Vulnerability Analysis of EMAP-An Efficient RFID Mutual Authentication Protocol

Conference Paper · January 2007

DOI: 10.1109/ARES.2007.159 · Source: DBLP

Robert H. Deng
Singapore Management University
672 PUBLICATIONS **22,121** CITATIONS
SEE PROFILE

Vulnerability Analysis of EMAP-An Efficient RFID Mutual Authentication Protocol

Tieyan Li
Institute for Infocomm Research (I^2R)
21 Heng Mui Keng Terrace, Singapore 119613
litieyan@i2r.a-star.edu.sg

Robert Deng
Singapore Management University
469 Bukit Timah Road, Singapore 259756
robertdeng@smu.edu.sg

Abstract

In this paper, we analyze the security vulnerabilities of EMAP, an efficient RFID mutual authentication protocol recently proposed by Peris-Lopez et al. *[15]. We present two effective attacks, a* de-synchronization attack *and a* full-disclosure attack*, against the protocol. The former permanently disables the authentication capability of a RFID tag by destroying synchronization between the tag and the RFID reader. The latter completely compromises a tag by extracting all the secret information stored in the tag. The de-synchronization attack can be carried out in just round of interaction in EMAP while the full-disclosure attack is accomplished across several runs of EMAP. We also discuss ways to counter the attacks.*

1 Introduction

Recent aggressive deployment of Radio Frequency Identification (RFID) systems in a variety of applications has raised many concerns about privacy. Chief among them is clandestine or unregulated scanning of RFID tags (*w.r.t.*, transponders) attached on objects being identified. RFID tags emit unique identifiers which could be used to track the itineraries of the objects or individuals carrying them. One solution to alleviate the problem is to change the identifier of a tag on every scan. On the other hand, however, RFID readers (*w.r.t.*, interrogators) need to ensure that the tags being queried are authentic, not compromised, cloned or spoofed. To address both security issues, many *privacy enhanced RFID (mutual) authentication protocols* have been put forward (refer to Section 5). The challenge in the design of such protocols is to strike the right balance between security and cost. RFID tags are generally low cost with extremely limited resources. They cannot perform standard cryptographic operations, such as encryption, that are used extensively in security solution for ordinary computing environments.

In this paper, we analyze the security properties of EMAP, an efficient mutual authentication protocol for low-cost RFID tags recently proposed by Peris-Lopez *et al.* in [15]. Different from the majority of existing solutions [17, 13, 11, 12, 1, 5] using standard cryptographic primitives, this protocol is *ultra-lightweight*, since it uses only simple bitwise operations to achieve mutual authentication between a RFID reader and a tag as well as protect the privacy of the tag's identifier. Consequently, only about 300 gates in the tag are required to implement the protocol. Note that a low-cost tag (with price $0.05 - 0.1$ US dollar) normally has $5K$ gates among them less than $1K$ gates for available for security operations. EMAP is claimed to be effective in "man-in-the-middle attack prevention" and "forgery resistance".

In this paper we present our security analysis of EMAP. First, we show that the protocol suffers from *de-synchronization attack* which destroys "synchronization" between RFID database [1] and a RFID tag, and therefore permanently disables the authentication capability of the tag. The attack can be carried out easily by only eavesdropping a single round of the protocol message exchange. Next, we present a more serious *full-disclosure attack*. By interacting with the reader and the tag across several protocol runs, this attack allows an attacker to extract the ID as well as all other secret information from the tag. Finally, to defend against the above attacks, we point out several potential countermeasures.

The rest of this paper is organized as follows. Section 2 provides a review of the EMAP protocol. Section 3 illustrates our detailed analysis on EMAP's vulnerabilities. Section 4 discusses several countermeasures. Related work on RFID security and privacy is reviewed in Section 5. Finally, conclusion remarks are drawn in Section 6.

[1] As in [15], the terms "database" and "RFID reader" are used synonymously; they both refer to the counterpart of a tag in the authentication protocol.

2 Review of EMAP

EMAP [15] is a highly efficient RFID mutual authentication protocol using only bitwise XOR ($\oplus$), bitwise OR ($\vee$) and bitwise AND ($\wedge$) operations. Costly operations such as multiplications and hash functions are not required at all, and random number generation is only done by the reader. In EMAP, a dynamic index-pseudonym (IDS) (m-bit length, $m = 96$ is cited for an EPC [4] tag ID) is used as the index to a table (a row) where all the information about a tag is stored. Each tag is associated with a key, which is divided in four parts each with 96 bits ($K = K1||K2||K3||K4$). As the IDS and the key (K) must be updated after each successful protocol execution, they need 480 bits of rewritable memory (EEPROM) in total. A ROM memory to store the 96-bit static identification number (ID) is also required. Due to the fact that most low-cost tags are passive, communications must be initiated by readers. EMAP assumes that both the backward and the forward channel can be listened by an attacker, but the communication channel between the reader and the database is secure. The nth protocol run of EMAP is shown in Table 1.

