by $PRNG()$:

$$C_9 = PRNG(EPC_{car} \oplus Key_{car} \oplus R_{R_1}' \oplus R_{T_2})$$ (14)

$$C_9' = PRNG(EPC_{car} \oplus Key_{car} \oplus R_{R_1}' \oplus R_{T_2}')$$ (41)

Because the attacker doesn’t know the EPC of the car’s tag, $T_{car}$, the secret key, $Key_{car}$, and the reader’s challenge number, $R_{R_1}'$, the attacker can’t compute a $C_9'$ to match the $C_8'$. Similarly, the tag $T_{driver}$ and $T_{container}$ also use the same method to authenticate others, respectively. Thus, our scheme can resist man-in-the-middle attacks.
3.2.4 Resist tag impersonation attack

When a reader queries the tags, each tag responds with the corresponding messages to the request. An attacker can eavesdrop on the transmission messages via an insecure channel and record the messages in their memory. Then, the attacker uses the messages to pass the reader’s authentication. Assume the attacker eavesdrops on the following communication messages; we use the car’s tag, $T_{\text{car}}$, to describe the scenario:

Tag $T_{\text{car}} \rightarrow$ Reader : $(C_8, C_9)$ where

\[ C_8 = EPC_{\text{car}} \oplus R_{T_2} \oplus M \]  

(13)

\[ C_9 = \text{PRNG}(EPC_{\text{car}} \oplus \text{Key}_{\text{car}} \oplus R'_{R_1} \oplus R_{T_2}) \]  

(14)

When intercepting message $(C_8, C_9)$, the attacker can resend the messages to spoof the reader in the next transaction. However, the reader uses the random number, $R'_{R_1}$, to query the tag. The reader can verify whether the received message is legal or not. Thus, the tag impersonation attack can’t succeed.

3.2.5 Resist tag tracking attack

In our scheme, an attacker can eavesdrop on the transmission messages between the tag and the reader, and then store the messages for later use. If the tag responds again, the attacker compares these new communication messages with the preceding message to ascertain whether the tag is the same. Here we use the car’s tag, $T_{\text{car}}$, to present the scenario as follows:

Tag $T_{\text{car}}$ sends the 1st and n-th communication messages to the reader.

The 1st legitimate transaction:

Step 2: Tag $T_{\text{car}} \rightarrow$ Reader: $(C_8, C_9)$

The n-th legitimate transaction:

Step 2: Tag $T_{\text{car}} \rightarrow$ Reader: $(C'_8, C'_9)$

The attacker can eavesdrop on message $(C_8, C_9)$ and $(C'_8, C'_9)$, but the attacker doesn’t know both message $(C_8, C_9)$ and $(C'_8, C'_9)$ are sent from the same tag. The reason is described as follows:

since

\[ C_8 = EPC_{\text{car}} \oplus R_{T_2} \oplus M \]  

(13)

\[ C'_8 = EPC_{\text{car}} \oplus R'_{T_2} \oplus M \]  

(42)

we can see

\[ C_8 \neq C'_8 \]  

(43)

Since

\[ C_9 = \text{PRNG}(EPC_{\text{car}} \oplus \text{Key}_{\text{car}} \oplus R'_{R_1} \oplus R_{T_2}) \]  

(14)

\[ C'_9 = \text{PRNG}(EPC_{\text{car}} \oplus \text{Key}_{\text{car}} \oplus R'_{R_1} \oplus R'_{T_2}) \]  

(44)
Using RFID Yoking Proof to Design A Supply-Chain Applications for Customs Check

Chin-Ling Chen, Chun-Yi Wu, Fang-Yie Leu and Yi-Li Huang

we can see

\[ C_9 \neq C'_9 \] (46)

Since random number, \( R_{T_2} \) of the car’s tag, \( T_{\text{car}} \), is different for each transaction, both the driver’s tag, \( T_{\text{driver}} \) and the container’s tag, \( T_{\text{container}} \) use different random numbers, respectively. Our scheme can resist the tracking attack.

