*Journal of Information Hiding and Multimedia Signal Processing*
**Ubiquitous International**
 ISSN 2073-4212
Volume **4**, Number **3**, July **2013**

# Security Analysis of a new Ultra-lightweight RFID Protocol and Its Improvement

Xu Zhuang[1], Zhi-Hui Wang[2], Chin-Chen Chang[3,4], Yan Zhu[5]

[1]School of Information Science and Technology
Southwest Jiaotong University
Chengdu, Sichuan, China

[2]School of Software
Dalian University of Technology
Dalian, Liaoning,China
Economy and Technology Development Area
Dalian, China, 116620
wangzhihui1017@gmail.com
+86-411-87571636

[3]Department of Information Engineering and Computer Science
Feng Chia University
Taichung City 40724, Taiwan

[4]Department of Computer Science and Information Engineering
Asia University
Taichung 41354, Taiwan

[5]School of Information Science and Technology
Southwest Jiaotong University
Chengdu, Sichuan, China

Received January, 2013; revised March, 2013

Abstract. *Retrieval of texture images, especially those with different orientation and scale changes, is a challenging and important problem in image analysis. This paper adopts spiking cortical model (SCM) to explore geometrical invariant texture retrieval schemes based on Discrete Cosine Transform (DCT) coefficients of pulse images. The series of pulse images, outputs of SCM, have a robust talent for extracting edge, segment and texture which are inherent in the original images, but they are large 2-dimensional image data so that it is difficult tSome ultra-lightweight RFID protocols have recently been developed. Unlike other RFID protocols, ultra-lightweight protocols generally only need the simplest bitwise operations in the tag side, such as XOR, AND, and OR. In 2012, Tian etal. proposed a new ultra-lightweight RFID protocol named RAPP (RFID authentication protocol with permutation) using a new bitwise operation Permutation in the protocol, which can achieve high security and privacy as claimed. Unfortunately, because of the incomplete session that might occur in RAPP, we present a replay attack which can lead to de-synchronization between a tag and the database, which means the tag can no longer be authenticated by any reader. In addition, we also present a simple de-synchronization attack that can break the synchronization state between a tag and the database, like the replay attack. Some potential threats resulting in more security concerns from RAPP are illustrated by using two properties of Permutation revealed in this paper. We also provide some countermeasures for RAPP to withstand attacks mentioned in the paper.*

**Keywords:** RFID; ultra-lightweight; security analysis; permutation; RAPP.

1. **Introduction.** Radio Frequency Identification (RFID) systems use radio signals to identify a special target (called a tag). RFID systems usually have the ability to write and read the data in the target. Due to the low computation cost and ease of implementation, RFID systems have been widely used in many applications, such as access control systems, e-passports, and food security. However, security and privacy issues are two serious obstacles for the development of RFID. Security problems refer to whether an RFID system has the ability to withstand various attacks, such as DoS attacks, replay attacks, and de-synchronization attacks. Privacy problems are more special in RFID systems as the tag is attached to a certain product or its owner. For each authentication, the tag uses its identifier for the authentication process. Thus, a malicious attacker can use this information to track a certain tag or reveal its locations privacy information. To deal with these problems, many researchers have proposed authentication protocols in order to achieve high performance in both security and privacy. However, many of these protocols do not consider the computation cost from the tag side; thus, although they can guarantee the security and privacy in their protocols, due to the high computation cost on the tag side, they are not applicable. Chien [3] classified the RFID protocols into four groups: full-fledged, simple, lightweight, and ultra-lightweight. Unlike the other three groups of RFID protocols, which generally require a random number generator and some functions like the one-way hashing function and Cyclic Redundancy Code (CRC) on the tag side, ultra-lightweight RFID systems only need the simplest bitwise operations in the tag side, such as XOR, AND, and OR. In this paper, we focus on the ultra-lightweight group, which requires the lowest computation cost on the tag side.

Lopez *etal.* [4-6] proposed a series of ultra-lightweight RFID protocols called LMAP, $M^2AP$, and EMAP in 2006. All three protocols are designed to guarantee data confidentiality and data integrity of the protocol as well as prevent the RFID systems from various kinds of attacks, such as man-in-the-middle and replay attacks. However, Li and Wang [10] pointed out that the LMAP and $M^2AP$ are vulnerable to de-synchronization and full disclosure attacks [10]. Later, as reported in [11], EMAP was also found to be vulnerable to de-synchronization and full disclosure attacks. In 2007, Chien [3] proposed an ultra-lightweight protocol called SASI, which provides strong authentication and strong data integrity. Unfortunately, SASI is vulnerable to traceability, replay, and DoS attacks, as reported in [2, 7, 9]. In 2012, Tian *etal.* [12] proposed a new ultra-lightweight RFID protocol named RAPP, which uses a new bitwise operation called *Permutation* in the protocol; the authors claimed that this protocol had the ability to withstand various attacks and provide strong data confidentiality and integrity. However, utilizing the property of the invariance of Hamming weight of *Permutation*, Avoine and Carpent [1] presented a traceability attack for it. More seriously, Wang *etal.* [8] proposed a way to fully compromise the secrets of a tag in RAPP, although their method is not efficient as it requires about $2^{30}$ times for an attacker to communicate with the target tag. In [13], Ahmadian *etal.* found an efficient way to lead a de-synchronization attack on RAPP.

