Cryptanalysis of Some RFID Authentication Protocols

Tianjie Cao, Peng Shen
School of Computer, China University of Mining and Technology
Sanhuannanlu, Xuzhou, Jiangsu, 221116, China
National Mobile Communications Research Laboratory, Southeast University
Sipailou No.2, Nanjing, Jiangsu, 210096, China
Email: tjcao@cumt.edu.cn, pshencscumt@gmail.com

Elisa Bertino
Purdue University, West Lafayette, IN 47907
Email: bertino@cs.purdue.edu

***Abstract*—Two effective attacks, namely de-synchronization attack and impersonation attack, against Ha et al. 's LCSS RFID authentication protocol, Song and Mitchell's protocol are identified. The former attack can break the synchronization between the RFID reader and the tag in a single protocol run so that they can not authenticate each other in any following protocol run. The latter can impersonate a legal tag to spoof the RFID reader by extracting the ID of a specific tag during the authentication process. An impersonation attack against Chen et al.'s RFID authentication scheme is also identified. By sending malicious queries to the tag and collecting the response messages emitted by the tag, the attack allows an adversary to extract the secret information from the tag and further to impersonate the legal tag.**

***Index Terms*—RFID, de-synchronization attack, Impersonation**

I. INTRODUCTION

RFID (Radio Frequency Identification) technology is fast gaining popularity and attracting interest from both the industry and academic institutes. This technology has been applied to many applications such as object tracking and monitoring, supply-chain management [1].

An RFID system consists of three parts: RFID tags, an RFID reader, and back-end database. Security requirements for RFID authentication protocol include authentication, untraceability and availability.

Authentication: Authentication is the process whereby one party is assured of the identity of a second party involved in a protocol, and that the second has actually participated. Spoofing is an attack on authentication.

Untraceability: Untraceability is the most important security requirement for user privacy [2]. Untraceability is the property that adversary can not trace tag by using interactions with tag. This concept means ID anonymity and indistinguishability.

Availability: Authentication should be available all the time between reader and tags. Authentication protocol should provide the data recovery against the data loss or falsification such as DoS, message hijacking, power interruption, etc. during the authentication processes. Especially, the de-synchronization attack by utilizing a man-in-the-middle attack must be prevented.

In [3], Rhee et al. proposed a challenge-response authentication protocol based on a hash function. However, the computational load on the back-end database is heavy when authenticating a tag. Another disadvantage of Rhee et al.'s protocols is that the protocols do not satisfy forward security. The RFID mutual authentication scheme presented by Lee et al. [4] introduces forward security based on synchronized secret information. However, Lee et al.'s protocol also requires many computational operations in the back-end database when finding a specific tag's ID. In [5], Ha et al. proposed a low-cost and strong-security (LCSS) authentication protocol for an RFID system. The main contribution of Ha et al.'s protocol is low computation in back-end database. In the case of de-synchronization between the back-end database and a tag, the protocol is able to recover the synchronization. As the correct ID can be found based on just comparing the transmitted hash message and the hashed values in the database, the computational load on the back-end system is efficient compared with Rhee and Lee et al.'s protocols.

Numerous authentication protocols for RFID systems were proposed in an attempt to provide privacy and security. Many of these attempts fail to enforce anonymity and offer only weak authentication and some fail under denial of service [6][7]. To secure RFID systems, various lightweight RFID schemes have been designed, where mostly hash functions and random number generators are involved. In [8], Song and Mitchell proposed a scheme that significantly reduces the necessary storage and computation in a tag by comparison with previous hash-based schemes. The ultra-lightweight schemes only involve simple bit-wise operations on tags [9-10]. However, de-synchronization attack, full-disclosure attack and tracing attack against

such schemes have been reported [11-12]. In [13], Chou et al. presented a simple scheme based on quadratic residue assumption. This scheme is much cheaper than the implementation of a hash function. Hsiang demonstrated that Chou et al.'s scheme is vulnerable to the masquerading attack and the parallel session attack [14]. Recently, Chen et al. proposed a new efficient scheme based on quadratic residues and claimed that the new scheme not only achieves the mutual authentication between the server and the tag but also can satisfy all the security requirements needed in an RFID system [15].

In this paper, we analyze the security vulnerabilities of the LCSS protocol [5] and the Song-Mitchell protocol [8]. In [5] and [8], the authors presented some security analysis and claimed that their protocol is secure against de-synchronization attack and spoofing attack. In the following sections, we will show that the claims unfortunately don't hold. We also identify an impersonation attack against Chen et al.'s RFID authentication protocol [15]. By sending malicious queries to the tag and collecting the response messages emitted by the tag, our attack allows an adversary to extract the secret information from the tag and further to impersonate the legal tag.

II. CRYPTANALYSIS OF THE LCSS PROTOCOL

A. Review of the LCSS protocol

Fig. 1 shows the process of the proposed protocol, and the following is a detailed description of each step:

1. The *R* broadcasts to the tags with a *Query* and a random number R_R as a challenge.

2. The *T* generates a random number R_T and computes *P* differently according to the state of *SYNC*. If *SYNC* is 0, then $P=H(ID)$, otherwise $P=H(ID||R_T||R_R)$. And then sets *SYNC* as 1. The *T* transmits *P* and R_T to the *R* as a response to *Query*.

3. The *R* forwards the *P* and R_T message received from the *T* together with R_R generated by itself in step 1 to the *DB*.

