International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **7(A)**, July **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 4431–4449

NEW FINDINGS ON RFID AUTHENTICATION SCHEMES AGAINST DE-SYNCHRONIZATION ATTACK

Kuo-Hui Yeh[1], Nai-Wei Lo[2], Yingjiu Li[3], Yung-Chun Chen[2] and Tzong-Chen Wu[2]

[1]Department of Information Management
National Dong Hwa University
No. 1, Sec. 2, Da Hsueh Rd., Shoufeng, Hualien 97401, Taiwan
khyeh@mail.ndhu.edu.tw

[2]Department of Information Management
National Taiwan University of Science and Technology
No. 43, Sec. 4, Keelung Rd., Taipei 106, Taiwan
{ nwlo; tcwu }@cs.ntust.edu.tw; D9409103@mail.ntust.edu.tw

[3]School of Information Systems
Singapore Management University
80 Stamford Road, Singapore 178902
yjli@smu.edu.sg

Received March 2011; revised July 2011

Abstract. *In order to protect privacy of RFID tag against malicious tag tracing activities, most RFID authentication protocols support forward/backward security properties by updating the same secret values held at both tag end and database end asynchronously during each authentication session. However, in real network environments an adversary may easily interrupt or interfere transmission of necessary key update message in each authentication session such that key re-synchronization between tag and database cannot be completed, which is named as de-synchronization attack. To defend against this security threat, recent RFID authentication schemes have applied redundant secret/key design to allow a tag with de-synchronized secret to successfully communicate with server/database in its next authentication session. In this paper, we first categorize existing authentication protocols into three types based on their key update mechanisms. Then security evaluation on de-synchronization attack is conducted for type I and II protocols. Two attack models and theorems show that synchronization mechanisms used in type I and II schemes cannot defend against de-synchronization attack. Finally, three remarks are further presented to support our important finding: most existing RFID authentication schemes cannot simultaneously provide forward/backward security and resistance for de-synchronization attack in practical setting.*

Keywords: De-synchronization attack, RFID authentication, Tag identification, Security

1. **Introduction.** RFID technology is massively adopted in various applications [36,57-60] to identify each target object on which a tag with RF (radio frequency) antenna is attached. An RFID application system is composed of a backend server/database, one or multiple readers and a lot of tagged objects. In order to prevent illegal and malicious access to information contained in an RFID tag, many theoretically-secure RFID authentication protocols [1,4-6,8-11,13,16,17,23,25-28,31-37,39,41-45,48-51,53-57] are proposed in recent years; even some of them [36] can be implemented in real world. As personal privacy has become a highly sensitive topic around the world, there exists real demand for RFID authentication scheme to support forward/backward security on tagged

objects to avoid an adversary tracing a person by tracking RFID tagged objects he carries or wears. Existing authentication protocols usually achieve forward/backward security by dynamically updating the secret key held at both tag end and the server end. However, it is easy for an adversary to destroy a key update process between tag and server by interfering communication channels. This security threat is named as de-synchronization attack. To defend against this attack, recently proposed protocols utilize the concept of key redundancy design; that is, the backend server stores both the currently involved key and the previously used key for a valid tag ID to allow a tag with de-synchronized key to successfully communicate and re-synchronize its key value with the server in the next authentication session. Note that in some schemes, key redundancy mechanisms are adopted at tag side instead of the backend server. In this paper, we want to verify whether existing RFID authentication protocols supporting forward/backward security can resist de-synchronization attack in practical setting (i.e., the network environment in real world).

In the design of RFID authentication, the extremely limited capabilities of RF tags make it difficult to maintain the computation cost of tags as low as possible and at the same time achieve strong security and privacy. The limitation inspires academic scholars to reform traditional cryptographic algorithms for the needs in constructing a secure and efficient authentication scheme for RFID systems. Along with this trend, the study on the formal analysis model for RFID security and privacy has promptly been focused by research community. In 2006, Juels and Weis [24] proposed a formal definition for RFID privacy (denoted as ind-privacy), and revealed the vulnerabilities of some privacy-aware RFID protocols. Later, Ha et al. [21] pointed out that previously proposed adversarial models have limitations on analyzing RFID location privacy. A formal analysis model (denoted as unp-privacy), which is based on random oracle and indistinguishability, was accordingly introduced. To pursue practicability, the proposed model considers passive and active attacks on the message flows between the reader and the tags as well as the tag compromise attack. Nevertheless, Deursen and Radomirović [18] had demonstrated that the formal model for RFID location privacy in [21] does not coincide with the intuitive notion of location privacy. At the same year, Damgrd et al. [14] introduced the completeness and soundness concepts based on the model proposed in [24]. In 2009, Ma et al. [35] presented their efforts and findings: (1) refining the unp-privacy model according to its own flaws pointed in the study [18]; (2) proving that unp-privacy implies ind-privacy; (3) determining the minimal condition for RFID tags to achieve unp-privacy in an RFID system; and (4) developing an RFID protocol possessing strong entity privacy and performance efficiency. Next, Ng et al. [38] presented privacy analysis on symmetric based RFID authentication schemes. The authors divided existing RFID authentication protocols into four classes and demonstrated the achievable privacy level for each class. In addition, a strong security claim is argued; that is, forward privacy is impossible in existing RFID authentication proposals if public key cryptography cannot be adopted. In 2010, Deng et al. [15] introduced a zero-knowledge based framework for RFID privacy. The proposed framework is stronger than ind-privacy [35]. Furthermore, an efficient and robust RFID authentication scheme is introduced with a formal proof.

2. **Preliminaries.** An RFID system generally consists of many objects attached with RFID tags (i.e., transponders), an RFID reader (i.e., transceiver) and a backend application server. An RFID tag is composed of limited memory space, basic control and computation circuits and a radio frequency communication module. An RFID reader is used to acquire data stored in tags without line of sight restriction. The backend application server is responsible for retrieving and utilizing the detail information of objects

attached by RFID tags from corresponding databases. In the normal operation process of an RFID-based application system, the reader broadcasts RF signals to energize and inquiry tags in their RF broadcast range. Once a tag is invoked by RF query signals from the reader, the tag will respond a reply message with pre-defined message format. In general, a unique identification number is given back to the reader. Afterwards, the reader processes the received tag message if necessary, and forwards it as a service request to the backend server. After receiving the service request, the backend server executes corresponding business logic and responds this service request with the processing result back to the reader and/or the tag if necessary. Figure 1 shows an RFID communication environment which consists of a reader, a backend server and multiple tags.