In this paper, we reveal a new replay attack that can break the synchronization between a tag and the database. Thus, the tag might fall into a DoS state in which the tag can no longer be authenticated by any reader. In addition, we explore the property found in [13] and give another simple de-synchronization attack. We then show some potential threats that can lead to some other security concerns in RAPP. The rest of the paper is organized as follows: Section 2 presents the notations used in this paper and an overview of RAPP protocol. Section 3 shows a replay attack and a de-synchronization attack that may cause a tag to fall into the DoS state. Some countermeasures to withstand our attacks are given in Section 4, and the conclusions are summarized in Section 5.

2. **Overview of RAPP.** RAPP includes three parties: the readers, the tags, and a back-end database. Generally, we assume the channel between a reader and the back-end database to be secure while an attacker can steal all the messages transmitted between the reader and a tag. All notations used in this paper are shown in Figure 1.

**Notations:**

Assume all binary strings used in RAPP have the length of $n$ bits, where $n > 0$.

$\oplus$ : Bitwise XOR operation.

$wt(x)$: The Hamming weight of the binary string $x$.

$Rot(x, y)$: Circular left rotate binary string $x$ by $wt(y)$ bit(s).

$Per(x, y)$: The permutation operation of $x$ according to $y$.

$[A]_i$: The $i^{th}$ bit of binary string $A$.

$[0]_{i,j}$: Represents an $n$-bits string that all bits are 0 except for the position $i$ and $j$, where $i > j > 0$.

$-$: Subtraction modulo $n$.

FIGURE 1. Notations used in the paper

| | |
|---|---|
| Reader's initialization state<br>$\{IDS^{old}, K_1^{old}, K_2^{old}, K_3^{old}, IDS^{new}, K_1^{new}, K_2^{new}, K_3^{new}\}$ | Protocol run:<br>Step1: Reader $\rightarrow$ Tag: "Hello"<br>Step2: Tag $\rightarrow$ Reader: $IDS$<br>Step3: Reader $\rightarrow$ Tag: $A$, $B$<br>Step4: Tag $\rightarrow$ Reader: $C$<br>Step5: Reader $\rightarrow$ Tag: $D$, $E$ |
| Tag's initialization state<br>$\{IDS, K_1, K_2, K_3\}$ | |
| **Computations**:<br>$A = Per(K_2, K_1) \oplus n_1$; $B = Per(K_1 \oplus K_2, Rot(n_1, n_1)) \oplus Per(n_1, K_1)$;<br>$C = Per(n_1 \oplus K_1, n_1 \oplus K_3) \oplus ID$; $D = Per(K_3, K_2) \oplus n_2$;<br>$E = Per(K_3, Rot(n_2, n_2)) \oplus Per(n_1, K_3 \oplus K_2)$ | |
| **Updating for reader**:<br>Notation $IDS^m$ represents that $IDS^m$ is matched in the database and it equals to $IDS^{old}$ or $IDS^{new}$.<br>$IDS^{old} = IDS^m$, $K_1^{old} = K_1^m$, $K_2^{old} = K_2^m$, $K_3^{old} = K_3^m$<br>$IDS^{new} = Per(IDS^{old}, n_1 \oplus n_2) \oplus K_1^{old} \oplus K_2^{old} \oplus K_3^{old}$<br>$K_1^{new} = Per(K_1^{old}, n_1) \oplus K_2^{old}$<br>$K_2^{new} = Per(K_2^{old}, n_2) \oplus K_1^{old}$<br>$K_3^{new} = Per(K_3^{old}, n_1 \oplus n_2) \oplus IDS^{old}$ | **Updating for tag**:<br>$IDS^* = Per(IDS, n_1 \oplus n_2) \oplus K_1 \oplus K_2 \oplus K_3$<br>$K_1^* = Per(K_1, n_1) \oplus K_2$<br>$K_2^* = Per(K_2, n_2) \oplus K_1$<br>$K_3^* = Per(K_3, n_1 \oplus n_2) \oplus IDS$<br>$IDS = IDS^*$, $K_1 = K_1^*$, $K_2 = K_2^*$<br>$K_3 = K_3^*$ |

FIGURE 2. Computations and updating of RAPP

The bitwise operation Permutation is defined as follows [12]:
**Definition** A and B are two $n$-bit binary strings, where $A = a_1a_2...a_n, a_i \in \{0,1\}, i = 1,2,...,n$ , $B = b_1b_2...b_n, b_i \in \{0,1\}, i = 1,2,...,n$ , Assume $wt(B) = m(0 \leq m \leq n)$ , and $b_{k_1} = b_{k_2} = ... = b_{k_m} = 1$, $b_{k_{m+1}} = b_{k_{m+2}} = ... = b_{k_n} = 0$, where $1 \leq k_1 < k_2 < \ldots < k_m \leq n$ and $1 \leq k_{m+1} < k_{m+1} < \ldots < k_n \leq n$. Then, $Per$(A, B) is

$$Per(A,B) = a_{k_1}a_{k_2}...a_{k_m}a_{k_n}a_{k_{n-1}}...a_{k_{m+2}}a_{k_{m+1}}.$$

Figure 3 helps clarify the definition.

RAPP is based on the use of index-pseudonym ($IDS$), which is an index of a table containing all the secrets of a tag. There are two main advantages to using $IDS$: 1) The database can identify a tag by its $IDS$ without any information of its unique identifier; in addition, the updating of $IDS$ achieves the goal of preventing the privacy leakage problem of a tag; and 2) after receiving the $IDS$ of a tag, the database can check the legality of the $IDS$ with computation complexity $O(1)$ instead of $O(n)$. RAPP is quite an efficient RFID protocol as it just uses the simple bitwise operations XOR and Hamming weight based *Rotation* and *Permutation* on the tag side. Complex operations such as a random number generator are only needed on the reader side.

Figure 2 presents all the computations in the RAPP protocol. Each step is described as follows:

Step1: Reader sends a "Hello" message to a tag to initiate a new protocol run.