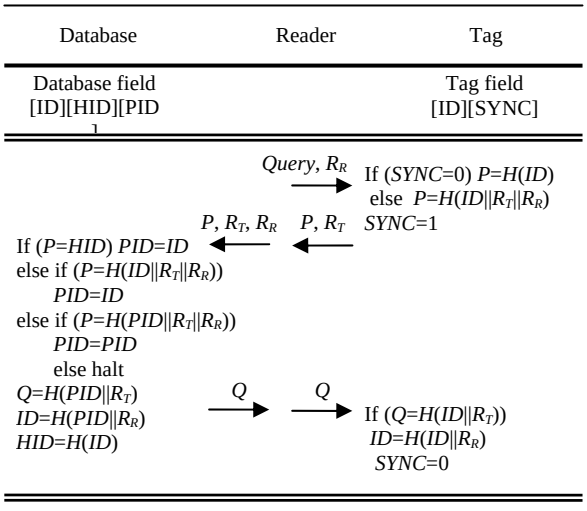

Figure 1 The LCSS protocol

4. As soon as the *DB* receives the message from the R, it searches for the specific tag via the received *P*. Firstly, the *DB* finds whether there is a record's *HID* value equivalent to the received *P*. If so, the *DB* regards the record's *ID* value as the identity of the *T*, which is requesting authentication. This is the general case when the previous session is terminated normally.

When the *DB* cannot find any record's *HID* value equivalent to *P* in the first search, the value of $H(ID||R_T||R_R)$ will be computed for all the *ID* in the database, with which compares the *P*.

However, if the *DB* cannot find the exact *ID* of the tag in the two above cases, it will compute the value of $H(PID||R_T||R_R)$ for all the *PID* in the database then compares it with the *P*.

If the *DB* is still unable to find the tag's *ID* in all the three above cases, it halts the search for the *ID* and orders the *R* to query again.

The *R* will be authenticated successfully as a legal one, as soon as the *DB* does find the *ID* or *PID* in one of the three searching cases. Then the *DB* updates ID with the value of $H(PID||R_R)$ and computes $HI = H(ID)$ for the next session. Finally, computes $Q= H(PID||R_T)$ and transmits it to the *R*.

5. The *R* forwards the message *Q* to the *T*.

6. The *T* verifies the correctness of *Q* by checking whether it is equivalent to the value of $H(ID||R_T)$. If so, the *T* updates its *ID* with the value of $H(ID||R_R)$, then sets the *SYNC* state as 0.

B. Desynchronization Attack

Fig. 2 depicts the message transmission of the de-synchronization attack. The detailed steps are as follows:

1. The attacker can eavesdrop in the insecure channel. When the reader *R* broadcasts a *Query* with a random number R_R to the tags, the attacker can obtain R_R, changes it to R_A. Then, sends the *Query* with R_A to the targeted tag.

2. We can suppose that the system is working normally now, that is, the *SYNC* state of the targeted tag *T* is 0, the *T* computes $P=H(ID)$ and transmits it to the *R* with random number R_T.

3. The *R* forwards the message received from the *T* to the *DB* together with the random number R_R generated by itself.

4. The *DB* receives the $P=H(ID)$, it does find a record's *HID* equivalent to *P*, then assigns the value of *ID* to *PID*. After performing these computations $Q=H(PID||R_T)=H(ID||R_T)$, $ID=H(PID||R_R)=H(ID||R_R)$ and $PID=H(ID)$, the *DB* transmits the *Q* to the *R*.

5. The *R* directly forwards the *Q* received from the *DB* to the *T*.

6. As the *T* receives the *Q* equivalent to the value of $H(ID||R_T)$, the *ID* in the tag will be updated with the value of $H(ID||R_A)$, *SYNC* state changes into 0. We can note that the *ID*'s value in the *T* is not equivalent to that in the *DB*. It seems like that the system has successfully completed a protocol run, actually, it is already trapped into de-synchronization permanently.

Now the value of *ID* in the *DB* is $ID_{DB}=H(ID||R_R)$, the value of *ID* in the *T* is another value $ID_T= H(ID||R_A)$. When the *T* transmits $P=H(ID_T)$ to the *R*, which will be

forwards to the *DB*. After the *DB* searching all the database records in all the three cases, it will not be able find a proper match for the *P* received from the *R*. Consequently, the authentication of the legal tag, in which the *ID* is updated under the above presented de-synchronization attack, will be halted. In the next protocol run, the tag will respond with $P=H(ID_T||R_T||R_R)$ and R_T. *DB* will still not be able find a proper match for the *P* in all the three cases.

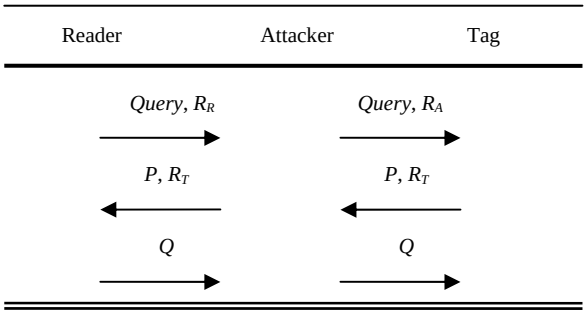

Figure 2 De-synchronization attack

C. Impersonation attack

The detailed attack includes two stages:

Stage 1. Supposing the system is working normally right now. An adversary sends a malicious query to a targeted tag with a *Query* and a random number R_A, then collects the response messages $P=H(ID)$ and R_T emitted by the tag.

Stage 2 is described as follows:

1. The attacker eavesdrops in the insecure channel, collecting the broadcasting *Query* message. Obviously, the random number R_R generated by the *R* can be obtained by the attacker. Therefore, the attacker is capable to impersonate the tag *T* transmit the message including $P=H(ID)$ and the random number R_R as a response to the *R*.