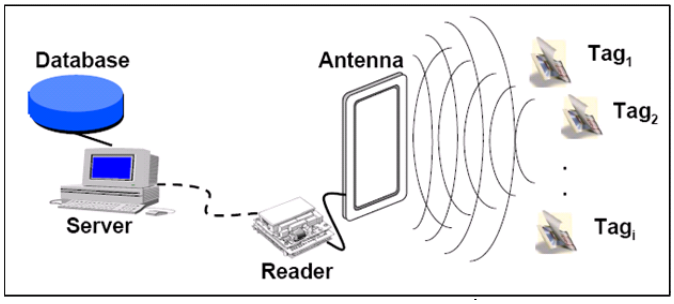

FIGURE 1. An RFID system

This subsection presents formal definitions for RFID systems and a new concept called *authentication availability*. An RFID system is considered as comprising a backend application server (with its own database) S, a single reader R and a set of n tags $T_1, \ldots, T_n$ in which all of them are probabilistic polynomial time Turing machines. Typically, a tag means a passive transponder identified by a unique ID, and has limited memory for secret keys and/or state information. All legitimate tags have registered at S side, and can only be identified and authenticated by S. In addition, R can request necessary data from S whenever it requires. During a protocol instance, all the messages exchanged between the tags and R are free to be intercepted, tampered and replayed. Moreover, the tags are not tamper-proofed and can be corrupted easily. Once corrupted, all the internal secrets and memory contents are assumed to be readily available to the adversary.

Since the design of RFID authentication, the security of backend communication channel between R and S is assumed. However, in real network environments it may pave a way for attackers to invoke simple transmission task as parts of malicious attacks (or behaviors). It is highly possible for an adversary to simulate the *server reachability* without breaking any secure communication or entity authentication mechanisms adopted between the reader and the server. For example, in a real network environment, if there is no authentication scheme deployed in packet level, an attacker can easily inject a message which eventually reaches the server S without breaking any application level security on communications. Based on the above clarifications, we believe that in real network environments an adversary Ad can control the communications among parties and interact with them through the following oracle queries.

- (O_1) InitReader( ). This oracle allows Ad to invoke a RF reader to start a session i of the target protocol, and get back a session identifier *sid* and a challenge message c_i.
- (O_2) Send(T_j, i, m). This oracle allows Ad to send a message m to any given tag T_j, and get back T_j's response in session i.

- (O_3) SendToReach(S, i, m). This oracle allows Ad to send a message m to reach the server S in session i. Note that Ad must not receive any response back.
- (O_4) Eavesdrop(R, T_j, i, m). This oracle models passive attack by allowing Ad to eavesdrop and get read access to the message m exchanged between R and any given tag T_j in session i.
- (O_5) Intercept(R, T_j, i, m). This oracle models active attack by allowing Ad to interrupt the message m transmitted between R and any given tag T_j in session i.
- (O_6) SetTag(T_j). This oracle models active attack by allowing Ad to update key and state information to tag T_j and return T_j's current key and internal state information.

Oracle O_1 which can be realized as an adversary can easily purchase and get access to a reader device in a realistic environment provided that the target authentication protocol is open to public usage or proposed as a standard. Once the oracle O_1 is supported, oracles O_2, O_4 and O_5 can be developed without much effort for an adversary under the assumption that the wireless communication channel between tag and reader is insecure. As mentioned before, the oracle O_3 is for an adversary Ad to simulate the *server reachability* instead of breaking secure communication or authentication mechanisms between reader and server. This query operation O_3 will be successful as long as an adversary can find a way to pass the message m to reach the server S. This is highly possible in real network environments once no extra authentication mechanism is deployed in packet level. Finally, the oracle O_6 is reasonable as the vulnerability of tag is assumed.

Definition 2.1. ***(Authentication Availability):*** *Assume that at the end of session $i-1$, the secret s_j shared between any given tag T_j and the backend application server S is synchronized. S and T_j will accept request/response messages $[c_i, r_i, f_i]$ with probability 100% during the next session i, where c_i is a challenge message, r_i is T_j's response protected by the secret s_j, and f_i is the final message based on c_i, r_i and s_j.*

Experiment $Exp_{Ad}^{Availability}[\boldsymbol{sp}, \boldsymbol{n}, \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}, \boldsymbol{v}, \boldsymbol{w}, \boldsymbol{x}]$

- Initialize $RAP()$: setup the reader R and a set T of n tags $T_1, T_2, \ldots, T_n$;
- $\{T_j, c_i, st\} \leftarrow A_1^{O_1,O_2,O_3,O_4,O_5,O_6}[R, S, T]$; *//learning stage*
- $b \in_R \{0, 1\}$;
- If $b = 0$ then $r_i \leftarrow RAP(R, S, T_j, c_i, s_j)$ and $f_i \leftarrow RAP(R, S, T_j, c_i, r_i, s_j')$ else $(r_i, f_i) \leftarrow RAP(R, S, T_j, c_i, r_i, s_j)$;
- $b' A_2^{O_1,O_2,O_3,O_4,O_5,O_6}[R, S, T_j, s_t, c_i, r_i, f_i]$; *//guessing stage*
- The experiment outputs 1 if $b' = b$, 0 otherwise.

The $Exp_{Ad}^{Availability}$ $[sp, n, p, q, r, v, w, x]$ is a game-based experiment for the adversary Ad to test the availability of any given target RFID authentication protocol $RAP()$ in which sp is the security parameter and n, p, q, r, v, w, x are experiment parameters. In the experiment, the adversary Ad (consisting of algorithms A_1 and A_2) is given $RAP()$ as the input and allowed to launch O_1, O_2, O_3, O_4, O_5 and O_6 oracle queries without exceeding n, p, q, r, v, w and x overall calls, respectively. At first, the experiment initializes $RAP()$ by producing a reader R and n-tags set $T = \{T_1, T_2, \ldots, T_n\}$ according to the security parameter sp. In the *learning stage*, algorithm A_1 selects the target tag T_j and a challenge message c_i. Meanwhile, a state information st is output. Next, the experiment selects a random bit b, and sets $r_i \leftarrow RAP(R, S, T_j, c_i, s_j)$ and $f_i \leftarrow RAP(R, S, T_j, c_i, r_i, s_j')$ if $b = 0$, and $(r_i, f_i) \leftarrow (R, S, T_j, c_i, r_i, s_j)$ otherwise. Note that s_j and s_j' are two different secret values. Next, in the *guessing stage*, algorithm A_2 has oracle accesses to R, S, T_j, st, c_i, r_i and f_i, and requires inferring whether r_i and f_i are involved with the same secret s_j or not.