Tag identification: Reader $\longrightarrow$ Tag: *hello* Tag $\longrightarrow$ Reader: $IDS^{(n)}_{tag(i)}$
EMAP mutual authentication: Reader $\longrightarrow$ Tag: $A\|\|B\|\|C$ Tag $\longrightarrow$ Reader: $D\|\|E$
where: $A = IDS^{(n)}_{tag(i)} \oplus K1^{(n)}_{tag(i)} \oplus n1$ $B = (IDS^{(n)}_{tag(i)} \vee K2^{(n)}_{tag(i)}) \oplus n1$ $C = IDS^{(n)}_{tag(i)} \oplus K3^{(n)}_{tag(i)} \oplus n2$ $D = (IDS^{(n)}_{tag(i)} \wedge K4^{(n)}_{tag(i)}) \oplus n2$ $E = (IDS^{(n)}_{tag(i)} \wedge n1 \vee n2) \oplus ID_{tag(i)} \bigoplus_{I=1}^{4} KI^{(n)}_{tag(i)}$

Table 1. EMAP Protocol

The protocol has three stages: tag identification, mutual authentication, index-pseudonym and key updating.

- *Tag Identification*: The reader identifies the tag by sending a *hello* message to the tag, which answers by sending its current index-pseudonym (IDS). By means of this IDS, only an authorized reader is able to access the tag's corresponding secret key ($K = K1||K2||K3||K4$) from the database, which is used in next stage.

- *Mutual Authentication*: The reader first generates two random numbers $n1$ and $n2$. With $n1$, $n2$, and the subkeys $K1$, $K2$ and $K3$, the reader generates the submessages A, B and C, and then sends them to the tag. With the submessages A and B, the tag can authenticate the reader and obtain $n1$. Once the reader is authenticated, the tag obtains the random number $n2$ from the submessage C, and then generates messages D and E. Note that the tag's static identifier is transmitted securely to the reader in E. The tag is successfully authenticated if the reader extracts a valid ID from E.

- *Index-Pseudonym and Key Updating*: After the reader and the tag have authenticated each other, they carry out the index-pseudonym and key updating operations based on the following equations:

$$\begin{aligned}
IDS^{(n+1)}_{tag(i)} &= IDS^{(n)}_{tag(i)} \oplus n2 \oplus K1^{(n)}_{tag(i)} \\
K1^{(n+1)}_{tag(i)} &= K1^{(n)}_{tag(i)} \oplus n2 \oplus ([ID_{tag(i)}]_{[1:48]}|| \\
&\quad F_p(K4^{(n)}_{tag(i)})||F_p(K3^{(n)}_{tag(i)})) \\
K2^{(n+1)}_{tag(i)} &= K2^{(n)}_{tag(i)} \oplus n2 \oplus (F_p(K1^{(n)}_{tag(i)})|| \\
&\quad F_p(K4^{(n)}_{tag(i)})||[ID_{tag(i)}]_{[49:96]}) \\
K3^{(n+1)}_{tag(i)} &= K3^{(n)}_{tag(i)} \oplus n1 \oplus ([ID_{tag(i)}]_{[1:48]}|| \\
&\quad F_p(K4^{(n)}_{tag(i)})||F_p(K2^{(n)}_{tag(i)})) \\
K4^{(n+1)}_{tag(i)} &= K4^{(n)}_{tag(i)} \oplus n1 \oplus (F_p(K3^{(n)}_{tag(i)})|| \\
&\quad F_p(K1^{(n)}_{tag(i)})||[ID_{tag(i)}]_{[49:96]})
\end{aligned}$$

where $F_p(X)$ is a parity function: the 96-bit number X is divided into twenty-four 4-bit blocks. A parity is generated for each block, with a total of 24 parity bits. In the above equations, $[ID_{tag(i)}]_{[j:k]}$ denotes bit sequence from the j-th to the k-th positions of $ID_{tag(i)}$.

In [15], the authors presented some security analysis and claimed that EMAP is secure against the man-in-the-middle attack, replay attack and forgery attack. In the next section, we show that the claims unfortunately don't hold.

3 Vulnerabilities of EMAP

First, we remark that EMAP in its present form is not robust, since the tag doesn't know if D and E are indeed received or verified by the reader. If they are not received or verified successfully, the reader will not update the tag's entry in the database, while the tag would have updated its memory. Obviously, the storages at the tag and the reader will be out of synchronization. But this is more about a practical assumption problem (See more discussions on threat model in section 4.1.), not as a serious security problem. To patch the problem, a standard two-phase commitment message exchange can be carried out. In the following, we will present the true vulnerabilities of EMAP.

3.1 *De-synchronization Attack*

To provide privacy protection, most RFID authentication protocols update a tag's ID and other secret information after a successful protocol run. This update is performed in the database as well as in the tag. So synchronization of secret information between the database and the tag is crucial for subsequent authentications. Protocol malfunctions, such as that discussed above, might leave the both sides in an unsynchronized state. The *de-synchronization attack*, to be introduced below, is a malicious action by an attacker which intentionally causes the database and a tag out of synchronization.

Attack 1: Changing message C. We start by presenting the simplest *de-synchronization attack*: a man-in-the-middle first eavesdrops on the protocol message exchange to obtain $A||B||C$. It then changes $A||B||C$ to $A||B||C'$, where $C' = C \oplus [I]_0$ and $[I]_0 = [000 \cdots 001]$ [2]. Similarly, the attacker changes the reply D and E from the tag to D' and E', respectively. This procedure is depicted in Table 2.