2. The *R* forwards the message containing $P=H(ID)$ and R_R to the *DB* together with the random number R_R generated by itself. Of course, the two random numbers are same.

3. The *DB* searches a record in the database to match the $P=H(ID)$ received from the *R*. And then updates *PID* with the value of *ID*. After performing these computations $Q=H(PID||R_R)=H(ID||R_R)$, $ID=H(PID||R_R)=H(ID||R_R)$ and $PID=H(ID)$, the *DB* transmits the *Q* to the *R*. We note that the new *ID* and the *Q* are equivalent to each other.

4. The *R* directly forwards the *Q* received from the *DB* to the *T*. So the attacker is capable to obtain the message *Q*, namely the new *ID* of the *T*.

From what mentioned above, we can see that the attacker is able to own the new *ID* in the *T*. So the attacker is capable to disguise as a legitimate tag to spoof the *R* and update the *ID* during the next session. Fig. 3 depicts the message transmission of the spoofing attack.

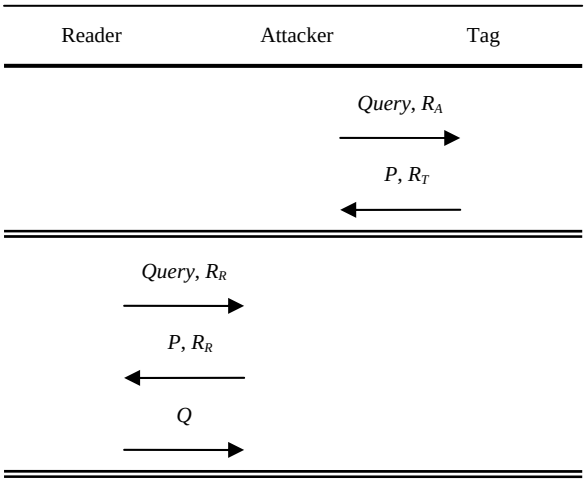

Figure 3 Impersonation attack

In the case of the de-synchronization attack, where random number R_R in protocol step 1 is replaced with R_A due to an malicious attacker, the database can not detect this attack in protocol step 4 because of lack integrity check on R_R. In the case of the spoofing attack, where random number R_T in protocol step 2 is replaced with R_R, the database can not detect this attack in protocol step 4 because of lack authentication on the tag in the case of *SYNC*=0.

We can add message authentication code $M=H(R_T||ID||R_R)$ in protocol step 2 to protect the system from de-synchronization attack and spoofing attack. If the attacker replaces R_T with R_A in step 2, the *DB* will detect this attack by check the validity of the value *M*. If the attacker replaces R_T with R_R in step 2, the attacker will unable to generate $M= H(R_R||ID||R_R)$ to respond to *Query*. Illustration of the message transmission of the improved protocol is depicted as Fig. 4.

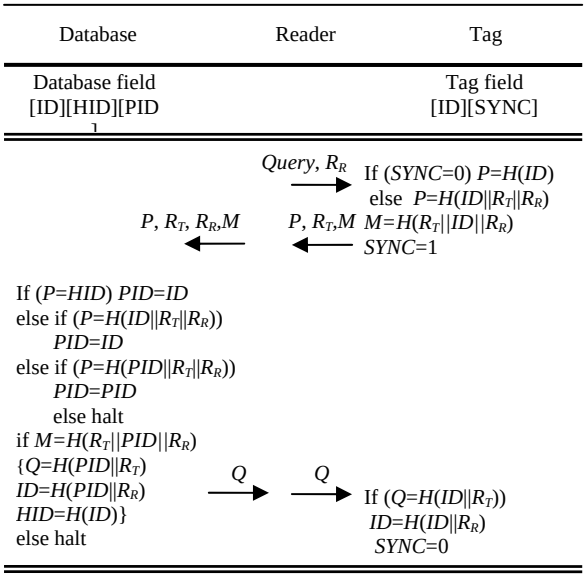

Figure 4 The improved LCSS protocol

III. CRYPTANALYSIS OF SONG AND MITCHELL'S PROTOCOL

A. Review of Song and Mitchell's Protocol

We use the following notation in Song and Mitchell's authentication scheme.

h: A hash function, h: $\{0, 1\}^l \rightarrow \{0, 1\}^l$.
f_k: A keyed hash function, f_k: $\{0, 1\}^l \times \{0, 1\}^l \rightarrow \{0, 1\}^l$
N: The number of tags
l: The bit-length of a tag identifier
T_i: The i-th tag ($1 \le i \le N$)
D_i: The detailed information associated with tag T_i
u_i: A string of l bits assigned to T_i
t_i: T_i's identifier of l bits, which equals $h(u_i)$
x_{new}: The new (refreshed) value of x
x_{old}: The most recent value of x
r: A random string of l bits
ε: Error message
$\oplus$: XOR operator
$||$:Concatenation operator
$\leftarrow$: Substitution operator
$x>>k$: Right circular shift operator, which rotates all bits of x to the right by k bits.
$x<<k$: Left circular shift operator, which rotates all bits of x to the left by k bits.
$\in_R$: The random choice operator.

An initiator (e.g. the tag manufacturer) assigns a string u_i of l bits to each tag T_i, computes $t_i = h(u_i)$, and stores t_i in the tag. The initiator stores the entries $[(u_i, t_i)_{new}, (u_i, t_i)_{old}, D_i]$ for every tag that it manages. The first pair is for the newly assigned values of u_i and t_i, the second pair is for the previously assigned values, and D_i is for the tag information (e.g., price, date, etc.). Initially $(u_i, t_i)_{new}$ is assigned the initial values of u_i and t_i, and $(u_i, t_i)_{old}$ is set to null.