Definition 2.2. *Let E be the event that occurs if either S or T_j does not accept $[c_i, r_i, f_i]$ during any given session i. An adversary $Ad(\varepsilon, t, p, q, r, v, w, x)$-breaks the availability of the target RFID authentication protocol $RAP()$ if the probability that E occurs, i.e., $Pr[E]$, is at least ε and the running time of Ad is at most t, where ε is non-negligible and t is a polynomial time which depends on the execution time of O_1, O_2, O_3, O_4, O_5 and O_6. In brief, the RFID authentication protocol $RAP()$ provides $(\varepsilon, t, p, q, r, v, w, x)$-availability if there exists no adversary $Ad(\varepsilon, t, p, q, r, v, w, x)$-breaks the availability of $RAP()$.*

3. Analysis on Existing RFID Authentication Mechanisms against De-synchr-onization Attack.

To defend against de-synchr-onization attack, recent RFID authentication protocols adopt a so-called key redundancy design; e.g., the backend server (or the tag) stores both the currently involved key and the previously used key for a valid tag ID to allow a tag with de-synchronized key to successfully communicate and re-synchronize its key value with the server in the next authentication session. In this section, we first categorize existing RFID authentication protocols into three types, i.e., types I, II and III. Security evaluation on de-synchronization attack is then conducted for protocols associated with types I and II, respectively. Our results show that the key synchronization mechanisms used in types I and II protocols cannot defend against de-synchronization attack.

3.1. General operation components and mechanisms used in protocol.

1. $O^{Tag}()$, $O^{Server}()$: A collection of operations denoted as an oracle following the protocol specification carried out on the tag and the server side, respectively.
2. K_{ID}^i: The tag key at session i where the initial key is K_{ID}^0.
3. S_{ID}^i: The tag state at session i denoted as an encapsulation of the tag key K_{ID}^i and other per instance generated and received values. If S_{ID}^i is updated to S_{ID}^{i+1}, K_{ID}^i is updated to K_{ID}^{i+1} as well.
4. $O^{Update}(S_{ID}^i)$: A tag key update operation performed on the tag side which takes S_{ID}^i as input and outputs K_{ID}^{i+1}.
5. *key redundancy design*: Two redundant records of secret key value shared between S and T_j (e.g., currently involved key K_{ID}^i and the key K_{ID}^{i-1} used in the last session).
6. *key independent update*: The newly updated key K_{ID}^{i+1} is independent of the input value S_{ID}^i at any given session i (e.g., $K_{ID}^{i+1} \neq K_{ID}^{i+2}$ in which $K_{ID}^{i+1} \leftarrow O^{Update}(S_{ID}^i)$ at session i, and $K_{ID}^{i+2} \leftarrow O^{Update}(S_{ID}^i)$ at session $i+1$).
7. *key dependent update*: The newly updated key K_{ID}^{i+1} is dependent on the input value S_{ID}^i at any given session i (e.g., $K_{ID}^{i+1} = K_{ID}^{i+2}$ in which $K_{ID}^{i+1} \leftarrow O^{Update}(S_{ID}^i)$ at session i, and $K_{ID}^{i+2} \leftarrow O^{Update}(S_{ID}^i)$ at session $i+1$).

Next, we classify existing RFID authentication protocols based on where *key redundancy design* is adopted (e.g., at the tag side or the server side) and which *key update* mechanism is utilized (e.g., *dependent or independent*). Protocols out of our classification either cannot guarantee forward/backward security properties [4,12,23,35-37,48,53] or are vulnerable to de-synchronization attack [10,13,16,26,27,39,41-43,45]. We briefly introduce each protocol subgroup as follows.

1. Type I protocols [1,5,6,28,31-34,49-51] involve with *key independent update*, and its *key redundancy design* is adopted the server side. (Please refer to Figure 2)
2. Type II protocols [9,44,56] involve with *key independent update*, and its *key redundancy design* is adopted the tag side. (Please refer to Figure 3)
3. Type III protocols [8,11,54,55] possess *key dependent update* and its *key redundancy design* is adopted either at the server side or at the tag side.

3.2. **Type I protocols are vulnerable to de-synchronization attack.**

Theorem 3.1. *Type I schemes [1,5,6,28,31-34,49-51] are vulnerable to de-synchronization attack. For any given tag T_j, Type I protocols cannot provide at least $(\varepsilon, t, 2, 1, 1, 0, 0, 0)$-availability (or at least $(\varepsilon, t, 1, 0, 1, 2, 1, 0)$-availability).*

Proof: We demonstrate how to break the availability of Type I protocols in a polynomial time. Given the target Type I RFID authentication protocol $RAP()$ and its corresponding security parameter *sp*, the adversary *Ad* considers the following de-synchronization attack processes. Note that in the session $i-1$, the secrets shared between T_j and S is synchronized. Let the key values at the server side are K_{ID}^{i} and K_{ID}^{i-1}, and the key value at the tag side is K_{ID}^{i}.

The first phase (session *i*):

- System initialization: *Ad* recognizes $RAP()$ with the security parameter *sp*.
- InitReader( ): *Ad* selects the target tag T_j, and utilizes the oracle O_1 to invoke a reader R and start a new session of $RAP()$. Then, *Ad* obtains the session identifier i, a state information *st* and a challenge message c_i.
- Send(T_j, i, c_i): *Ad* utilizes the oracle O_2 to send c_i to T_j, and receive a tag response r_i. These two values c_i and r_i are temporarily maintained and will be exploited in the third phase. Note that the first two steps, i.e., InitReader( ) and Send (T_j, i, c_i), can also be accomplished via the combination of oracle queries O_4 and O_5. That