Step2: After receiving the query message, the tag responds to the reader with its $IDS$.

Step3: The reader checks whether the received $IDS$ matches the database. If the $IDS$ matches, the reader generates a random number $n_1$ and then computes messages $A$ and $B$ based on which $IDS$ is found in the database ($IDS^{old} or IDS^{new}$). Then the reader transmits $A$ and $B$ to the tag. If the $IDS$ is not matched, the reader ends this session.

Step4: Upon receiving the messages, the tag extracts $n_1$ from $A$ and uses local secret keys to compute $B'$ and check whether the equation $B = B'$ is true or not. If it is true, the tag computes message $C$ and sends it to reader. Otherwise, the tag terminates the protocol.

Step5: The reader checks the received $C$ with its secrets. If the reader accepts $C$, it generates a random number $n_2$, messages $D$ and $E$. Then the reader updates all its secrets and transmits $D$ and $E$ to the tag. Otherwise, the reader ends the session.

Step6: The tag gets $n_2$ from message $D$ and checks $E$. If the tag accepts $E$, it updates all its secrets; otherwise, it terminates the protocol.

## 3. Security analysis of RAPP protocol.

In this section, we give a replay attack and a de-synchronization attack that can both cause a tag to fall into a denial-of-service (DoS) state. This means that the tag can no longer be authenticated as valid. In addition, we discuss some potential attacks that might cause more security concerns in RAPP. Before presenting our attacks, it is necessary to discuss some properties of the operation *Permutation*.

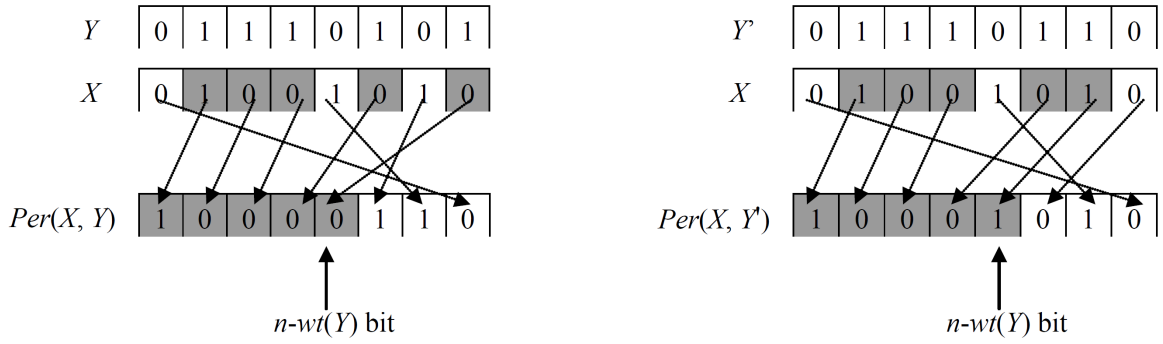
Property 1: For $Per(X,Y)$, the $lsb$ of $Y$ does not influence the output of $Per(X,Y)$.
Property 2: For $Per(X,Y)$ and $Per(X,Y')$ where $Y' = Y \oplus [0]_{1,0}$, if $[Y]_{1 \neq} [Y]_0$, which means $wt(Y) > 0$, $[Per(X,Y')]_{n-wt(Y)} = [Per(X,Y)]_{n-wt(Y)-1}$ and $[Per(X,Y')]_{n-wt(Y)-1} = [Per(X,Y)]_{n-wt(Y)}$.

Property 1 is obviously true, so we just provide the proof for Property 2.

Proof for Property 2:
Because $[Y]_1 \neq [Y]_0, Y' = Y \oplus [0]_{1,0}$, we have $[Y]_1 \oplus 1 \neq [Y]_0 \oplus 1$, hence $[Y']_1 \neq [Y']_0, wt(Y) = wt(Y')$.
If $[Y]_1 = 0$, we get $[Y]_0 = 1, [Y']_1 = 1, [Y']_0 = 0$ and $[Per(X,Y)]_{n-wt(Y)} = [X]_0$, $[Per(X,Y)]_{n-wt(Y)-1} = [X]_1$
Therefore, $[Per(X,Y')]_{n-wt(Y)} = [X]_1, [Per(X,Y')]_{n-wt(Y)-1} = [X]_0$. So that, we conclude that $[Per(X,Y')]_{n-wt(Y)} = [Per(X,Y)]_{n-wt(Y)-1}$ and $[Per(X,Y')]_{n-wt(Y)-1} = [Per(X,Y)]_{n-wt(Y)}\sqrt{2}$
If $[Y]_0 = 0$, we get $[Y]_1 = 1, [Y']_0 = 1, [Y']_1 = 0$ and $[Per(X,Y)]_{n-wt(Y)} = [X]_1$, $[Per(X,Y)]_{n-wt(Y)-1} = [X]_0$.
Therefore, $[Per(X,Y')]_{n-wt(Y)} = [X]_0, [Per(X,Y')]_{n-wt(Y)-1} = [X]_1$. Hence, we conclued that $[Per(X,Y')]_{n-wt(Y)} = [Per(X,Y)]_{n-wt(Y)-1}$ and $[Per(X,Y')]_{n-wt(Y)-1} = [Per(X,Y)]_{n-wt(Y)}$.

Q.E.D

As shown in Figure 3, we use the example in [12] to describe the Property 2.




$Y' = Y \oplus [0]_{1,0}$, the results $Per(X,Y)$ and $Per(X,Y')$ show that the $(n - wt(Y))^{th}$ and $(n - wt(Y) - 1)^{th}$ bits of $Per(X,Y)$ and $Per(X,Y')$ are exchanged.