Song and Mitchell's authentication scheme is summarized in Fig. 5.

1. A reader generates a random bit-string $r_1 \in_R \{0,1\}^l$ and sends it to T_i.
2. The tag T_i generates a random bit-string $r_2 \in_R \{0,1\}^l$, and computes $M_1=t_i \oplus r_2$ and $M_2=f_{t_i}(r_1 \oplus r_2)$. T_i then sends M_1 and M_2 to the reader.
3. The reader transmits M_1, M_2 and r_1 to the server.
4. The server chooses t_i from amongst the values $t_{i(new)}$ or $t_{i(old)}$ stored in the database and puts $r_2 \leftarrow M_1 \oplus t_i$. If $M_2=f_{t_i}(r_1 \oplus r_2)$, then the server has identified T_i. Otherwise, it chooses another t_i. If no match is found, the server sends ε and stops the session. The server computes $M_3=u_i \oplus (r_2>>l/2)$ and sends it with D_i to the reader. At last, the server updates $u_{i(old)}$ and $t_{i(old)}$ to u_i and t_i, and sets $u_{i(new)} \leftarrow (u_i<<l/4) \oplus (t_i>>l/4) \oplus r_1 \oplus r_2$ and $t_{i(new)} \leftarrow h(u_{i(new)})$.
5. The reader forwards M_3 to T_i.
6. T_i computes $u_i \leftarrow M_3 \oplus (r_2>>l/2)$ and checks that $h(u_i)= t_i$. If the check succeeds, the tag has authenticated the server, and sets $t_i \leftarrow h((u_i<<l/4) \oplus (t_i>>l/4) \oplus r_1 \oplus r_2)$. If the check fails, the tag keeps the current value of t_i unchanged.

We note $M_3 \oplus (M_1>>l/2)= u_i \oplus (t_i>>l/2)$.

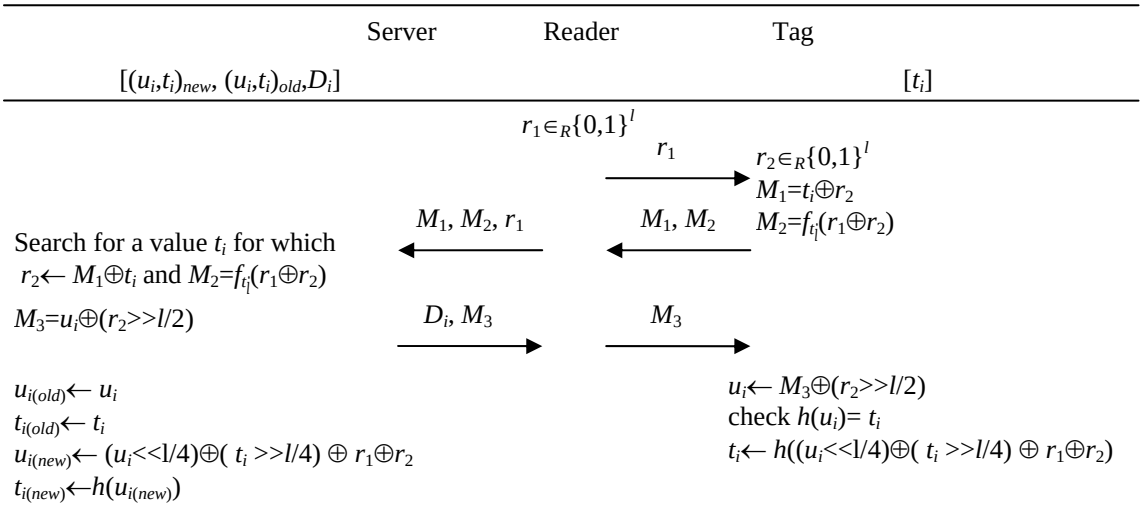

Figure 5 Song and Mitchell's Protocol

B. Impersonation Attack

In [8], the authors claimed that their scheme is robust to tag impersonation attack based on the idea that the adversary cannot compute a valid response (M_1, M_2) without knowledge of t_i. However, if the adversary attacks the system by the following way, the protocol will show its vulnerability to the impersonation attack although the adversary does not know t_i. The detailed attack includes two stages:

Collection stage: Supposing the system is working normally right now. An adversary sends a malicious query to a targeted tag with a random number r_1, then collects the response messages M_1 and M_2 emitted by the tag, where $r_2 \in_R \{0,1\}^l$, $M_1=t_i \oplus r_2$ and $M_2=f_{t_i}(r_1 \oplus r_2)$. In the impersonation stage, the adversary will replay M_2.

Impersonation stage: After collecting the response messages M_1 and M_2, the adversary can modify the data, and then replay the messages to masquerade as the legal tag.

1. A reader generates a random bit-string $r_1' \in_R \{0,1\}^l$ and broadcasts it. The adversary eavesdrops in the insecure channel, collecting the broadcasting message. Obviously, the random number r_1' can be obtained by the adversary.

2. The adversary computes $M_1' = M_1 \oplus r_1' \oplus r_1$, $M_2' = M_2$ and impersonate the tag transmit the message M_1', M_2' as a response to the reader.

3. The reader transmits M_1', M_2' and r_1' to the server.

4. The server chooses t_i from amongst the values $t_{i(new)}$ or $t_{i(old)}$ stored in the database and recovers r_2' from M_1' using $r_2' \leftarrow M_1' \oplus t_i$. If $M_2' = f_{t_i}(r_1' \oplus r_2')$, then the server has identified T_i. Otherwise, it chooses another t_i. The server computes $M_3' = u_i \oplus (r_2' >> l/2)$ and sends it with D_i' to the reader. At last, the server updates $u_{i(old)}$ and $t_{i(old)}$ to u_i and t_i, and sets $u_{i(new)} \leftarrow (u_i << l/4) \oplus (t_i >> l/4) \oplus r_1 \oplus r_2$ and $t_{i(new)} \leftarrow h(u_{i(new)})$.