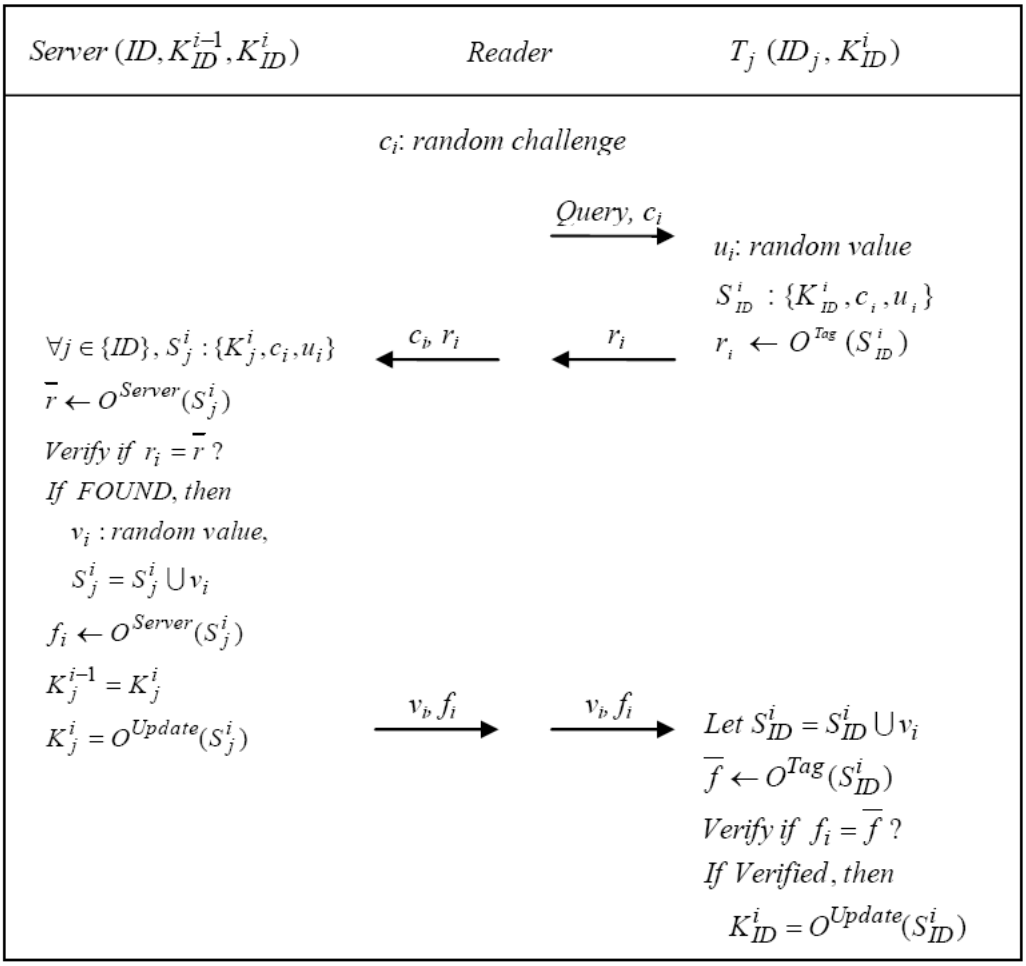

FIGURE 2. The normal operation process of Type I protocols in session i

is, Ad can execute two times of oracle query O_4 to eavesdrop c_i and r_i, respectively, exchanged between a legitimate reader R' and T_j. Next, the oracle query O_5 is invoked to interrupt the rest transmission (i.e., r_i, v_i and f_i) between R' and T_j.

At the end of this phase, the key values at the server side are K_{ID}^i and K_{ID}^{i-1}, and the key value at the tag side is K_{ID}^i.

The second phase (session $i+1$):

- In this session, Ad is suspended and monitors the channel involved with T_j until a whole operation process of $RAP()$ between another legitimate reader R'' and T_j is performed successfully. Note that in session $i+1$, c_{i+1}, r_{i+1}, v_{i+1} and f_{i+1} are transmitted.
- So far, the key values at the server side are $K_{ID}^{i+2} \leftarrow O^{Update}(S_{ID}^i)$ and K_{ID}^{i-1}, and the key value at the tag side is $K_{ID}^{i+2} \leftarrow O^{Update}(S_{ID}^i)$.

The third phase (session $i+2$):

- Once the second phase is done, Ad performs the following procedures immediately.
- InitReader( ): Ad selects the target tag T_j, and uses the oracle O_1 to invoke R to start a new session of $RAP()$. Ad then gets the session identifier $i+2$, a state information st and a challenge message c_{i+2}.
- SendToReach(S, $i+2$, $\{c_i, r_i\}$): Ad uses the oracle query O_3 to send $\{c_i, r_i\}$ to S. Since $\{c_i, r_i\}$ are involved with key K_{ID}^i, $\{c_i, r_i\}$ will be successfully verified at S side. After that, S performs the key update mechanism, i.e., $K_{ID}^{i+3} \leftarrow O^{Update}(S_{ID}^i)$ and K_{ID}^{i-1}.

Finally, Ad finishes the experiment and outputs a bit b' as its conjecture of the value of b from $Exp_{Ad}^{Availability}$. As $RAP()$ adopts *key independent update*, the key value shared between S and T_j is out-of-synchronization now. The secret keys at S side are K_{ID}^{i-1} and K_{ID}^{i+3}, and the key at T_j side is K_{ID}^{i+2}. Since in *key independent update* the updated key is independent of the input value, it is obvious that K_{ID}^{i+3} is not the same with K_{ID}^{i+2}. In that case, the adversary Ad can always make a correct guess of b with the above three attack steps, where only 2, 1, 1, 0, 0 and 0 execution times of the oracle queries O_1, O_2, O_3, O_4, O_5 and O_6 are required, respectively. As the probability that $Ad(\varepsilon, t, 2, 1, 1, 0, 0, 0)$-break the availability of $RAP()$ is significant, i.e., $Adv(\varepsilon, t, 2, 1, 1, 0, 0, 0) = |Pr[Ad's\ guess\ is\ correct] - 50\%| = 50\%$, and the running time of Ad is polynomial, we can conclude that Type I protocols cannot provide at least $(\varepsilon, t, 2, 1, 1, 0, 0, 0)$-availability. Note that the oracle queries O_4 and O_5 can be utilized to replace the oracles O_1 and O_2 in the first phase. This leads to another conclusion that Type I protocols cannot guarantee at least $(\varepsilon, t, 1, 0, 1, 2, 1, 0)$-availability. Theorem 3.1 is proved.

Example 3.1. ***The CLLD Protocol [6] Is Vulnerable to De-synchronization Attack.***

•Review of CLLD Protocol

Every tag T_j is assigned with an l-bit identifier $t_j = h(u_j)$, where u_j is an l-bit string and $h()$ is a one-way hash function. For each T_j, the server (with a database) stores an entry $[(u_j, t_j)_{new}, (u_j, t_j)_{old}, D_j]$ in which $(u_j, t_j)_{new}$ denotes the currently involved identity, $(u_j, t_j)_{old}$ represents the last successfully verified identity, and D_j is T_j's information. The normal process of CLLD protocol is as follows.

$$R \rightarrow T_j : r_i.$$

The reader R generates a random bit-string $r_1 \in_R \{0,1\}^l$, and sends it to tag T_j. Then, T_j generates a random bit string $r_2 \in_R \{0,1\}^l$ as a secret, and computes $M = t_j \oplus r_2$

and $M_2 = f_j(r_1 \| r_2)$, where $f()$ is a keyed hash function. Next, T_j sends M_1 and M_2 to R which soon forwards them along with r_1 to the backend server S.