FIGURE 3. Example of Property 2

Avoine and Carpent [12] presented a traceability attack to RAPP. This attack utilizes the property that Permutation is Hamming weight-invariant, which means that $wt(Per(X,Y)) = wt(X)$. Thus, it is possible for an attacker to trace a victim tag. Based on the traceability in RAPP, we next describe a replay attack, which can lead a tag to a DoS state.

3.1. **Replay attack.** In RAPP, the end messages are sent from the reader to the tag, and it is not easy to ensure data integrity between the reader and the tag because a reader cannot check whether the tag has updated its secrets or not. Because of the incomplete session and the traceability of RAPP, an attacker can first steal some messages transmitted between a reader and a tag and then use these messages to impersonate a valid reader to initiate some new sessions, which can make the tag carry out its updating while the reader does not, thereby breaking the synchronization between the reader and the tag.

Figure 4 shows all the procedures of our replay attack. We assume that there are initialization states for a reader $R$ and a tag $T$ shown in the first two boxes in Figure 4. Then, the first protocol run between $R$ and $T$ begins. In the first protocol run, an attacker notes down all the messages in the transition and intercepts the messages $D_2$

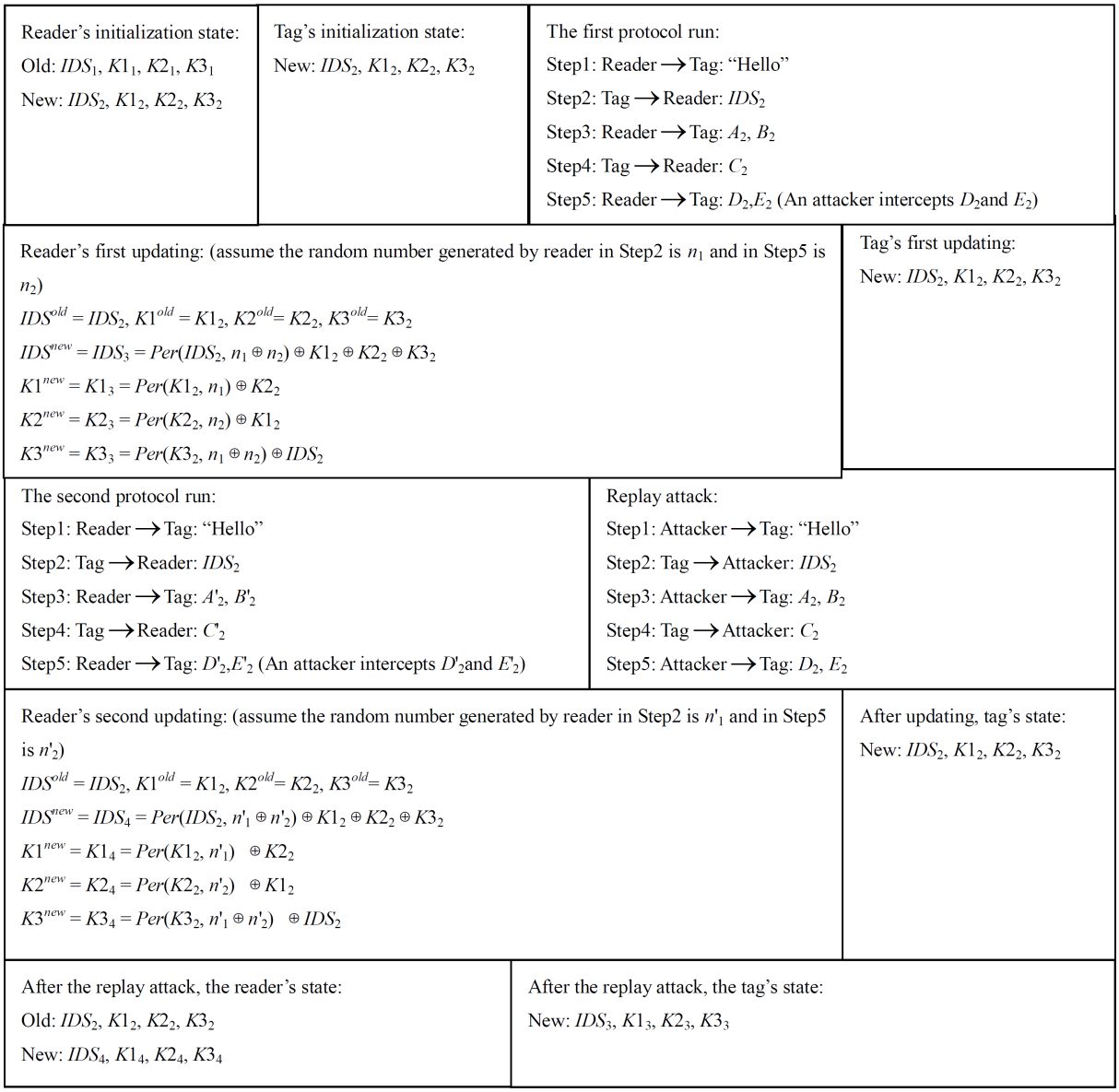



FIGURE 4. Example of replay attack

and $E_2$ in the last step, so that $T$ would not update its secrets while $R$ does. The first updating for $T$ and $R$ are shown in Figure 4.

In the second protocol run, the attacker also intercepts the last two messages transmitted in the last step, which causes $R$ to update all its secrets again while $T$ does not. The second updating shown in Figure 4 presents the results of the second protocol run. When $T$ leaves the valid range of the reader, the attacker impersonates a valid reader to send $T$ a hello message to initiate a new protocol run. The replay attack is described as follows:

Step1: Attacker → Tag: Hello. The attacker sends a hello message to $T$ to initiate a new protocol run.