5. The reader forwards M_3' to the adversary.

In the step 4, we prove that the server will authenticate the tag. We have

$f_{t_i}(r_1' \oplus r_2') = f_{t_i}(M_1' \oplus M_1 \oplus r_1 \oplus r_2') = f_{t_i}(M_1' \oplus t_i \oplus r_2 \oplus r_1 \oplus M_1' \oplus t_i) = f_{t_i}(r_1 \oplus r_2) = M_2 = M_2'$

Fig. 6 depicts the impersonation attack.

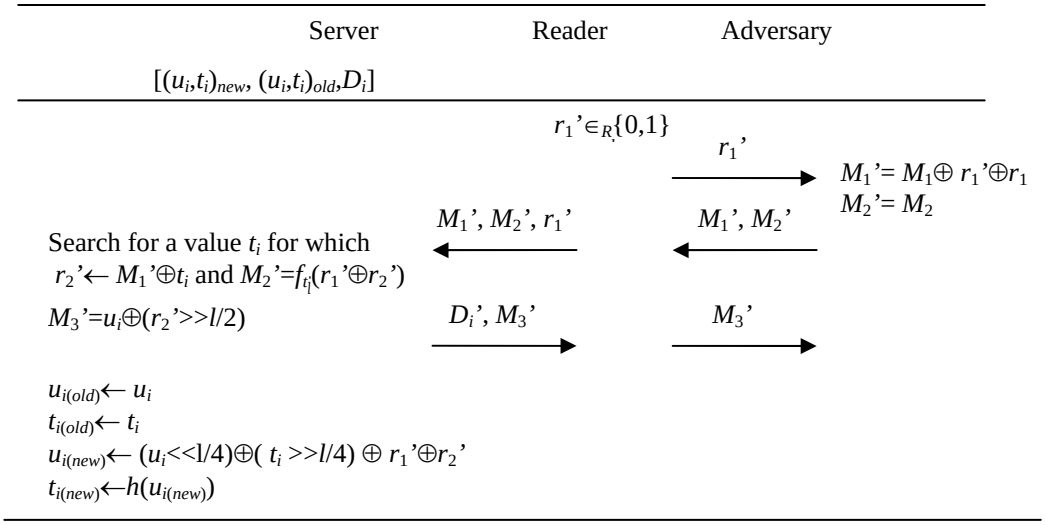

Figure 6 Impersonation attack

C. De-synchronization Attack

To provide privacy protection, most RFID authentication schemes update tag's secret information after a successful protocol run. This update is performed in the back-end database as well as in the tag. So synchronization of secret information between the database and the tag is crucial for subsequent authentications. Some kinds of protocol malfunctions might leave the both sides in a un-synchronization 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. Fig. 7 depicts the message transmission of the de-synchronization attack.

The detailed steps are as follows:

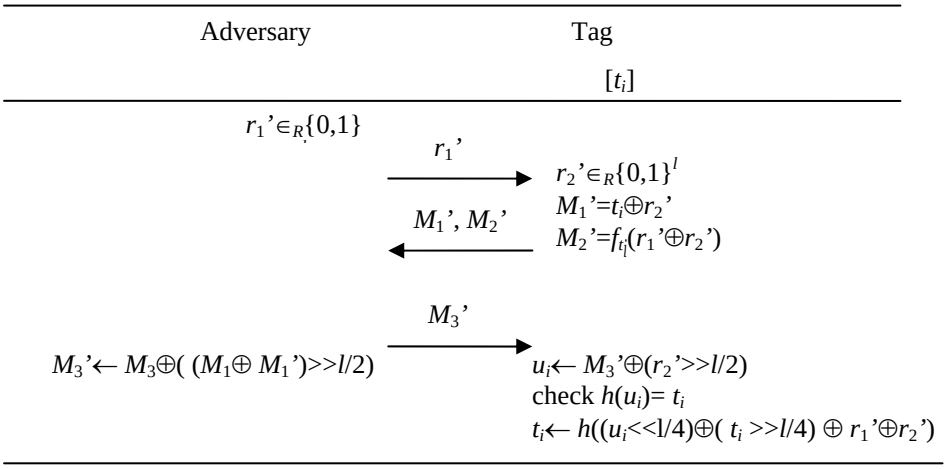

Figure 7 De-synchronization attack

1. Supposing the system is working normally right now. The server stores the entries $[(u_i, t_i)_{new}, (u_i, t_i)_{old}, D_i]$ for every tag and the tag stores the value of t_i. We have $t_i=t_{i(new)}=h(u_{i(new)})$. Based the tag impersonation attack described in section 2.2, the adversary first impersonates the tag to spoof the server and record M_1 and M_3. We note $M_3\oplus(M_1>>l/2)= u_{i(new)}\oplus(t_{i(new)} >>l/2)$. After authenticating the adversary, the server updates its secrets.

2. Now the adversary disguises as a legitimate reader to spoof the tag and update the tag's secret. The adversary generates a random bit-string $r_1' \in_R \{0,1\}^l$ and sends it to T_i.

3. The tag T_i generates a random bit-string $r_2' \in_R \{0,1\}^l$, and computes $M_1'=t_i\oplus r_2'$ and $M_2'=f_{t_i}(r_1'\oplus r_2')$. T_i then sends M_1' and M_2' to the reader.