$$T_j \rightarrow R \rightarrow S : M_1, M_2, r_1$$

Upon receiving M_1, M_2 and r_1, S retrieves each t_j from all stored tag identity pairs (new and old), and verifies (for each t_j) whether the received value M_2 equals to the computed value $f_1(r_1 \| r_2)$ in which $r_2 \leftarrow M_1 \oplus t_j$. If no t_j satisfies the above verification, S sends an error message to R and terminates the protocol. On the other hand, if t_j is found among the $(u_j, t_j)_{old}$ pairs, the server S recognizes that the tag T_j failed to complete the whole process at the last authentication session, and T_j's identity is not updated. S then sets $(u_j, t_j)_{new} \leftarrow (u_j, t_j)_{old}$, and continues with the protocol as normal. With the corresponding u_j, the server computes $M_3 = s_j \oplus h(r_2)$, and sends it to R along with D_i. Meanwhile, S updates the secrets, i.e., $(u_j, t_j)_{old} = (u_j, t_j)_{new}$, $u_{j(new)} \leftarrow (u_j \ll l/4) \oplus (t_j \gg l/4) \oplus r_1 \oplus r_2$ and $t_{j(new)} \leftarrow h(u_{j(new)})$.

$$S \rightarrow R \rightarrow T_j : M_3$$

The reader sends M_3 to T_j. Once T_j receives M_3, it computes $s_j \leftarrow M_3 \oplus h(r_2)$ and $h(s_j)$, and checks if $h(s_j) = t_j$. If it holds, T_j updates t_j to $h((u_j \ll l/4) \oplus (t_j \gg l/4) \oplus r_1 \oplus r_2)$. Otherwise, t_j remains the same.

•De-synchronization Attack on CLLD Protocol [6]

A synchronized tag is assumed in which the secret information, i.e., $t_j = (t_j)_i$, maintained at tag side equals to the values, i.e., $(u_j, t_j)_{old} = (u_j, t_j)_{i-1}$ and $(u_j, t_j)_{new} = (u_j, t_j)_i$, stored in the server/database (DB). Note that we denote the secrets as $(t_j)_i$ and $(u_j, t_j)_i$ during session i.

The first phase (session i):

- System initialization: Ad recognizes CLLD protocol with the security parameter sp.
- InitReader( ): Ad selects the target tag T_j, and utilizes the oracle query O_1 to invoke a reader R to start a new session of CLLD protocol. After that, Ad obtains the session identifier i, a state information st and a challenge message r_{1_i}.
- Send(T_j, i, r_{1_i}): Ad utilizes the oracle query O_2 to send r_{1_i} to T_j, and gets back a tag response $\{M_{1\_i}, M_{2\_i}\}$. The value $\{M_{1\_i}, M_{2\_i}, r_{1\_i}\}$ are temporarily maintained and will be used in the third phase.
- At the end of this phase, the key values at the server side are $(u_j, t_j)_{i-1}$ and $(u_j, t_j)_i$, and the key value at the tag side is $(t_j)_i$.

The second phase (session $i+1$):

- In this phase, Ad monitors T_j's communication channel until a whole operation process of CLLD protocol between another reader R'' and T_j is performed completely. Note that in session $i+1$, M_{1_i+1}, M_{2_i+1}, r_{1_i+1}, r_{2_i+1} and M_{3_i+1}, are produced.
- So far, the key values at the server side are $(u_j, t_j)_i$ and $(u_j, t_j)_{i+2}$ and the key value at the tag side is $(t_j)_{i+2}$.

The third phase (session $i+2$):

- Once the second phase is done, Ad performs the following procedures immediately.
- InitReader( ): Ad selects the target tag T_j, and uses the oracle query O_1 to invoke R to start a new session of CLLD protocol. Ad then gets the session identifier $i+2$, a state information st and a challenge message r.
- SendToReach$(S, i+2, \{M_{1_i}, M_{2_i}, r_{1_i}\})$: Ad uses the oracle query O_3 to send $\{M_{1\_i}, M_{2\_i}, r_{1\_i}\}$ to S. Since $\{M_{1\_i}, M_{2\_i}, r_{1\_i}\}$ are involved with key $(u_j, t_j)_i$, the legitimacy of $\{M_{1\_i}, M_{2\_i}, r_{1\_i}\}$ will be examined successfully at S side. Then, S updates the keys, i.e., $(u_j, t_j)_i$ and $(u_j, t_j)_{i+3}$.

Finally, Ad finishes the experiment and outputs a bit b' as its conjecture of the value of b from $Exp_{Ad}^{Availability}$. Obviously, the key values at S side are $(u_j, t_j)_i$ and $(u_j, t_j)_{i+3}$, and the key value at (T_j side is $(t_j)_{i+2}$. Since CLLD protocol adopts *key independent update*, the key value shared between S and T_j is out-of-synchronization now, i.e., $(t_j)_{i+2}$ is not equal to $(t_j)_{i+3}$. In that case, the adversary Ad can make a correct guess of b with the above attack steps, where only 2, 1, 1, 0, 0 and 0 execution times of the oracle O_1, O_2, O_3, O_4, O_5 and O_6 are required, respectively. As the probability that $Ad(\varepsilon, t, 2, 1, 1, 0, 0, 0)$-break the availability of CLLD protocol is significant, i.e., $Adv(\varepsilon, t, 2, 1, 1, 0, 0, 0) = |Pr[Ad's\ guess\ is\ correct] - 50\%| = 50\%$, and the running time of Ad is polynomial, we argue that CLLD protocol cannot provide $(\varepsilon, t, 2, 1, 1, 0, 0, 0)$-availability.

Example 3.2. ***The ACA Protocol [1] Is Vulnerable to De-synchronization Attack.***

• Review of ACA Protocol

In ACA protocol, each tag T_j is assigned with two parameters, i.e., an l-bits id_j and an l-bits $val_j = h(seed_j)$. Note that l should be large enough to prevent exhaustive search attack of $seed_j$. For each T_j, the server (with a database) stores an entry $[(id_j, seed_j)_{new}, (id_j, seed_j)_{old}, D_j]$ in which $(id_j, seed_j)_{new}$ is the currently involved identity and $(id_j, seed_j)_{old}$ represents the last successfully verified identity. At the system initialization, $(id_j, seed_j)_{new}$ is equal to $(id_j, seed_j)_{old}$. The normal operation process of ACA protocol is as follows.