Step2: Tag → Attacker: $IDS_2$. Upon receiving the query, $T$ responds the attacker with its $IDS_2$.

Step3: Attacker → Tag: $A_2$, $B_2$. After receiving $IDS_2$, the attacker sends the stolen messages $A_2$ and $B_2$ to $T$.

Step4: Tag $\rightarrow$ Attacker: $C_2$. Since $A_2$ and $B_2$ are generated by a valid reader using secrets $K1_2$, $K2_2$, $K3_2$, and a random number $n_1$, the tag can extract $n_1$ from $A_2$ and accept $B_2$. So $T$ will compute $C_2$ and send it to the attacker.

Step5: Attacker $\rightarrow$ Tag: $D_2$, $E_2$. The attacker can ignore the received $C_2$ and then send $TD_2$ and $E_2$. After receiving $D_2$ and $E_2$, $T$ updates all its secrets to $IDS_3$, $K1_3$, $K2_3$, and $K3_3$.

As previously discussed, after this replay attack, the secrets stored in $T$ are different from the secrets stored in the database, which causes $T$ to fall into a DoS state. However, in this replay attack, the attacker needs to trace a target tag, which is possible in RAPP, as pointed out in [12].

3.2. **De-synchronization attack: Changing messages $A$, $B$ and $E$.** In [13], Ahmadian and Salmasizadeh revealed Property 2 shown in this paper using another version that does not contain the relationship between the Hamming weight of $Y$ and the output of $Per(X, Y \oplus [0]_{1,0})$. Using this property, Ahmadian and Salmasizadeh [13] presented a simple but efficient de-synchronization attack for RAPP by changing message $D$. We found that this attack can be also applied to change message $A$, which we will explore next.

Step1: Reader $\rightarrow$ Tag: "Hello"
Step2: Tag $\rightarrow$ Reader: *IDS*
Step3: Reader $\rightarrow$ Tag: $A'$, $B'$, where $A' = A \oplus [0]_{i,i\text{-}1}$ and $B' = B \oplus [0]_{j,k}$, $0 \le i \le n\text{-}1$, $0 \le j \le n\text{-}1$, $0 \le k \le n\text{-}1$, and $j \neq k$.
Step4: Tag $\rightarrow$ Reader: $C$
Step5: Reader $\rightarrow$ Tag: $D$, $E'$, where $E' = E \oplus [0]_{m,l}$, $0 \le m \le n\text{-}1$, $0 \le l \le n\text{-}1$,

FIGURE 5. The second attack

Assume that a reader $R_1$ and a tag $T_1$ have the initialization state shown in Figure 2 and there is a successive protocol run between $R_1$ and $T_1$ without the last step in the protocol. Thus, an attacker can note down all the messages ($IDS$, $A$, $B$, $C$, $D$, $E$) transmitted between $R_1$ and $T_1$ but does not send $D$ and $E$ to $T_1$ in the last step. This would cause $R_1$ to update its secrets while $T_1$ does not. After the protocol is run, the state of $R_1$ is $\{IDS^{old} = IDS, K_1{}^{old} = K_1, K_2{}^{old} = K_2, K_3{}^{old} = K_3, IDS^{new} = IDS^*, K_1{}^{new} = K_1{}^*, K_{2new} = K_2{}^*, K_3{}^{new} = K_3{}^*\}$ and the state of T1 is $\{IDS, K_1, K_2, K_3\}$. After $T_1$ leaves the valid range of $R_1$, an attacker can send a hello message to $T_1$ to impersonate a valid reader to initiate a new protocol run. Upon receiving the query, $T_1$ responds to it with its $IDS$. Then the attacker sends $T_1$ messages $A'$ and $B'$. Note that the operation on $A$ actually changes the corresponding bits of the random number $n_1$ generated by $R_1$ in Step 3. If $T_1$ accepts $B'$, it would respond to the attacker with message $C$. As message $C$ in RAPP is used to authenticate the tag by a reader, the attacker can ignore it and send messages $D$ and $E'$ to $T_1$. If $T_1$ accepts $E'$, it will update its secrets by using a modified $n_1$. This would cause a de-synchronization between the $T_1$ and the database. Next, we analyze the success rate of this attack, as shown in Figure 5.

As evident, changing $[A]_i$ and $[A]_{i-1}$ actually changes $[n_1]_i$ and $[n_1]_{i-1}$. We use the notation $n_1{}'$ to represent the changed $n_1$. If $[n_1]_i$ and $[n_1]_{i-1}$ are different, $[n_1]_i \oplus 1$ and $[n_1]_{i-1} \oplus 1$ are also different. In such a case, $wt(n_1) = wt(n_1{}')$. Thus, there is a 1/2 probability that $wt(n_1) = wt(n_1{}')$. If $[n_1]_i$ and $[n_1]_{i-1}$ are rotated to the least two significant bits after $Rot(n_1, n_1)$, the least two significant bits between $Rot(n_1, n_1)$ and

$Rot(n_1', n_1')$ are different. For $i = 1$ and $[n_1]_1 \neq [n_1]_0$, since $1 \leq wt(n_1) \leq n-1$, the probability that $[n_1]_1$ and $[n_1]_0$ are shifted to the least two significant bits after $Rot(n_1, n_1)$ is 0; for $i \neq 1$, the probability that $[n_1]_i$ and $[n_1]_{i-1}$ are shifted to the least two significant bits after $Rot(n_1, n_1)$ is $1/(n\text{-}1)$. Thus, in the case of $[n_1]_i \neq [n_1]_{i-1}$, the probability that $[n_1]_i$ and $[n_1]_{i-1}$ are shifted to the least two bits after $Rot(n_1, n_1)$ is

$$\frac{1}{n} \times 0 + \frac{n-1}{n} \times \frac{1}{n-1} = \frac{1}{n} \tag{1}$$