EMAP mutual authentication:
Reader $\longrightarrow$ Tag: $A\|\|B\|\|C'$
Tag $\longrightarrow$ Reader: $D'\|\|E'$
where:
$n2' = n2 \oplus [I]_0$
$C' = C \oplus [I]_0 \quad D = (IDS^{(n)}_{tag(i)} \wedge K4^{(n)}_{tag(i)}) \oplus n2'$
$D' = D \oplus [I]_0$
$E = (IDS^{(n)}_{tag(i)} \wedge n1 \vee n2') \oplus ID_{tag(i)} \bigoplus_{I=1}^{4} KI^{(n)}_{tag(i)}$
$E' = E \oplus [I]_0$

Table 2. 1-bit *de-synchronization attack* against EMAP

We say that the attack is successful if D' and E' are both accepted by the reader, or it is a failure if they are not accepted (note that C' is always accepted by the tag in this attack). Now we analyze the success rate of the attack. At the tag side, the attack doesn't affect the first round of the protocol interaction: "tag identification". In the second round, when the tag receives the message $A||B||C'$, it can still authenticate the reader as A and B are intact. However, the tag will get a wrong random number $n2'$ (as $C' = C \oplus [I]_0$, we get $n2' = n2 \oplus [I]_0$). The tag will accept this value and compute the reply based on $n2'$ (refer to the values of D and E in table 2). The reader will get $D' = D \oplus [I]_0 = (IDS^{(n)}_{tag(i)} \wedge K4^{(n)}_{tag(i)}) \oplus n2$ and accept it as correct value. Also, the reader will get $E' = (IDS^{(n)}_{tag(i)} \wedge n1 \vee n2') \oplus [I]_0 \oplus ID_{tag(i)} \bigoplus_{I=1}^{4} KI^{(n)}_{tag(i)}$. Obviously, the reader will accept E' if $result_1 = (IDS^{(n)}_{tag(i)} \wedge n1 \vee n2)$ equals to $result_2 = (IDS^{(n)}_{tag(i)} \wedge n1 \vee n2') \oplus [I]_0$. We list below the truth table of the relevant parameters:

$[IDS^{(n)}_{tag(i)}]_0$	$[n1]_0$	$[n2]_0$	$[n2']_0$	$[result_1]_0$	$[result_2]_0$
0	0	0	1	0	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	0	1	0

Table 3. Truth table of the least significant bit of $result_1$ and $result_2$

From Table 3, the attack has an average 75% success rate for E' being accepted, no matter what the values of $IDS^{(n)}_{tag(i)}$, $n1$ and $n2$ are. However, since $IDS^{(n)}_{tag(i)}$ is known, if $[IDS^{(n)}_{tag(i)}]_0 = 0$, the attack succeeds with 100% rate and if $[IDS^{(n)}_{tag(i)}]_0 = 1$, the success rate is only 50%. Assume that the attack is successful, that means the reader accepts the values of D' and E' and it needs to update the tag's secret information in the database with the pair $(n1, n2)$. However, the tag uses another pair $(n1, n2')$ to update its secrets. *E.g.*, $IDS^{(n+1)}_{tag(i)} = IDS^{(n)}_{tag(i)} \oplus n2^{(')} \oplus K1^{(n)}_{tag(i)}$. It is obvious that there is a mismatch at the secret storages on the tag and the database (refer to table 4). So far, the simplest attack assumes that only the least significant bit of the message C is toggled. The purpose is mainly for illustrating the efficacy of the attack. In fact, the attack can be extended to toggle a single bit of C at any bit position j, so that it can be a general attack with the same success rate. Most seriously, given that $IDS^{(n)}_{tag(i)}$ is known, an adversary can choose to toggle the bit at position j of C such that $[IDS^{(n)}_{tag(i)}]_j = 0$, to achieve 100% success rate.

For on-going nth protocol run of EMAP, an adversary can eavesdrop the index-pseudonym of the tag $IDS^{(n)}_{tag(i)}$ and intercept the message C. Instead of sending the message C to the tag, the adversary toggles the j-th bit of C to get C' as $C' = C \oplus [I]_j$ (in which, $[IDS^{(n)}_{tag(i)}]_j = 0$ for any $j \in \{0, 95\}$). The new messages $A||B||C'$ are then sent to the tag. Upon receiving the reply messages D and E from the tag, the adversary changes them to $D' = D \oplus [I]_j$ and $E' = E \oplus [I]_j$, respectively. Then, the adversary sends $D'||E'$ to the reader. As per analysis above, it has 100% success rate for the reader to accept the message. A successful attack may change the tag's secret status on the database, i. e., de-synchronize the database and the tag with this generalized attack.

[2] Set the more significant 95 bits of I as 0 and the least significant bit as 1, or equivalently toggle the least significant bit of C.

Attack 2: Changing messages A and B. This attack targets on $n1$. In this case, the attacker intercepts the messages $A||B||C$ and sends $A'||B'||C$ to the tag, where $A' = A \oplus [I]_j$ and $B' = B \oplus [I]_j$ (toggling the j-th bit of A and B). Since $A = IDS^{(n)}_{tag(i)} \oplus K1^{(n)}_{tag(i)} \oplus n1$ and $B = (IDS^{(n)}_{tag(i)} \vee K2^{(n)}_{tag(i)}) \oplus n1$, the tag will get $n1' = n1 \oplus [I]_j$ from both A' and B'. Thus, the tag authenticates the reader and generates the reply messages D and E. From the truth table 3, we know that if $[IDS^{(n)}_{tag(i)}]_j = 0$, the attacker does not need to do anything but forwards them to the reader, and the reader will accept them with 100% rate. In the case that $[IDS^{(n)}_{tag(i)}]_j = 1$, the attacker can send either $D||E$ or $D||E'$ ($E' = E \oplus [I]_j$) to the reader with 50% success rate for each of them to be accepted by the reader. Upon a successful attack, both the reader and the tag need to update their secret information. The reader will perform update using the pair $(n1, n2)$, while the tag uses $(n1', n2)$. This will cause mismatch in the next execution of the authentication protocol (refer to Table 4).