(1) The reader R generates a random bit-string $r_1 \in_R \{0,1\}^l$ and sends $h(r_1)$ to tag T_j. Next, T_j generates a random bit string $r_2 \in_R \{0,1\}^l$, and computes $M_1 = pf(h(r_1)\|id_j)$ and $M_2 = pf(h(r_2)\|id_j)$. Then, T_j sends M_1 and r_2 to R. Upon receiving M_1 and r_2, the reader R queries the backend server S with $\{M_1, r_1, r_2, h(r_1)\}$.

(2) Once the server S obtains M_1, r_1, r_2 and $h(r_1)$, S retrieves each id_j from all stored tag identity pairs (new and old) from database, and verifies whether the received value M_1 equals to the computed value $M_1 = pf(h(r_1)\|id_j)$. If no id_j satisfies the examination, the server sends an error message to the reader and the protocol stops. If id_j is found among the $(id_j, seed_j)_{old}$ pairs, the server then sets $(id_j, seed_j)_{new} \leftarrow (id_j, seed_j)_{old}$, and continues the protocol. With the corresponding entry $[(id_j, seed_j)_{new}, (id_j, seed_j)_{old}, D_j]$, the server S computes $M_1 = pf(h(r_2)\|id_j)$ and $M_3 = seed_j \oplus M_2$, and sends it to the reader R along with D_j. Meanwhile, S updates the secrets, i.e., $(id_j, seed_j)_{old} = (id_j, seed_j)_{new}$, $id_{jnew} \leftarrow g(h(r_1) \oplus r_2 \oplus seed_{j(new)} \oplus id_{j(new)})$ and $seed_{j(new)} \leftarrow r_1$.

(3) Upon receiving the server response, R sends M_3 to T_j. After the tag T_j receives M_3, T_j computes $seed_j \leftarrow M_3 \oplus M_2$, and checks if $h(seed_j) = val_j$. If it holds, the tag T_j updates id_j to $g(h(r_1) \oplus r_2 \oplus seed_j \oplus id_j)$ and $val_j = h(r_1)$.

• De-synchronization Attack on ACA Protocol [1]

Given ACA protocol, the adversary Ad performs the following malicious attack phases to de-synchronize the secrets, i.e., id_j and $seed_j$, shared between the server S and the tag T_j. We assume that in the session $i-1$, the secrets shared between T_j and S are as follows: the secrets at the server side are $(id_j, seed_j)_{old} = (id_j, seed_j)_{i-1}$ and $(id_j, seed_j)_{new} = (id_j, seed_j)_i$ and the one at the tag side is $(id_j) = (id_j)_i$.

The first phase (session i):

- System initialization: Ad recognizes ACA protocol with the security parameter sp.
- InitReader( ): the adversary Ad selects the target tag T_j, and utilizes the oracle query O_1 to invoke a reader R to start a new session of the ACA protocol. After

that, Ad obtains a session identifier i, a state information st and a challenge message $\{r_{1\_i}, h(r_{1\_i})\}$.

- Send$(T_j, k, h(r_{1_i}))$: the adversary Ad uses the oracle query O_2 to send $h(r_{1\_i})$ to T_j, and gets back T_j's response $\{M_{1\_i}, r_{2\_i}\}$. The response message $\{M_{1\_i}, r_{1\_i}, r_{2\_i}, h(r_{1\_i})\}$ are temporarily maintained and will be used in the third phase.
- At the end of this phase, the secret values at the server side are $(id_j, seed_j)_{old} = (id_j, seed_j)_i$ and $(id_j, seed_j)_{new} = (id_j, seed_j)_{i+1}$ and the one at the tag side is $(id_j) = (id_j)_i$.

The second phase (session $i+1$):

- In this phase, the adversary Ad is suspended and monitors the channel involved with T_j until a new session of ACA protocol is held between another reader R' and T_j. Note that in session $i+1$, $M_{1\_i+1}, M_{2\_i+1}, r_{1\_i+1}, h(r_{1\_i+1}), r_{2\_i+1}$ and M_{3_i+1} are generated.
- At the end of this phase, the secret values at the server side are $(id_j, seed_j)_i$ and $(id_j, seed_j)_{i+2}$, and the secret value at the tag side is $(id_j)_{i+2}$.

The third phase (session $i+2$):

- Once the second phase is done, Ad performs the following procedures immediately.
- InitReader( ): the adversary Ad selects the target tag T_j, and uses the oracle query O_1 to invoke R to start a new session of the ACA protocol. Ad then gets the session identifier $i+2$, a state information st and a challenge message $\{r_{1\_i+2}, h(r_{1\_i+2})\}$.
- SendToReach$(S, i+2, \{M_{1_i}, r_{1_i}, r_{2_i}, h(r_{1_i})\})$: Ad performs the oracle query O_3 to send $\{M_{1\_i}, r_{1\_i}, r_{2\_i}h(r_{1\_i})\}$ to the backend server S. As $\{M_{1\_i}, r_{1\_i}, r_{2\_i}h(r_{1\_i})\}$ are actually involved with secrets $(id_j, seed_j)_i$, $\{M_{1\_i}, r_{1\_i}, r_{2\_i}h(r_{1\_i})\}$ will be successfully verified at S side. Then, S performs the secrets update mechanism.
- Finally, the secret values at S side are $(id_j, seed_j)_i$ and $(id_j, seed_j)_{i+3}$, and the secret value at the tag side is $(t_j)_{i+2}$. Since ACA protocol utilizes *key independent update*, the secrets shared between S and T_j are out-of-synchronization now.

With the above attack procedures, the adversary Ad does make a correct guess of b in which only 2, 1 and 1 execution times of O_1, O_2 and O_3 is required, respectively. As the probability that $Ad(\varepsilon, t, 2, 1, 1, 0, 0, 0)$-break the availability of ACA protocol is significant, i.e., $Adv(\varepsilon, t, 2, 1, 1, 0, 0, 0) = |Pr[Ad's\ guess\ is\ correct] - 50\%| = 50\%$, and the running time of Ad is polynomial, the insecurity of ACA protocol is proved.

3.3. **Type II protocols are vulnerable to de-synchronization attack.**

Theorem 3.2. *Type II schemes [9,44,56] are vulnerable to de-synchronization attack. For any given tag T_j, Type II protocols cannot provide at least $(\varepsilon, t, 1, 3, 0, 1, 1, 1)$-availability (or $(\varepsilon, t, 1, 3, 0, 1, 1, 1)$-availability).*

Proof: Given the target Type II RFID authentication protocol $RAP()$ and its corresponding security parameter sp, the adversary Ad considers the following de-synchronization attack procedures. Note that in the session $i-1$, the secrets shared between T_j and S are synchronized. Let the key value at the server side is K_{ID}^i, and the key values at the tag side are K_{ID}^i and K_{ID}^{i-1}.