### 3.2.6 Resist DoS attack

An attacker responds with a message to spoof the reader impersonating a legal member of the group. On the other hand, the attacker interrupts the transmission to interfere with the legal members; these situations result in a faulty proof. The scenario is described as follows:

Step 1: Reader → Tag \( T_{\text{car}} : (C_6, C_7) \)

\[ C_6 = EPC'_{\text{car}} \oplus R_{R_1} \oplus M \] (9)

\[ C_7 = \text{PRNG}(EPC'_{\text{car}} \oplus Key'_{\text{car}} \oplus R_{R_1}) \] (10)

Upon receiving message \((C_6, C_7)\), the car’s tag, \( T_{\text{car}} \) computes the \( R_{R_1} \) and verifies \( C_7 \):

\[ C_7? = \text{PRNG}(EPC'_{\text{car}} \oplus Key'_{\text{car}} \oplus R_{R_1}) \] (12)

Since the car’s tag, \( T_{\text{car}} \’s \), is unique to solve the correct number, \( R_{R_1} \), and the \( T_{\text{car}} \’s \) secret key, \( Key'_{\text{car}} \), only the car’s tag, \( T_{\text{car}} \), can authenticate the message and perform the protocol. Thus, our scheme can resist a DoS attack.

### 3.3 Mechanism analysis

#### 3.3.1 Mutual authentication

Case 1: Reader authenticates tag

The reader uses the random number, \( R_{R_2} \); the tag’s identity, \( EPC'_{\text{driver}} \); the secret value, \( M \), and the tag’s key, \( Key'_{\text{driver}} \), to compute \( C_{14} \) and \( C_{15} \) as follows:

\[ C_{14} = EPC'_{\text{driver}} \oplus R_{R_1} \oplus M \] (25)

\[ C_{15} = \text{PRNG}(EPC'_{\text{driver}} \oplus Key'_{\text{driver}} \oplus R'_{R_1} \oplus R_{R_2}) \] (26)

The driver’s tag, \( T_{\text{driver}} \), uses the tag’s identity, \( EPC'_{\text{driver}} \), and the secret value, \( M \), to decompose message \( C_{14} \) and obtains the reader’s challenge number, \( R_{R_1} \). Then, the driver’s tag, \( T_{\text{driver}} \), uses the random number, \( R_{R_2} \), and computes \( C_{16} \) as follows:

\[ C_{16} = \text{PRNG}(EPC'_{\text{driver}} \oplus Key'_{\text{driver}} \oplus R'_{R_1} \oplus R_{R_2} \oplus R_{T_1} \oplus R'_{R_2}) \] (29)

When receiving, the reader verifies as follows:

\[ C_{16}? = \text{PRNG}(EPC_{\text{driver}} \oplus Key_{\text{driver}} \oplus R'_{R_1} \oplus R_{R_2} \oplus R_{T_1} \oplus R_{R_3}) \] (30)
If Eq.(30) holds, the driver’s tag, \( T_{\text{driver}} \), is correct.

Case 2: Tag authenticates reader

The driver’s tag, \( T_{\text{driver}} \), generates a random number, \( R_{T_1} \), to compute and sends a message \((C_3, C_4, C_5)\) to the reader. Upon receiving the message, the reader computes the random number, \( R_{S}^i \), generated by the server, each member’s EPC \((EPC_{\text{driver}}', EPC_{\text{car}}', EPC_{\text{container}}')\) and the secret key \((\text{Key}_{\text{driver}}', \text{Key}_{\text{car}}', \text{Key}_{\text{container}}')\) as follows:

\[
R_{T_1}^i = C_5 \oplus M \tag{6}
\]

\[
(EPC_{\text{driver}}, EPC_{\text{car}}, EPC_{\text{container}}, \text{Key}_{\text{driver}}', \text{Key}_{\text{car}}', \text{Key}_{\text{container}}', R_{S}^i) = D_{RSK}(C_4 \oplus R_{T_1}^i) \tag{7}
\]

\[
(EPC_{\text{driver}}' \oplus R_{S}^i, EPC_{\text{car}}' \oplus R_{S}^i, EPC_{\text{container}}' \oplus R_{S}^i) \oplus V_{SPK}(C_3 \oplus R_{T_1}^i) \tag{8}
\]

and then sends to the driver’s tag, \( T_{\text{driver}} \). The driver’s tag, \( T_{\text{driver}} \), verifies \( C_{15} \):

\[
C_{15} = PRNG(EPC_{\text{driver}}' \oplus \text{Key}_{\text{driver}}', R_{S}^i \oplus R_{S}^i) \tag{28}
\]

If Eq. (28) holds, the reader is legal. The authentication of other tags is similar to that of a driver’s tag, \( T_{\text{driver}} \).