Attack 3: A combined attack. Naturally, the above attacks can be combined in which the attacker changes $n1$ and $n2$ simultaneously. In this case, the attacker intercepts the messages $A||B||C$ and sends $A'||B'||C'$ to the tag. Also, the attacker intercepts the reply messages $D||E$ and sends $D'||E'$ (where $M' = M \oplus [I]_j, \forall M \in \{A, B, C, D, E\}$) to the reader. The success rate is 100% if j is chosen such that $[IDS^{(n)}_{tag(i)}]_j = 0$. Consequently, the updating pair at the reader is $(n1, n2)$, while at the tag is $(n1', n2')$. The effects on updating on both the database and tag are summarized in Table 4.

Attacks	Secrets stored on Reader	Secrets stored on Tag(i)
Attack 1	$[IDS, K1, K2, K3, K4]$	$[IDS', K1', K2', K3, K4]$
Attack 2	$[IDS, K1, K2, K3, K4]$	$[IDS, K1, K2, K3', K4']$
Attack 3	$[IDS, K1, K2, K3, K4]$	$[IDS', K1', K2', K3', K4']$

Table 4. Updated secret values in the database and in the tag after the attacks at round (n), in which $M' \neq M$ ($\forall M \in \{IDS, K1, K2, K3, K4\}$), due to $n2'$ only, $n1'$ only or both of $n1'$ and $n2'$.

All the above attacks are based on toggling 1 bit only. They can be extended to toggle multiple bits simultaneously, as long as any one bit is selected as described above, this multi-bit *de-synchronization attack* works equally effective as that of the 1-bit attacks.

3.2 *Full Disclosure Attack*

EMAP implicitly assumes that a RFID tag is stateless, i. e., it does not remember the state of the EMAP protocol execution. That means we can repeatedly run a uncomplete protocol many times with any tag. The full disclosure attack takes advantage of this stateless nature of tags, to extract all the secret information in the tags.

Plot of the attack. The objective of the attack is to obtain a tag's ID, as well as all the secret values associated with it (*w.r.t.*,$K1$, $K2$, $K3$ and $K4$). Since a tag's ID is hidden inside message E, the attacker has to derive all other parameters in E and then solve the equation to find $ID_{tag(i)}$. Initially, this seems like an impossible task, because all secret values are random numbers in the current protocol run and will be updated in the next protocol run. *How to derive all the secret values in one or multiple protocol runs* is the key to fully disclosing the ID of a tag. As we will see, exchanged messages in a single protocol run in fact disclose quite a bit of secret information to the attacker. By launching the attack across several protocol runs, the attacker is able to obtain all secret information of the tag with high probability. The *full disclosure attack* consists of four stages. Stage 1 derives some bit values of the random number $n2$; stage 2 is more subtle and derives the other bit values of $n2$; stage 3 based on $n2$ derives as much as possible the tag's secret information in a single protocol run; and the last stage derives all the secret information including the ID of the tag.

Stage 1: Deriving "half" of $n2$. For the nth protocol run, an attacker impersonates a legitimate reader and gets the current IDS of a tag. Using this valid IDS the attacker impersonates the tag to get a valid message $A||B||C$ from a legitimate reader. The attacker then intercepts the reply messages $(D||E)$. Since $D = (IDS^{(n)}_{tag(i)} \wedge K4^{(n)}_{tag(i)}) \oplus n2$ and $IDS^{(n)}_{tag(i)}$ is a known value, we can derive some bit values of $n2$ from the bitwise expression. Specially, let ϕ be the set of bit positions in which the corresponding bit values in $IDS^{(n)}_{tag(i)}$ are 0 and τ be the set of bit positions in which the corresponding bit values in $IDS^{(n)}_{tag(i)}$ are 1, we have $\phi = \{j|[IDS^{(n)}_{tag(i)}]_j = 0, \forall j \in \{0, 95\}\}$ and $\tau = \{k|[IDS^{(n)}_{tag(i)}]_k = 1, \forall k \in \{0, 95\}\}$. Thus, we derive the bitwise expression of D as:

$$\Rightarrow \quad [D]_j = [n2]_j \quad (\forall j \in \phi) \qquad (1)$$

$$\Rightarrow \quad [D]_k = [K4^{(n)}_{tag(i)}]_k \oplus [n2]_k \quad (\forall k \in \tau) \qquad (2)$$

Obviously, "half"[3] of $n2$ is derived directly from D in equation (1). Another "half" of $n2$ is not directly derived from equation (2) since $K4^{(n)}_{tag(i)}$ is unknown. Following the same approach, we can derive "half" of $n1$, since $[B]_k = [n1]_k$, $(\forall k \in \tau)$. Next, we show a more subtle attack to fully derive the value of $n2$.

[3]Suppose $IDS^{(n)}_{tag(i)}$ is randomly chosen, its hamming weight could be nearly "half" ($\lfloor m/2 \rfloor$).

Stage 2: Deriving the other "half" of $n2$. This attack makes use of the *de-synchronization attack* (i. e., **Attacks 1, 2, 3**) introduced in the last section. There are two interactions between the attacker and the tag in this stage, as illustrated in Fig. 1.