As the $lsb$ of $Rot(n_1, n_1)$ and $Rot(n_1', n_1')$ do not influence the output of $Per(K_1 \oplus K_2, Rot(n_1, n_1))$ and $Per(K_1 \oplus K_2, Rot(n_1', n_1'))$, utilizing Property 2, we can find that the $(n - wt(n_1))^{th}$ and $(n - wt(n_1) - 1)^{th}$ bits of $Per(K_1 \oplus K_2, Rot(n_1', n_1'))$ are exchanged compared with $Per(K_1 \oplus K_2, Rot(n_1, n_1))$. If $[Per(K_1 \oplus K_2, Rot(n_1, n_1))]_{n-wt(n_1)} = [Per(K_1 \oplus K_2, Rot(n_1, n_1))]_{n-wt(n_2)-1}$, which indicates that $[K_1 \oplus K_2]_0 = [K_1 \oplus K_2]_1$ (see the proof of Property 2), there is a 1/2 probability that $Per(K_1 \oplus K_2, Rot(n_1, n_1))$ is unchanged. As such, there is a $1/4n$ probability that $Per(K_1 \oplus K_2, Rot(n_1, n_1))$ is unchanged. In such a case, two bits of $Per(n_1, K_1)$ are changed, which would cause two bits of $B'$ to change compared with $B$, but it is not easy to find the positions of these two bits. For $A'$, which causes $Per(K_1 \oplus K_2, Rot(n_1, n_1))$ to remain unchanged, the attacker can randomly change two bits of $B$ to check whether $T_1$ would respond to message $C$ or not. This would be successful at most $C_n^2$ times, where $n$ is the length of a binary string used in the protocol. So, for each test of $n_1'$, the probability that $T_1$ accepts $B'$ is $1/4n$ at most $C_n^2$ times. Once $T_1$ responds to the attacker's message $C$, it indicates that $T_1$ accepts message $B'$. An attacker must note down the $A'$ and $B'$, which leads to responses from $T_1$(denoted as $A'_s$ and $B'_s$). Then, the attacker sends messages $D$ and $E'$ to $T_1$. Since $Per(K_3, Rot(n_2, n_2))$ is unchanged and there should be two bits of $Per(n_1, K_3 \oplus K_2)$ that are changed, the attacker can test all possible $E'$ to check whether $T_1$ has done its updating by sending $T_1$ a new hello message and checking the $IDS$. If the attacker fails in the last step, s/he can repeat these steps using $A'_s$, $B'_s$, $D$, and another $E'$ in the attack. Once an attacker has found $A'_s$ and $B'_s$, $T_1$ would accept $E'$ at most $C_n^2$ times attacks.

Based on this discussion, it is obvious that this attack must be successful if an attacker can find the $A'_s$ and $B'_s$ when there are no other mechanisms to prevent the protocol. For each test of $n_1'$ and $B'$, the success rate of acceptance of $B'$ by $T_1$ is $\frac{1}{4n \times C_n^2}$. Once an attacker has found $A'_s$ and $B'_s$, for each test of message $E'$, the success rate of acceptance of $E'$ by $T_1$ is $\frac{1}{C_n^2}$.

3.3. **Potential threats.** In addition to the two aforementioned attacks and the one presented in [13], an attacker can utilize some potential threats to attack RAPP. In this subsection, we reveal some potential insecure factors of RAPP.

The attack shown in [13], which changes message $D$ and tries to lead the tag to update its secrets with a modified $n_2$, still works when $K3_0 \neq K3_1$. In such a case, two bits of $E$ are exchanged, and one can test it at most $C_n^2$ times. If one can find the correct bits that are exchanged, referring to Property 2, s/he finds the Hamming weight of $n_2$. This theory is also applied to find the Hamming weight of $n_1$, but it may cost more times as there is an interference $Per(n_1, K_1)$ in message $B$. Yet as one can prevent the updating for a tag and there is a responding message $C$, it is quite useful for an attacker to test the changed bits. Moreover, once an attacker has found the Hamming weight of $n_1$, it is possible for her/him to reveal some potential relationships between the changed $n_1$ and secret $K_1$ by controlling the Hamming weight of $n_1$ and keeping $K_1$ unchanged. All these potential insecure factors might threaten the security and privacy of RAPP.

4. **Patches for RAPP.** The replay attack shown in Section 3 is based on the traceability of RAPP; thus, one can consider thatXif the RAPP can resist the traceability attackXour replay attack might not work in such a case. However, in real life, it is quite easy for an attacker to trace a tag in a very short time as a tag is usually attached to a certain product or its owner. Thus, we must find a more efficient way to withstand this replay attack.

In the RFID system, we cannot really generate a complete message indicating that both the reader side and the tag side are carrying out their updating. Therefore, there are only two choices for the protocols' designers: the end messages are transmitted from a reader to a tag or from a tag to a reader. In the mechanism used in RAPP, the end messages are transmitted from a reader to a tag, which is considered as to be more secure than from a tag to a reader. However, in such a case, the reader cannot know the state of the tag. To avoid potential attacks caused by the incomplete session, some auxiliary data can be stored on the reader side rather than the tag side.

4.1. **Denial of the old $IDS$.** The state information with a tag used in LMAP+ [6] is useful in RAPP, but this method needs much more space to store the potential index-pseudonyms. Thus, we present a more efficient way to resist the de-synchronization state caused by our replay attack.