### 3.3.2 Anonymity

In our protocol, message \((C_1, C_2)\) is transmitted in a secure channel. We do not transmit the tag’s secret value, \( M \), and the tag’s EPC \((EPC_{\text{driver}}, EPC_{\text{car}}, EPC_{\text{container}})\) in plain text. We camouflage EPC codes with the generated random numbers, \( R_{S}, R_{R}, R_{T_i} \) and \((\text{Key}_{\text{driver}}', \text{Key}_{\text{car}}', \text{Key}_{\text{container}}')\) between the sender and receiver in an insecure channel. So, if attackers interrupt this information to analyze the identity of a sender or receiver, he or she only obtains the protected identity of members in the current transaction. The attackers fail to obtain the real identity information of the members.

### 3.3.3 Conform EPCglobal C1G2

In our protocol, tags only use comparison operations, exclusive-or operations and PRNG operations. These operations conform to the EPCglobal C1G2 standards and low-cost. They can decrease the computational load of tags.

### 4 Discussions

In Table 1, we compare our scheme with other yoking proof schemes for known RFID attacks. Our scheme can resist numerous known attacks (Table 1), where other schemes may suffer from these attacks. Notably, our scheme improves upon existing security standards. We can see our protocol is more robust than others.

In Table 2, we compare our mechanism with other yoking proof schemes.

In Table 3, we compare our scheme with other protocols in the context of time complexity. First, we store the server’s information in tags so that the reader can use the information to verify whether the transaction is legal. In Table 1, we show how the proposed scheme can defend against more known
Using RFID Yoking Proof to Design
A Supply-Chain Applications for Customs Check

Chin-Ling Chen, Chun-Yi Wu,
Fang-Yie Leu and Yi-Li Huang

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Prevention of replay attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevention of subset replay attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevention of man-in-the-middle attack</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevention of impersonation tag attack</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevention of forgery attack</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevention of tracking attack</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevention of DoS attack</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1: Security comparison of the related yoking-proof schemes

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag encryption method</td>
<td>HASH</td>
<td>PRNG</td>
<td>PRNG</td>
<td>PRNG</td>
</tr>
<tr>
<td>Mutual authentication</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Off-line</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Anonymity</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Conform to EPCglobal C1G2</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 2: Mechanism comparison of the related yoking-proof schemes

attacks than other schemes. In Table 2, we also elucidate how our protocol offers mutual authentication. For conforming to EPCglobal C1G2 standards, tag uses simple operations to process computing. We also use the RSA to improve security between the reader and server. The related proof is illustrated in Subsection 3.1.1. In such design, we can avoid the session key construction and updated problems. Moreover, we have more tags than other schemes. This is why our scheme requires more computation time.
Using RFID Yoking Proof to Design A Supply-Chain Applications for Customs Check
Chin-Ling Chen, Chun-Yi Wu, Fang-Yie Leu and Yi-Li Huang

 Roles  Schemes

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>$21 T_{\text{XOR}} + 2 T_{\text{PRNG}} + 9 T_{H}$</td>
<td>$21 T_{\text{XOR}} + 2 T_{\text{ADD}} + 23 T_{\text{PRNG}} + 2 T_{\text{COMP}}$</td>
<td>$2 n T_{\text{XOR}} + 3 n T_{\text{PRNG}}$</td>
<td>$30 T_{\text{XOR}} + 6 T_{\text{PRNG}} + 3 T_{\text{COMP}}$</td>
</tr>
<tr>
<td>Reader (server)</td>
<td>$T_{\text{COMP}}$</td>
<td>$T_{\text{SYE}}$</td>
<td>$T_{\text{SYE}}$</td>
<td>$33 T_{\text{XOR}} + 6 T_{\text{PRNG}} + T_{\text{AYS+E}} + T_{\text{ASYD}} + T_{\text{ASYS}} + T_{\text{ASYV}} + 4 T_{\text{COMP}}$</td>
</tr>
</tbody>
</table>