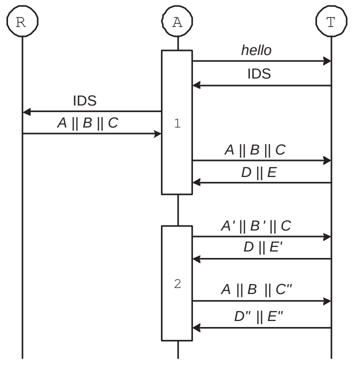

Figure 1. *Full Disclosure Attack* **in a single protocol run**

Firstly, the attacker launches **Attack 2** by toggling multiple bits of $n1$. This is done by sending $A'||B'||C$ to the tag and intercepts the reply messages $D||E'$, where A' is set as $[A']_\tau = [A]_\tau \oplus [I]_\tau$, or equivalently toggling all the bit values at positions of τ on A (where $[IDS^{(n)}_{tag(i)}]_\tau = 1$). Similarly, B' is set as $[B']_\tau = [B]_\tau \oplus [I]_\tau$, for which, we set $n1'$ as $[n1']_\tau = [n1]_\tau \oplus [I]_\tau$. Upon receiving the request, the tag obtains $n1'$ and $n2$ and replies with D and E'. Since E' is calculated from $n1'$ and $n2$, it must be different from E intercepted in stage 1. Let $result_3 = (IDS^{(n)}_{tag(i)} \wedge n1 \vee n2)$ and $result_4 = (IDS^{(n)}_{tag(i)} \wedge n1' \vee n2)$, we get the following truth table (Table 5) for all the k-th bit value ($k \in \tau$) of $result_3$ and $result_4$. Clearly, if $[n2]_k = 0$ ($k \in \tau$), the toggle operation on $[n1]_k$ causes a toggle operation on $[result_3]_k$ (therefore, on $[E]_k$, too). Else if $[n2]_k = 1$ ($k \in \tau$), the toggle operation on $[n1]_k$ doesn't change the bit value of $[result_3]_k$. Bit by bit, we compare the bit values in $[result_3]_k$ and $[result_4]_k$ (actually by comparing $[E]_k$ and $[E']_k$), and derive the bit information as $[n2]_k = \{0|[E]_k \neq [E']_k, 1|[E]_k = [E']_k\}$ ($\forall k \in \tau$). To this end, $n2$ is fully disclosed with only 1 interaction with the tag. Consequently, some other secret values, $K3$ and $[K4]_\tau$, are derived by solving the expressions of C and Equation (2).

Secondly, the attacker uses a similar method to **Attack 1** by toggling multiple bits in $n2$ (refer to Fig. 1, the second interaction with the tag.) and obtains "half" (*w.r.t.,* $[n1]_\tau$) of $n1$ (note that we can derive the same "half" of $n1$ in stage 1). From the expression of A, we further derive the "half" of $K1$ as $[K1]_\tau$. The details are not presented here in order to keep the presentation compact. The other "half" (*w.r.t.,* $[n1]_\phi$) of $n1$ is still unknown; however, we will show in stage 3 that knowing this other "half" is not important as it can be eliminated from the final equations.

$[IDS^{(n)}_{tag(i)}]_k$	$[n1]_k$	$[n1']_k$	$[n2]_k$	$[result_3]_k$	$[result_4]_k$
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	0	1	1	1

Table 5. Truth table of the k-th bit value ($k \in \tau$) of $result_3$ and $result_4$

Stage 3: Deriving secrets in a single protocol run. We regard round (n) only in this stage. Now, the known parameters are $[n1]_\tau$, $n2$, $[K1]_\tau$, $K3$ and $[K4]_\tau$. While the unknown parameters are $[n1]_\phi$, $[K1]_\phi$, $K2$, $[K4]_\phi$, and ID. As mentioned above, we need to solve $ID_{tag(i)}$ in expression E, which is also a half by half process. From $E = (IDS^{(n)}_{tag(i)} \wedge n1 \vee n2) \oplus ID_{tag(i)} \bigoplus_{I=1}^{4} KI^{(n)}_{tag(i)}$, we eliminate it to get the following bitwise expressions:

$$[E]_j = [n2]_j \oplus [ID_{tag(i)}]_j \bigoplus_{I=1}^{4} [KI^{(n)}_{tag(i)}]_j \qquad (\forall j \in \phi) \quad (3)$$

$$[E]_k = ([n1]_k \vee [n2]_k) \oplus [ID_{tag(i)}]_k \bigoplus_{I=1}^{4} [KI^{(n)}_{tag(i)}]_k \qquad (\forall k \in \tau) \quad (4)$$

In Equation (3), one of the unknown parameters is $[K1^{(n)}_{tag(i)}]_\phi \oplus [K2^{(n)}_{tag(i)}]_\phi$ that needs to be solved first. Fortunately, from A and B, we have $[A]_\phi = [K1^{(n)}_{tag(i)}]_\phi \oplus [n1]_\phi$ and $[B]_\phi = [K2^{(n)}_{tag(i)}]_\phi \oplus [n1]_\phi$. Eliminating $[n1]_\phi$, we get $[K1^{(n)}_{tag(i)}]_\phi \oplus [K2^{(n)}_{tag(i)}]_\phi = [A]_\phi \oplus [B]_\phi$. We move the unknown parameters to the left hand side and the known parameters to the right hand side and Equation (3) is transformed as:

$$[ID_{tag(i)}]_\phi \oplus [K4^{(n)}_{tag(i)}]_\phi = [n2]_\phi \oplus [E]_\phi \oplus [A]_\phi \oplus [B]_\phi \oplus [K3^{(n)}_{tag(i)}]_\phi \quad (5)$$