4. The adversary sends $M_3' \leftarrow M_3\oplus((M_1\oplus M_1')>>l/2)$ to the tag.

5. T_i computes $u_i \leftarrow M_3'\oplus(r_2'>>l/2)$ and checks that $h(u_i)= t_i$. If the check succeeds, the tag has authenticated the server, and sets $t_i \leftarrow h((u_i<<l/4)\oplus(t_i >>l/4) \oplus r_1'\oplus r_2')$.

We prove that in step 5 the tag will accept M_3'.

$u_i= M_3'\oplus(r_2'>>l/2)$

$= M_3\oplus((M_1\oplus M_1')>>l/2) \oplus(r_2'>>l/2)$

$= M_3\oplus(M_1>>l/2) \oplus (M_1'>>l/2) \oplus(r_2'>>l/2)$

$=u_{i(new)}\oplus(t_{i(new)}>>l/2)\oplus(t_{i(new)}\oplus r_2'>>l/2)\oplus(r_2'>>l/2)$

$= u_{i(new)}$

After the tag updates its secret information t_i using two random bit-strings r_1' and r_2', the RFID system will be involved in DoS state and can not provide availability.

IV. Cryptanalysis of Chen et al.'s Protocol

A. Review of Chen et al.'s Protocol

There are two phases in Chen et al's scheme: an initialization phase and an authentication phase.

In the initialization phase, the server generates two large primes p and q, and computes $n = pq$. It also chooses a one-way hash function, $h()$, and a pseudo-random number generator, $PRNG()$. The value of n and $h()$, $PRNG()$ are both made public. The server chooses a random number $r \in Z_n$ and writes TID, $h(TID)$ and r into tag's memory, where TID may include EPC codes depending on the user's specification. Meanwhile, the server saves $<h(TID), TID, r, r_{old}>$ into its database, where $r_{old} = r$ at the beginning.

The authentication phase of Chen et al's scheme is described as follows. It is also illustrated in Fig. 8.

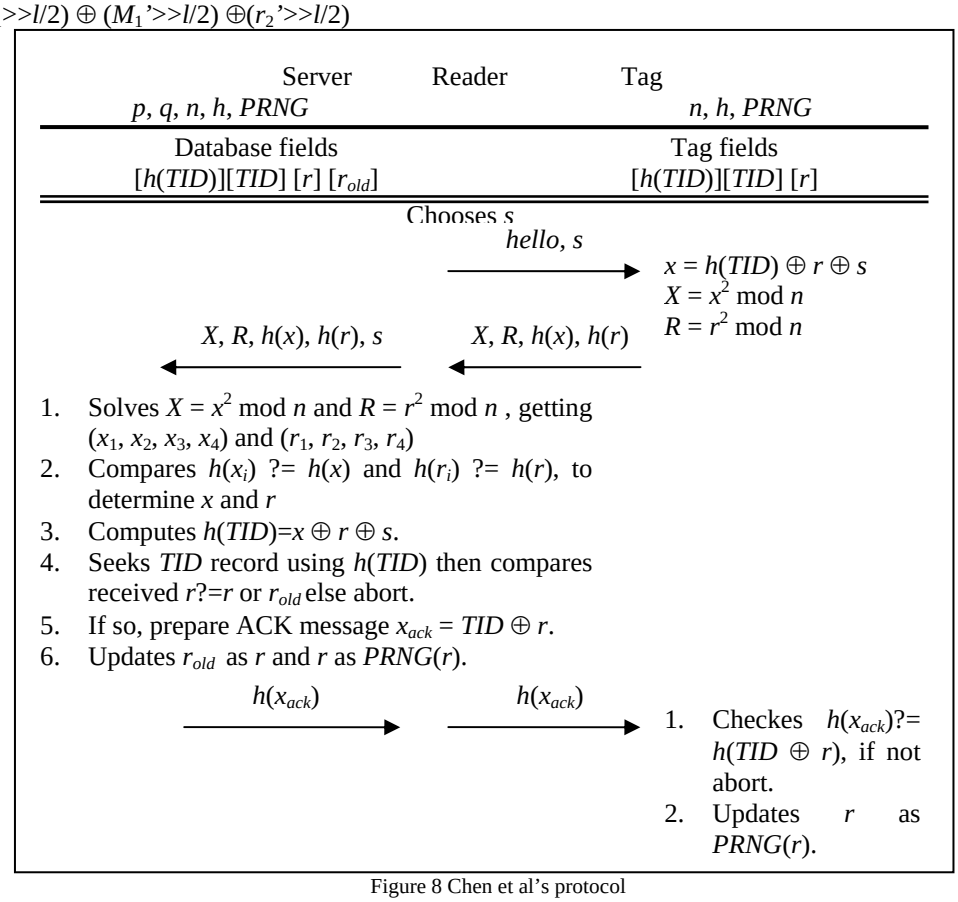

Figure 8 Chen et al's protocol

Step 1. The reader chooses a random challenge $s \in Z_n$ and broadcasts a *hello* message together with s to the tag.

Step 2. After receiving the *hello* message and challenge s, the tag reads TID, $h(TID)$ and r from its memory and computes $x = h(TID) \oplus r \oplus s$, $X = x^2 \bmod n$, and $R = r^2 \bmod n$. The tag responses $<X, R, h(x), h(r)>$ to the reader.

Step 3. After receiving tag's response $<X, R, h(x), h(r)>$, the reader forwards this response together with s to the server.