If a reader receives an old $IDS$ from a tag, indicating that the tag did not get messages $D$ and $E$ in the last protocol run and, thus, the tag did not update its secrets, it is not applicable for a reader to accept the old $IDS$ and then begin the protocol again as it might be a trap made by an attacker. A more secure way is to make the tag and reader synchronous as soon as possible rather than treat it as a normal protocol run. To achieve this goal, the reader needs to store the random numbers $n_1$ and $n_2$ generated in the last protocol's execution. The reader then uses the numbers $n_1$ and $n_2$ to begin a new protocol run, meaning that the messages $A$, $B$, $C$, $D$, and $E$ will be the same as the last protocol. Note that, the so-called new protocol run is not a genuine authentication process as the reader received an old $IDS$, as we said before, and the reader must reject it straight away. Indeed, the purpose of the new protocol run is to make the tag and the reader synchronized.

| | |
|---|---|
| State of $R$:<br>Old: $IDS_1$, $K1_1$, $K2_1$, $K3_1$<br>New: $IDS_2$, $K1_2$, $K2_2$, $K3_2$ | Last protocol run:<br>Step1: $R \rightarrow T$: "Hello"<br>Step2: $T \rightarrow R$: $IDS_1$<br>Step3: $R \rightarrow T$: $A_1$, $B_1$<br>Step4: $T \rightarrow R$: $C_1$<br>Step5: $R \rightarrow T$: $D_1$, $E_1$ (These two messages are intercepted or lost) |
| State of $T$:<br>$IDS_1$, $K1_1$, $K2_1$, $K3_1$ | |

FIGURE 6. Example for the patch of denial of the old $IDS$

For example, the states of $R$ and $T$ and all messages generated in last protocol run between them are shown in Figure 6. In the next protocol, upon receiving the old $IDS$ of $T$, using the same random number $n_1$, $R$ sends $A_1$ and $B_1$ to $T$, who will respond to $R$ with $C_1$. In a normal protocol, $C$ is used to authenticate the tag by a reader; however, in such case, as the $IDS$ of $T$ received by $R$ is the old one, $R$ accepts $C_1$ but it does not offer "service" for the tag $T$ and then sends $D_1$ and $E_1$ to $T$. If $D_1$ and $E_1$ can be received by $T$ in the new protocol run, then $T$ will update its secrets to $IDS_2$, $K1_2$, $K2_2$, and $K3_2$, and $R$ does not need update its secrets as the random numbers used in the new protocol

run are same as the last protocol. Therefore, after this protocol, the secrets between $R$ and $T$ are synchronous and $T$ can be authenticated by any reader again.

In this way, our replay attack is useless and will not lead to some other replay attacks. In RAPP, after the mutual authentication phase (the reader accepts message $C_1$), the reader must update its secrets. An attacker can get all messages ($IDS_1$, $A_1$, $B_1$, $C_1$, $D_1$, and $E_1$) transmitted in the protocol, but as the $IDS_1$ is the old $IDS$ stored in the reader after updating, all messages related to $IDS_1$ cannot cause a real authentication protocol but rather serve as a trigger to lead the reader to synchronize the tag. Due to the constant space cost $2n$, where $n$ is the length of binary string used in the protocol, this scheme is more efficient than the one using state information proposed in [6], which costs $kn$ space, where $k$ is the number of times that a tag remains in the uncertainty state.

4.2. **Modifications of messages $A$ and $B$.** The main idea of the second attack and the attack described in [13] is to utilize the two properties of the operation *Permutation* presented in Section 3. In order to withstand these attacks, we must ensure that changing messages $A$ and $D$ will cause a chain changing all factors of messages $B$ and $E$ so that one cannot find out how many and which positions are changed for $B$ and $E$. In our improvement scheme, messages $B$ and $E$ are modified as:

| | LMAP[6] | M$^2$AP[4] | EMAP[5] | SASI[3] | RAPP[12] | Our improvement |
|---|---|---|---|---|---|---|
| Types of computation operations | +, ⊕ , OR | +, ⊕ , AND, OR | ⊕ , AND, OR | +, ⊕ , OR, *Rot* | ⊕ , *Rot, Per* | ⊕ , *Rot, Per* |
| Storage requirement | 6 *L* | 6 *L* | 6 *L* | 7 *L* | 5 *L* | 5 *L* |
| Communication messages | 2 *L* | 3 *L* | 3 *L* | 2 *L* | 2 *L* | 2 *L* |
| Resistance to de-synchronization attacks | No | No | No | No | No | Yes |
| Resistance to disclosure attacks | No | No | No | No | No | Yes |
| Resistance to tag tracking | No | No | No | No | No | Yes |

$L$ is the length of messages used in protocols

FIGURE 7. Comparisons among our work and others'

$$B = Per(K_2 \oplus n_1, Rot(n_1, n_1)) \oplus Per(n_1, n_1 \oplus K_1), \tag{2}$$

$$E = Per(K_3 \oplus n_2, Rot(n_2, n_2)) \oplus Per(n_1, n_2 \oplus K_2). \tag{3}$$