Similarly, Equation (4) is transformed as:

$$[ID_{tag(i)}]_\tau \oplus [K2^{(n)}_{tag(i)}]_\tau = ([n1]_\tau \vee [n2]_\tau) \oplus [E]_\tau \oplus [K1^{(n)}_{tag(i)}]_\tau \oplus [K3^{(n)}_{tag(i)}]_\tau \oplus [K4^{(n)}_{tag(i)}]_\tau \quad (6)$$

Since $[K4^{(n)}_{tag(i)}]_\phi$ and $[K2^{(n)}_{tag(i)}]_\tau$ are still unknown, we can not derive $ID_{tag(i)}$ in a single protocol run. The reason is that the unknown "halves" of $[K2^{(n)}_{tag(i)}]$ and $[K4^{(n)}_{tag(i)}]$ are hidden in B and D by the $\vee$ or $\wedge$ operations. Thus, like $ID_{tag(i)}$, they are being used only once in E in a single protocol run. Apparently, we need more protocol runs to fully extract the secrets.

Stage 4: Deriving secrets in multiple protocol runs. Now the attacker initiates the $n+1$th protocol run with the tag using the information it learns in the last stage. As described in Section 2, the index pseudonym updating algorithm is $IDS^{(n+1)}_{tag(i)} = IDS^{(n)}_{tag(i)} \oplus n2 \oplus K1^{(n)}_{tag(i)}$. By eavesdropping a valid $IDS^{(n+1)}_{tag(i)}$, and given that $n2$ is known, $K1^{(n)}_{tag(i)}$ can be derived. From A and B obtained in the nth protocol run, we recover the complete $n1$ and $[K2^{(n)}_{tag(i)}]_{\phi}$. Thus far, we derive all secret values in the nth protocol in the tag except $[K4^{(n)}_{tag(i)}]_{\phi}$, $[K2^{(n)}_{tag(i)}]_{\tau}$ and $ID_{tag(i)}$. We list below in (Table 6) the recovered secrets in the nth and the $n+1$th protocol runs.

Recall the updating algorithm on $K1^{(n)}_{tag(i)}$. With the known values in Table 6, denote $\mathcal{L}$ as the set including the most significant 48 bits of $ID_{tag(i)}$, we directly derive:

$$[ID_{tag(i)}]_{\mathcal{L}} = [K1^{(n+1)}_{tag(i)}]_{\mathcal{L}} \oplus [K1^{(n)}_{tag(i)}]_{\mathcal{L}} \oplus [n2]_{\mathcal{L}} \quad (7)$$

However, deriving the right set $\mathcal{R}$ (the least significant 48 bits) of $ID_{tag(i)}$ is not straightforward. With the updating algorithms of $K2^{(n+1)}_{tag(i)}$ and $K4^{(n+1)}_{tag(i)}$, we can derive:

$$[ID_{tag(i)}]_{\mathcal{R}} = [K2^{(n+1)}_{tag(i)}]_{\mathcal{R}} \oplus [K2^{(n)}_{tag(i)}]_{\mathcal{R}} \oplus [n2]_{\mathcal{R}} \quad (8)$$

$$[ID_{tag(i)}]_{\mathcal{R}} = [K4^{(n+1)}_{tag(i)}]_{\mathcal{R}} \oplus [K4^{(n)}_{tag(i)}]_{\mathcal{R}} \oplus [n1]_{\mathcal{R}} \quad (9)$$

Furthermore, with the known values in Table 6, some of the least significant 48 bits of $ID_{tag(i)}$ can be derived as:

$$[ID_{tag(i)}]_{\mathcal{R}\cap\phi} = [K2^{(n+1)}_{tag(i)}]_{\mathcal{R}\cap\phi'} \oplus [K2^{(n)}_{tag(i)}]_{\mathcal{R}\cap\phi} \oplus [n2]_{\mathcal{R}\cap\phi} \quad (10)$$

$$[ID_{tag(i)}]_{\mathcal{R}\cap\tau} = [K4^{(n+1)}_{tag(i)}]_{\mathcal{R}\cap\tau'} \oplus [K4^{(n)}_{tag(i)}]_{\mathcal{R}\cap\tau} \oplus [n1]_{\mathcal{R}\cap\tau} \quad (11)$$

The number of bits which can be derived from the above equations is not deterministic due to the randomness of ϕ and τ. We now deduce the probability of deriving the bit information in above equations. Let $j \in \mathcal{R}$ be a bit position in all parameters. Under the randomness assumption of these parameters, we have $Pr\{j \in \phi\} = Pr\{j \in \tau\} = Pr\{j \in \phi'\} = Pr\{j \in \tau'\} = 1/2$. Also, we get $Pr\{j \in \phi \cap \phi'\} = Pr\{j \in \tau \cap \tau'\} = 1/4$. Therefore, from Equation (10), we can roughly derive $\sim$12 bits. Since $\phi \cap \tau = \emptyset$ and $\phi' \cap \tau' = \emptyset$, we can derive another $\sim$12 bits from Equation (11). Totally, we derived $\sim$24 less significant bits of $ID_{tag(i)}$ with 1 *full disclosure attack* on two continuous protocol runs.