Step 4. After receiving <X, R, $h(x)$, $h(r)$, s>, the server solves $X = x^2 \bmod n$ and $R = r^2 \bmod n$ by using Chinese Remainder Theorem, obtaining four roots (x_1, x_2, x_3, x_4) and (r_1, r_2, r_3, r_4) respectively. It then compares $h(x_i)$ with $h(x)$ and $h(r_i)$ with $h(r)$, for i = 1 to 4, to determine the unique values of x and r. The server then computes $x \oplus r \oplus s$, obtaining $h(TID)$. Having obtained $h(TID)$, the server uses it as a searching key to find the tag record in its database. If it is not found, the server will abort the session; otherwise, it verifies whether the solved r is equal to the value of r or r_{old} stored in the found record. If it is, the server will compute $x_{ack} = TID \oplus r$ and then sends the acknowledgement message $h(x_{ack})$ to the tag through the reader. Simultaneously, the server updates the tag's record by replacing r_{old} with r, and r with $PRNG(r)$.

Step 5. After receiving the server/reader's $h(x_{ack})$, the tag verifies whether $h(TID \oplus r)$ is equal to the received $h(x_{ack})$. If so, the tag updates r with $PRNG(r)$.

B. Impersonation Attack

An important observation of Chen et al.'s scheme is that if the adversary could compute the secret value $h(TID) \oplus r$ then the adversary could impersonate the legal tag. By utilizing responses from a tag, an adversary may try to get knowledge of the tag. Fig. 9 depicts the message transmission of the impersonation attack.

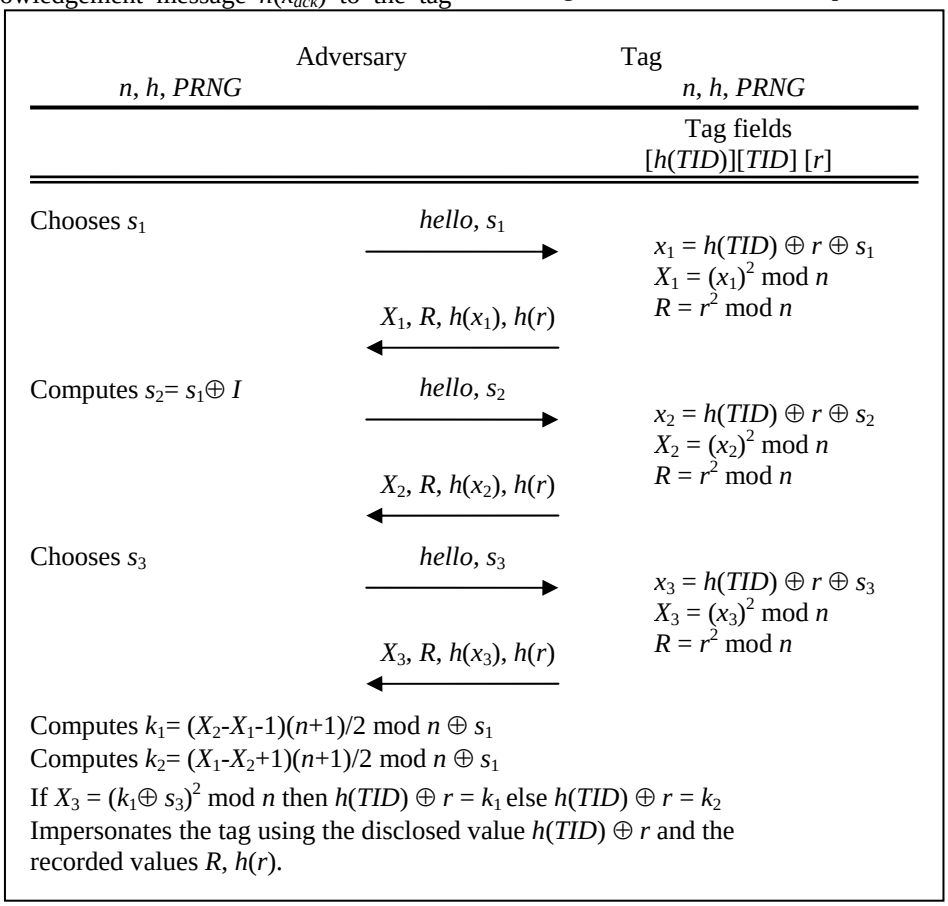

Figure 9 Impersonation attack

The detail impersonation attack is described as follows.

Step 1. Supposing the system is working normally right now. The adversary chooses a random challenge $s_1 \in Z_n$ and sends a *hello* message together with s_1 to the tag. The tag responds the message < X_1, R, $h(x_1)$, $h(r)$>. The adversary records this message. We have:

$$x_1 = h(TID) \oplus r \oplus s_1 \quad (1)$$

$$X_1 = (x_1)^2 \bmod n \quad (2)$$

Step 2. The adversary computes $s_2 = s_1 \oplus I$, where I=[000…001] (set the least significant bit as 1). The adversary sends a *hello* message together with s_2 to the tag. The tag responds the message < X_2, R, $h(x_2)$, $h(r)$>. The adversary records this message. We have:

$$x_2 = h(TID) \oplus r \oplus s_2 \quad (3)$$

$$X_2 = (x_2)^2 \bmod n \quad (4)$$

Step 3. The adversary sends a random number s_3, and records the responding message < X_3, R, $h(x_3)$, $h(r)$>. We have

$$X_3 = (h(TID) \oplus r \oplus s_3)^2 \bmod n \quad (5)$$

Step 4. After obtaining X_1 and X_2, the adversary can recover two candidate secrets of $h(TID) \oplus r$, and then can check the validity through the equation (5).