The first phase (session i):

The adversary Ad continuously monitors the communication channel involved with T_j. Once a session i of $RAP()$ is invoked between the reader R' and T_j, Ad acts as follows.

- Eavesdrop(R', T_j, i, f_i): Ad invokes the oracle query O_4 to eavesdrop f_j transmitted between R' and T_j, and temporarily records f_i.

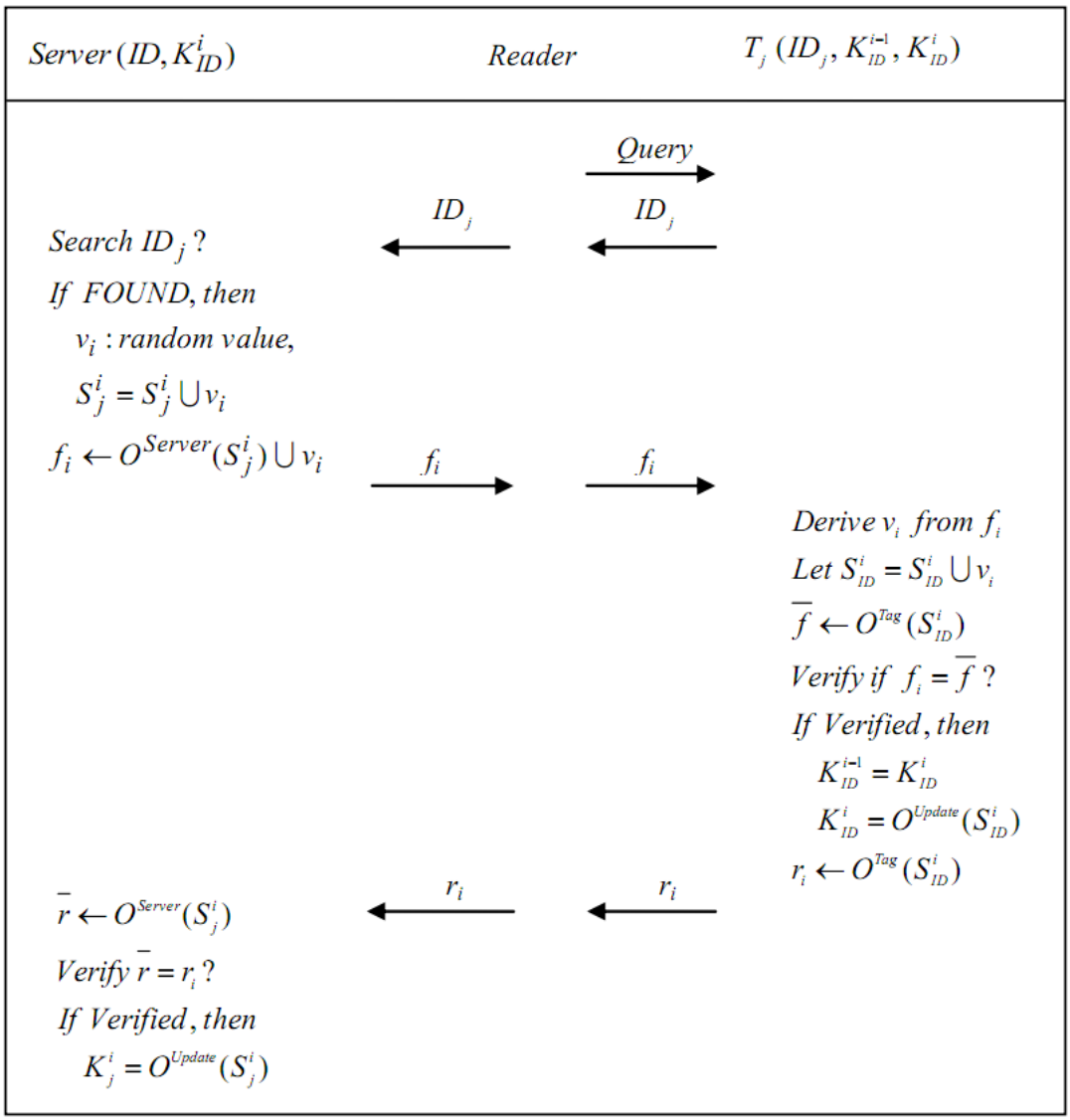

FIGURE 3. The normal operation process of session i in type II protocols

- Intercept(R', T_j, i, r_i): Ad utilizes the oracle query O_5 to interrupt r_i transmitted between R' and T_j.
- After that, the key value at the server side is K_{ID}^i, and the key value at the tag side is $K_{ID}^{i+1} \leftarrow O_{update}(S_{ID}^i)$ and K_{ID}^i.

The second phase (session $i+1$):

Ad monitors the channel involved with T_j until a new session (i.e., $i+1$) of $RAP()$ held between another reader R'' and T_j is completed. Note that in session $i+1$, c_{i+1}, r_{i+1}, v_{i+1} and f_{i+1} are transmitted. So far, the key values at the tag side are $K_{ID}^{i+2} \leftarrow O^{Update}(S_{ID}^i)$ and K_{ID}^i, and the key value at the server side is $K_{ID}^{i+2} \leftarrow O^{Update}(S_{ID}^i)$.

The third phase (session $i+2$):

Once the second phase is done, Ad performs the following procedures immediately.

- InitReader( ): Ad selects the target tag T_j, and uses the oracle query O_1 to invoke R to start a new session of $RAP()$. Ad then gets back a session identifier $i+2$, state information st and a challenge message c_{i+2}. Next, Ad queries T_j which first replies message involved with K_{ID}^{i+2} and then sends message involved with K_{ID}^i, once

Ad pretends that he/she cannot find the corresponding K_{ID}^{i+2} in the backend server. This step consumes two oracle queries O_2.

- Send$(T_j, i+2, f_i)$: Ad uses the oracle query O_2 to send f_i to T_j, where f_i are involved with key K_{ID}^i. Hence, f_i will be examined successfully by T_j. Next, T_j updates the key, i.e., $K_{ID}^{i+3} \leftarrow O^{Update}(S_{ID}^i)$ and K_{ID}^i.