Now, we will analyze the security of these modifications. For the second attack, changing $[n_1]_i$ and $[n_1]_{i-1}$ would cause $[K_2 \oplus n_1]_i$, $[K_2 \oplus n_1]_{i-1}$, $[n_1 \oplus K_1]_i$ and $[n_1 \oplus K_1]_{i-1}$ to be changed, where $i \geq 0$. Assuming that $[n_1]_i$ and $[n_1]_{i-1}$ are different and are shifted to the least two significant bits after $Rot(n_1, n_1)$, the least two significant bits between $Rot(n_1, n_1)$ and $Rot(n'_1, n'_1)$ are different, where $n'_1 = n_1 \oplus [0]_{i,i-1}$. However, for the modified $B$, since $[K_2 \oplus n_1]_i$ and $[K_2 \oplus n_1]_{i-1}$ are also changed, the output of $Per(K_2 \oplus n_1, Rot(n_1, n_1))$ must be uncertain. In addition, the output of $Per(n_1, n_1 \oplus K_1))$ also becomes quite uncertain as not only $n_1$ and $n_1 \oplus K_1$ would be changed, but also the Hamming weight of $n_1 \oplus K_1$ will be uncertain. Thus, due to the modification of message $B$ in our improvement scheme, it is quite difficult for an attacker to find out which positions of message Bare changed after the changing of $n_1$. This means that the attacker cannot easily produce a $B'$ that can be accepted by the tag. This analysis is also applied to message $E$. Ultimately, the improved scheme can prevent the system from the two aforementioned attacks.

In Figure 7, we present comparisons among our work and others' using the format made in [12] without considering the storage cost caused by storing extra information to prevent schemes from replay attack.

5. **Conclusions.** Security and privacy are the two most important issues for RFID systems. For ultra-lightweight RFID protocol, it is quite necessary to minimize the cost for the tags as even the simplest bitwise operations can be performed in these tags due to the extremely limited resources. To reduce the overhead of a tag, some new bitwise operations are used in ultra-lightweight protocols, such as Hamming weight-based *Rotation* and *Permutation*. But the most serious drawback of these two operations is their invariant Hamming weight. Moreover, since most the existing ultra-lightweight RFID protocols do not offer mechanisms to prevent the systems from replay attack except for storing the old messages in the tag or reader side, it is quite easy for an attacker to use some stolen messages to break the synchronization between a tag and the database.

In this paper, we presented two attacks that can cause a tag to fall into the DoS state for a recently proposed ultra-lightweight RFID protocol RAPP, meaning that the tag can no longer be authenticated by any reader. The first attack utilizes the incomplete session in RAPP while the second one utilizes the two properties shown in this paper. Furthermore, we discussed some potential threats for RAPP by revealing the Hamming weight of the two random numbers used in the protocol. As we can see, the recently proposed RAPP is vulnerable to the de-synchronization and replay attacks presented in this paper. We also give some countermeasures for RAPP to withstand the attacks discussed in this paper. The idea of withstanding the replay attack is quite useful for many other schemes as it does not increase the computation cost in the tag. The modification versions of messages $B$ and $E$ are guides for RFID protocols' designers to conceal relationships among all factors in a message. The security analysis demonstrated that our improved scheme is more secure than RAPP.

## REFERENCES

[1] G. Avoine, and X. Carpent, Yet another ultralightweight authentication protocol that is broken, *Proc. of the 8th Workshop on RFID Security*, 2012.

[2] H. N. Sun, W. C. Ting, and K. H. Wang, On the security of chien's ultralightweight RFID authentication protocol, *IEEE Trans. Dependable and Secure Computing*, vol. 8, no. 2, pp. 315-317, 2011.

[3] H. Y. Chien, SASI: A new ultralightweight RFID authentication protocol providing strong authentication and strong integrity, *IEEE Trans. Dependable and Secure Computing*, vol. 4, no. 4, pp. 337-340, 2007.

[4] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, $M^2AP$: a minimalist mutual-muthentication protocol for low-cost RFID tags, *Proc. of the 3rd international conference on Ubiquitous Intelligence and Computing*, pp. 912-923, 2006.

[5] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, EMAP: An efficient mutual authentication protocol for low-cost RFID tags, *Proc. of the international conference on On the Move to Meaningful Internet Systems*, pp. 352-361, 2006.

[6] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. E. Tapiador, and A. Ribagorda, LMAP: a real lightweight mutual authentication protocol for low-cost RFID tags, *Proc. of the 2nd Workshop on RFID Security*, 2006.

[7] R. C. W. Phan, Cryptanlysis of a new ultralightweight RFID authentication protocol-SASI, *IEEE Trans. Dependable and Secure Computing*, vol. 6, no. 4, pp. 316-320, 2009.

[8] S. H. Wang, Z. J. Han, S. J. Liu, and D. W. Chen, Security analysis of RAPP: an RFID authentication protocol based on Permutation, *Cryptology ePrint Archive 2012/327*, available at http://eprint.iacr.org/2012/327.pdf.

[9] T. Cao, E. Bertino, and H. Lei, Security analysis of the SASI protocol, *IEEE Trans. Dependable and Secure Computing*, vol. 6, no. 1, pp. 73-77, 2009.

[10] T. Li, and G. Wang, Security analysis of two ultra-lightweight RFID authentication protocols, *IFIP Advances in Information and Communication Technology*, springer, vol. 232, pp. 109-120, 2007.

[11] T. Li, and R. Deng, Vulnerability analysis of EMAP-an efficient RFID mutual authentication protocol, *Proc. of the 2nd International Conference on Availability, Reliability and Security*, pp. 238-245, 2007.

[12] Y. Tian, G Chen, and J. Li, A new ultralightweight RFID authentication protocol with permutation, *Journal of IEEE Communications Letters*, vol. 16, no. 5, pp. 702-705, 2012.

[13] Z. Ahmadian, M. Salmasizadeh, and M. R. Aref, Desynchronization attack on RAPP ultralightweight authentication protocol, *Cryptology ePrint Archive 2012/490*, available at http://eprint.iacr.org/2012/490.pdf.