From equation (1), (2), (3) and (4), we have

$X_2 = (x_2)^2 \bmod n = (h(TID) \oplus r \oplus s_2)^2 \bmod n = (h(TID) \oplus r \oplus s_1 \oplus I)^2 \bmod n = (x_1 \oplus I)^2 \bmod n$

If the least significant bit of x_1 is 0 then $X_2 = (x_1+1)^2 \bmod n = X_1 + 2x_1 + 1 \bmod n$. We can obtain $x_1 = (X_2 - X_1 - 1)(n+1)/2 \bmod n$. Let k_1 denote the first candidate of $h(TID) \oplus r$, we have $k_1 = (X_2 - X_1 - 1)(n+1)/2 \bmod n \oplus s_1$.

If the least significant bit of x_1 is 1 then $X_2 = (x_1-1)^2 \bmod n = X_1 - 2x_1 + 1 \bmod n$. We can obtain $x_1 = (X_1$-

X_2+1)(n+1)/2 mod n. Let k_2 denote the second candidate of $h(TID)\oplus r$, we have k_2= (X_1-X_2+1)(n+1)/2 mod $n \oplus s_1$.

We denote the value of $h(TID)\oplus r$ by k. Now we determine which one of two candidates is the value of $h(TID)\oplus r$ through the equation (5). If $X_3 = (k_1 \oplus s_3)^2$ mod n then $k = k_1$ else $k = k_2$.

Step 5. Once the adversary obtains the value of $h(TID) \oplus r$, he can impersonate the tag using $h(TID) \oplus r$ and the recorded values R, $h(r)$. When the reader chooses a random challenge $s \in Z_n$ and broadcasts a *hello* message together with s. The adversary computes $x = k \oplus s$, $X = x^2$ mod n. The adversary responses to the reader with <X, R, $h(x)$, $h(r)$> and will be authenticated by the reader.

V. Conclusions

In this paper, we have identified two effective attacks, namely impersonation attack and de-synchronization attack, against the LCSS protocol and the Song-Mitchell RFID authentication protocol. We also have identified an impersonation attack against Chen et al.'s RFID authentication scheme. These attacks should be considered in the designing the new RFID authentication protocol.

Acknowledgment

This work is supported by the Jiangsu Provincial Natural Science Foundation of China (BK2007035), the open research fund of National Mobile Communications Research Laboratory, Southeast University (W200817) and the Science and Technology Foundation of CUMT (0D080309).

References

[1] D. Lin, H. G. Elmongui, E. Bertino, and B. C. Ooi, "Data Management in RFID Applications", *International Conference on Database and Expert Systems Applications*, LNCS 4653, pp. 434-444, 2007.

[2] G. Avoine. "Radio frequency identification: adversary model and attacks on existing protocols", *Technical Report LASEC-REPORT-2005-001*, EPFL, Lausanne, Switzerland, September 2005.

[3] K. Rhee, J. Kwak, S. Kim and D. Won, "Challenge-Response Based on RFID Authentication Protocol for Distributed Database Environment", *SPC 2005*, LNCS 3450, pp. 70-84, 2005.

[4] S. Lee, T. Asano and K. Kim, "RFID Mutual Authentication Scheme based on Synchronized Secret Information", *SCIS'06*, 2006.

[5] J.C. Ha, S.J. Moon, J. M. G. Nieto and C. Boyd, "Low-Cost and Strong-Security RFID Authentication Protocol", *EUC Workshops 2007*, LNCS 4809, pp. 795-807, 2007.

[6] H. Lei, T.J. Cao, "RFID Protocol enabling Ownership Transfer to protect against Traceability and DoS attacks", *International Symposium on Data, Privacy, & E-Commerce*, pp. 508-510, 2007.

[7] H. Lei, T.J. Cao, "Cryptanalysis of SPA Protocol", *Security and Privacy in Telecommunications and Information System*, Shanghai China, December 16-19, 2007.

[8] B. Song, C. Mitchell, "RFID authentication protocol for low-cost tags", "First ACM Conference on Wireless Security", *WiSec 2008*, pp.40-147, 2008.

[9] P. Peris-Lopez, J. Hernandez-Castro, J. Estevez-Tapiador, and A. Ribagorda, "LMAP: A Real Lightweight Mutual Authentication Protocol for Low-Cost RFID Tags", *Second Workshop RFID Security*, July 2006.

[10] H. -Y. Chien, "SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity", *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 4, 2007, pp. 337-340.

[11] T. Li, G. Wang, "Security Analysis of Two Ultra-Lightweight RFID Authentication Protocols", *22nd IFIP TC-11 Int'l Information Security Conf.*, May 2007.

[12] T. Cao, E. Bertino, H. Lei, "Security Analysis of the SASI Protocol", *IEEE Transactions on Dependable and Secure Computing*, 20 May 2008. IEEE Computer Society Digital Library. IEEE Computer Society, 26 May 2008 http://doi.ieeecomputersociety.org/10.1109/TDSC.2008.32

[13] J.-S. Chou, G.-C. Lee, C.-J. Chan, "A novel mutual authentication scheme based on quadratic residues", *Cryptology ePrint Archive*, http://eprint.iacr.org/2007/224

[14] H.-C. Hsiang, "Weaknesses of a Novel Mutual Authentication Scheme Based on Quadratic Residues for RFID Systems", *2008 RFID workshop*, Feb. 2008

[15] Y. Chen, "A novel mutual authentication scheme based on quadratic residues", *Computer Network*, Vol. 52, no. 12, pp. 2373-2380, August 2008.

Tianjie Cao is a professor of China University of Mining and Technology. His research interests are in security protocols and network security.

Peng Shen is currently working toward the Master degree in the School of Computer Science and Technology, China University of Mining and Technology.

Elisa Bertino is a professor of Purdue. Her research interests are in information security.