Suppose we launch attack on l (not necessarily on continuous) protocol runs on the tag, we can roughly derive $48 \times (1 - 1/2^l)$ bits of $[ID_{tag(i)}]_{\mathcal{R}}$. We can fully disclose the ID of the tag in about 5 or 6 independent attacks against the protocol. In summary, we need l ($\approx \lceil \log_2 m - 1 \rceil$) attacks to fully disclose a tag's ID (with m-bit length). $\Box$

4 Discussions

4.1 Reasonable Threat Model

Threats in RFID Systems: RFID tags suffer from a variety of attacks: (1) *physical invasive attack*, where an adversary can physically compromise the inlay of an RFID tag and read the memory for any information; (2) *side channel attack*, where an adversary uses timing analysis, power analysis [14] or electromagnetic analysis to get tag information; (3) *jamming attack*, where an adversary blocks all RF channels between reader and tags; (4) *spoofing attack*, where a man-in-the-middle can impersonate a legitimate tag; (5) *eavesdropping*, where an attacker is able to intercept messages sent between reader and tags; (6) *cloning attack*, where an attacker writes the information of a compromised tag to a set of new (blank) tags.

Our threat model: Some of the earlier research work draw assumptions on practical limitations of RFID deployments: *i.e.,* firstly, the tag-to-reader channel is assumed to be private, since the backscatter channel from the tag to the reader has a relatively shorter range (*e.g.,* several centimeters) than that of the forward channel. Thus, an attacker, not within the range, cannot get reply from the tag. Secondly, it is not easy for an attacker to hide himself between a legitimate reader and a tag in an active session. Thirdly, it is not easy to intercept a message and modify the message over the air in real time, because of shared bearing medium. With these assumptions, active man-in-the-middle attacks such as our attacks described above are not possible. However the above assumptions are considered too restrictive. Some of the most research efforts [9, 3] assume the most reasonable or strongest threat model that an active man-in-the-middle can eavesdrop, intercept or modify messages in real time on both forward and backward channels. The EMAP protocol [15] is designed specifically to counter such active attacks.

4.2 Efficacy of *De-synchronization Attack*

Our de-synchronization attack flips some of the bits in protocol messages. To counter such "bit errors", one may employ error correcting codes. This can be effective in correcting random bit errors in transmission, but it does not help on defending against our (active) *de-synchronization attacks*, because our multi-bit attacks can change the updating pair ($n1$, $n2$) to an arbitrary random pair ($\hat{n1}$, $\hat{n2}$). The

Runs	Known secret values	Unknown secret values
n	$n1^{(n)}, n2^{(n)}, K1^{(n)}_{tag(i)}, K3^{(n)}_{tag(i)}, [K2^{(n)}_{tag(i)}]_{\phi}, [K4^{(n)}_{tag(i)}]_{\tau}$	$[K2^{(n)}_{tag(i)}]_{\tau}, [K4^{(n)}_{tag(i)}]_{\phi}, ID_{tag(i)}$
$n+1$	$n1^{(n+1)}, n2^{(n+1)}, K1^{(n+1)}_{tag(i)}, K3^{(n+1)}_{tag(i)}, [K2^{(n+1)}_{tag(i)}]_{\phi'}, [K4^{(n+1)}_{tag(i)}]_{\tau'}$	$[K2^{(n+1)}_{tag(i)}]_{\tau'}, [K4^{(n+1)}_{tag(i)}]_{\phi'}, ID_{tag(i)}$

Table 6. Known secret values in the nth and the $n+1$th protocol runs where ϕ' and τ' are the corresponding parameters associated with the $n+1$th run.

secrets stored at the reader can no longer be re-synchronized with that of the tag due to different updating pairs. As the attacker can launch this attack in any protocol run at will, a legitimate tag takes such a huge risk on conducting the protocol as being permanently destroyed unless being recalled and reassembled by its owner.

4.3 Severity of *Full Disclosure Attack*

This attack takes advantage of the tag's inability to remember protocol states. To counter our attack, it is necessary for a tag to store state information of a protocol run. To this end, we assign an additional status bit s, and set $s = 0$, if the protocol is completed (or synchronized) successfully; or $s = 1$, if the protocol is uncompleted (or asynchronized) due to some reason. The protocol status bit is set for the purpose of indicating the completion of a protocol execution. Only a successful completed protocol can trigger the updating operations at both the reader and the tag sides. As mentioned in the beginning of section 3, a completion message is very important for updating the secret values at both sides. That means an attacker can not learn the bit values of $n1$ or $n2$ within a single (incomplete) protocol by launching multiple (failed) trials.

In case of asynchronization, the tag will only expect a completion message from the reader, and will not reply to any other request. The reader, having sent a completion message, updates its database with the new secret values regarding the tag as well as stores the current updating pair $(n1, n2)$. If the completion message is not received or verified by the tag, the tag will not update its secret values. Later on, on receiving a former IDS of the tag, the reader needs to recalculate the completion message using $(n1, n2)$ and sends it to the tag for re-synchronization. If neither the former IDS is found, nor the completion message is valid, a tag is considered as being compromised permanently.

At the tag side, only 1 status bit is added and two additional random numbers $n1$, $n2$ (192 bits) are stored in EEPROM to remember the current status of the incomplete protocol. This mechanism increases a tag's memory size by $\sim 33\%$ $(= 193/(96 \times 6))$, while nearly all other hardware implementations for algorithm logic units or control units are not changed.

5 Related Works

Many research papers addressing RFID security and privacy problems have been published in the literature (please refer to [2, 10, 6] for a detailed literature survey). Our interest here is on RFID reader/tag (mutual) authentication protocols.