Searching cost for identifying a tag

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Time complexity comparison of the related yoking-proof schemes

NOTES:

$T_{\text{COMP}}$: the time for comparison

$T_{\text{XOR}}$: the time for executing an exclusive-or operation

$T_{\text{PRNG}}$: the time for executing a pseudo random number generation

$T_{H}$: the time for executing a hash function

$T_{\text{ASYE}}$: the time for executing an asymmetric encryption operation

$T_{\text{ASYD}}$: the time for executing an asymmetric decryption operation

$T_{\text{ASYS}}$: the time for executing an asymmetric signature operation

$T_{\text{ASYV}}$: the time for executing an asymmetric verification operation

$n$: the number of group tags

5 Conclusions

In this paper, we propose a novel scheme of proof by demonstrating two or more RFID tags simultaneously for the RFID system. The proposed scheme not only conforms to EPCglobal C1G2 standards, but also resists known attacks, such as replay attacks, man-in-the-middle attacks, impersonation tag attacks, tag tracking attacks and DoS attacks. The proposed scheme can increase the efficiency of transactions through our yoking proof protocol.

Furthermore, our scheme can provide information to the registered device to verify members for processing verification requests without a server. On the basis of the aforementioned factors, using RFID conforming to EPCglobal C1G2 standards to check whether the member belongs, we can reduce manpower requirements for supply-chain applications. In summary, our scheme can provide a convenient, low-cost, and improved security mechanism within customs and supply-chain systems. Since the tags only use lightweight operations (exclusive-or operation and pseudorandom number generation), our scheme can conform to EPCglobal C1G2 standards. The proposed scheme can also resist known attacks.

51
Notably, we expect our scheme to be applied widely in similar applications. For example, we can implement the RFID application with medical management to enhance medical service and decrease harm caused by human factors.

References


Using RFID Yoking Proof to Design A Supply-Chain Applications for Customs Check

Chin-Ling Chen, Chun-Yi Wu, Fang-Yie Leu and Yi-Li Huang


Author Biography

Chin-Ling Chen was born in Taiwan in 1961. He received a B.Sc. degree in Computer Science and Engineering from Feng Cha University in 1991, and the M.Sc. degree and Ph.D. in Applied Mathematics at National Chung Hsing University, Taichung, Taiwan, in 1999 and 2005, respectively. He is a member of the Chinese Association for Information Security. From 1979 to 2005, he was a senior engineer Chunghwa Telecom Co., Ltd. He is currently a professor in the Department of Computer Science and Information Engineering at Chaoyang University of Technology, Taiwan. His research interests include cryptography, network security and electronic commerce. Dr. Chen has published over 50 articles on the above research fields in SCI/SSCI international journals.

Chun-Yi Wu was born in 1987. He received a B.S degree in Computer Science and Information Engineering from Fu Jen Catholic University Department in 2009. He received his M.S degree at the institute of Information Engineering and Computer Science, Chung Hsing University in 2011. His research interests include information security and RFID.
Fang-Yie Leu received his BS, master and Ph.D. degrees all from National Taiwan University of Science and Technology, Taiwan, in 1983, 1986 and 1991, respectively, and another master degree from Knowledge Systems Institute, USA, in 1990. His research interests include wireless communication, network security, Grid applications and Chinese natural language processing. He is currently a workshop organizer of CWECS and MCNCS workshops, a professor of Tunghai University, Taiwan, and director of database and network security laboratory of the University. He is also a member of IEEE Computer Society.

Yi-Li Huang received his master degree from National Central University of Physics, Taiwan, in 1983. His research interests include security of network and wireless communication, solar active-tracking system, pseudo random number generator design and file protection theory. He is currently a senior instructor of Tunghai University, Taiwan, and director of information security and grey theory laboratory of the University.