Authentication normally involves the use of secret data. Note that not all RFID tags are able to be authenticated since their inabilities of storing the secret data, *e.g.*, EPC class I tags. In the literature, one widely adopted assumption is using hash function within a tag. Weis *et al.* propose a randomized "hash lock" based mutual authentication protocol in [17]. Ohkubo *et al.* proposed a hash chain model embedding two hash functions in a tag [13]. Some solutions assume Pseudo-Random Function (PRF) in a tag. Molnar and Wagner use a tree scheme for authentication [11]. They further propose a scalable pseudonym protocol for ownership transfer [12]. Other assumptions includes using symmetric cipher, like [1, 5], in which Feldhofer *et al.* proposed a simple two way challenge-response mutual authentication protocol with AES encryption algorithm. The other work [7] even assume public key cryptographic primitive, in which tags update their IDs with re-encryption scheme.

To reduce the gate numbers in RFID tags, some approaches have been proposed without assumptions on classic cryptographic primitives. In [18], Weis introduced the concept of human computer authentication protocol due to Hopper and Blum, adaptable to low-cost RFID tags. Weis and Juels proposed a lightweight symmetric-key authentication protocol named HB$^+$ [9]. The security of both the HB and the HB$^+$ protocols is based on the Learning Parity with Noise (LPN) Problem, whose hardness over random instances still remains as an open question. In [16], the authors proposed a set of extremely-lightweight challenge-response authentication protocols, but their protocols can be broken by a powerful adversary. In [8], Juels proposed a solution based on pseudonyms without using hash functions at all. RFID tags store a short list of random identifiers or pseudonyms (known by authorized verifiers). When a tag is queried, it emits the next pseudonym in the list. However, the list of pseudonyms need be reused or updated via an out-of-band channel after a number of authentications. Due to those reasons, Peris-Lopez *et al.* proposed the efficient mutual authentication protocol for low-cost RFID tags: EMAP [15], in which only simple bitwise operations are used. This protocol is extremely lightweight and claimed to be secure against many attacks. However, we show the vulnerabilities of the protocol due to our *de-synchronization attack* and *full disclosure attack.*

6 Conclusions and Future Work

In this paper, we presented several effective attacks against EMAP, a ultra-lightweight RFID mutual authentication protocol recently proposed in [15]. The efficacy and severity of the attacks clearly indicate the insecure design of the protocol. It seems to us that it may be quite dangerous using only simple bitwise operations to achieve secure mutual authentication under powerful adversarial models without rigorous analysis. To counter those attacks, some potential countermeasures were also presented. Taken these attacks and countermeasures in mind, our next step aims to design secure ultra-lightweight mutual authentication protocols for low-cost (EPC) RFID tags.

References

[1] M. Aigner and M. Feldhofer. Secure Symmetric Authentication for RFID Tags. *Telecommunication and Mobile Computing*, March 2005.

[2] G. Avoine. Security and Privacy in RFID Systems. `http://lasecwww.epfl.ch/~gavoine/rfid/`

[3] J. Bringer, H. Chabanne, and E. Dottax. HB^{++}: a Lightweight Authentication Protocol Secure against Some Attacks. In: *Proc. of SecPerU'06*, pp. 28-33. IEEE Computer Society Press, 2006.

[4] EPCglobal, 13.56 MHz ISM band class 1 radio frequency (RF) identification tag interface specification.

[5] M. Feldhofer, M. Aigner, and S. Dominikus. An Application of RFID Tags using Secure Symmetric Authentication. In: *Proc. of SecPerU'05*, IEEE Computer Society Press, 2005.

[6] S. Garfinkel and B. Rosenberg. *RFID: Applications, Security, and Privacy*. Addison-Wesley Professional, 2005.

[7] A. Juels and R. Pappu. Squealing euros: Privacy protection in RFID-enabled banknotes. In: *Proc. of FC'03*, LNCS 2742, pp. 103-121. Springer-Verlag, 2003.

[8] A. Juels. Minimalist Cryptography for Low-Cost RFID Tags. In: *Proc. of SCN'04*, LNCS 3352, pp. 149-164. Springer-Verlag, 2004.

[9] A. Juels and S. Weis. Authenticating pervasive devices with human protocols. In: *Proc. of CRYPTO'05*, LNCS 3126, pp. 293-308. Springer-Verlag, 2005.

[10] A. Juels. RFID Security and Privacy: A Research Survey. *IEEE Journal on Selected Areas in Communications*, 24(2): 381-394, Feb. 2006.

[11] D. Molnar and D. Wagner. Privacy and Security in Library RFID: Issues, Practices, and Architectures. In: *Proc. of CCS'04*, pp. 210-219. ACM Press, 2004.

[12] D. Molnar, A. Soppera, and D. Wagner. A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags. In: *Proc. of SAC'05*, LNCS 3897, pp. 276-290. Springer-Verlag, 2005.

[13] M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic approach to privacy-friendly tags. In: *Proc. of RFID Privacy Workshop*, 2003.

[14] Y. Oren and A. Shamir. Power analysis attacks against RFID tags. Announced at the RSA Conference 2006. `http://www.wisdom.weizmann.ac.il/~yossio/rfid/`.

[15] 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. In: *OTM Federated Conferences and Workshop: IS Workshop*, November 2006.

[16] I. Vajda and L. Buttyan. Lightweight authentication protocols for low-cost RFID tags. In: *Proc. of UBICOMP'03*, 2003.

[17] S. Weis, S. Sarma, R. Rivest, and D. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In: *Proc. of 1st Int. Conf. on Security in Pervasive Computing*, LNCS 2802, pp. 201-212. Springer-Verlag, 2003.

[18] S. Weis. Security parallels between people and pervasive devices. In: *Proc. of PERSEC'05*, pp. 105-109. IEEE Computer Society Press, 2005.