Finally, Ad finishes the experiment and outputs a bit b' as its conjecture of the value of b from $Exp_{Ad}^{Availability}$. The key value shared between S and T_j is out-of-synchronization now as $RAP()$ adopts *key independent update* mechanism. Note that the key value at S side is K_{ID}^{i+2}, and the key values at T_j side are K_{ID}^i and K_{ID}^{i+3}. Since in $RAP()$ the updated key is always independent of the input value, it is obvious that K_{ID}^{i+3} is not equal to K_{ID}^{i+2}. In that case, the adversary Ad will make a correct guess of b with the above attack steps in which only 1, 3, 0, 1, 1 and 0 execution times of the oracle O_1, O_2, O_3, O_4, O_5 and O_6 are needed, respectively. As the probability that $Ad(\varepsilon, t, 1, 3, 0, 1, 1, 0)$-break the availability of $RAP()$ is significant, i.e., $Adv(\varepsilon, t, 1, 3, 0, 1, 1, 0) = |Pr[Ad's\ guess\ is\ correct] - 50\%| = 50\%$, and the running time of Ad is polynomial, we can conclude that the Type II protocols cannot provide at least $(\varepsilon, t, 1, 3, 0, 1, 1, 0)$-availability. Note that some Type II protocols such as [56] need one more attack step to invoke oracle query O_6. Theorem 3.2 is proved.

Example 3.3. ***Gossamer Protocol [44] Is Vulnerable to De-synchronization Attack.***

• Review of Gossamer Protocol

In Gossamer protocol, each tag T_j stores a static identifier (ID), two index pseudonym $(IDS_{old}$ and $IDS_{new})$ and four secret keys $(k_{1\_old}, k_{1\_new}, k_{2\_old}, k_{2\_new})$, where *new/old* represents the parameter used in the current/last session. The backend server maintains a static identifier (ID), an index-pseudonym (IDS) and two keys (k_1 and k_2). The tag can operate simple bitwise functions such as $XOR(\oplus)$, $AND(\vee)$, $OR(\wedge)$, *Addition* mod $2^m(+)$, circular shift rotation $(Rot(x, y))$ and *MixBits* operation. At the beginning of Gossamer protocol, the reader R sends a *hello* message to the tag T_j which soon responds with its *IDS*. Based on this *IDS*, R probes the corresponding information of T_j from the backend server.

$$R \rightarrow T_j : Hello$$
$$T_j \rightarrow R : IDS$$
$$R \rightarrow T_j : A \| B \| C$$

With the information, i.e., ID, IDS, k_1 and k_2, retrieved from the backend server, the reader R computes $A\|B\|C$ and sends them to T_j, where n_1 and n_2 are random numbers.

$$A = Rot(Rot(IDS + k_1 + \pi + n_1, k_2) + k_1, k_1);$$
$$B = Rot(Rot(IDS + k_2 + \pi + n_2, k_2) + k_2, k_2);$$
$$n_3 = MixBits(n_1, n_2); \quad n_1' = MixBits(n_3, n_2);$$
$$k_1^* = Rot(Rot(n_2 + k_1 + \pi + n_3, n_2) + k_2 \oplus n_3, n_1) \oplus n_3;$$
$$k_2^* = Rot(Rot(n_1 + k_2 + \pi + n_3, n_1) + k_1 + n_3, n_2) + n_3;$$
$$C = Rot(Rot(n_3 + k_1^* + \pi + n_1', n_3) + k_2^* \oplus n_1', n_2) \oplus n_1';$$
$$\pi = 0x3243F6A8885A308D313198A2.$$

From A and B, T_j can obtain two nonce values n_1 and n_2 respectively. T_j then computes C' and checks whether the result is equal to the received C. If both of them are equal, T_j

sends D to R, and updates its own secret parameters.

$C' = Rot(Rot(n_3 + k_1^* + \pi + n_1', n_3) + k_2^* \oplus n_1', n_2) \oplus n_1';$
$D = Rot(Rot(n_2 + k_2^* + ID + n_1', n_2) + k_1^* + n_1', n_3) + n_1';$
$n_2' = MixBits(n_1', n_3); \quad IDS_{old} = IDS; \quad k_{1\_old} = k_1; \quad k_{2\_old} = k_2;$
$IDS_{new} = Rot(Rot(n_1' + k_1^* + IDS + n_2', n_1') + k_2^* \oplus n_2', n_3) \oplus n_2';$
$k_{1\_new} = Rot(Rot(n_3 + k_2^* + \pi + n_2', n_3) + k_1^* + n_2', n_1') + n_2';$
$k_{2\_new} = Rot(Rot(IDS_{new} + k_2^* + \pi + k_{1\_new}, IDS_{new}) + k_1^* + k_{1\_new}^*, n_2') + k_{1\_new};$
$T_j \rightarrow R : D$

The reader R calculates D' and check whether the computed D' is equal to the received D. It it holds, R updates IDS, k_1 and k_2 in the same way as T_j does.

$$D' = Rot(Rot(n_2 + k_2^* + ID + n_1', n_2) + k_1^* + n_1', n_3) + n_1';$$
$$n_2' = MixBits(n_1', n_3);$$
$$IDS = Rot(Rot(n_1' + k_1^* + IDS + n_2', n_1') + k_2^* \oplus n_2', n_3) \oplus n_2';$$
$$k_1 = Rot(Rot(n_3 + k_2^* + \pi + n_2', n_3) + k_1^* + n_2', n_1') + n_2';$$
$$k_2 = Rot(Rot(IDS + k_2^* + \pi + k_1, IDS) + k_1^* + k_1, n_2') + k_1.$$

- **De-synchronization Attack on Gossamer Protocol [44]**

A synchronized tag T_j is given, where the secret information (IDS_{i-1} and IDS_i, k_{1_i-1}, $k_{1\_i}, k_{2\_i-1}, k_{2\_i}$) maintained at T_j side equals to the values (IDS_i, k_{1_i}, k_{2_i}) stored in the backend server. Note that we denote the secret as (IDS_i, k_{1_i}, k_{2_i}) during session i.

The first phase (session i):

Let the adversary Ad continuously monitor the communication channel involved with T_j. Once the normal process of session i of Gossamer protocol is invoked between the reader R' and T_j, Ad acts as follows.

- Eavesdrop($R', T_j, i, A\|B\|C$): Ad invokes the oracle query O_4 to eavesdrop $A\|B\|C$ transmitted between R' and T_j, and temporarily records $A\|B\|C$.
- Intercept(R', T_j, i, D): Ad utilizes the oracle query O_5 to interrupt D transmitted between R' and T_j.
- At the end of this phase, the backend server will not update the secret information (IDS, k_1, k_2) associated with T_j. However, T_j updates its own secrets. Therefore, the current status of shared secrets is as follows: ($IDS_{i+1}, k_{1\_i+1}, k_{2\_i+1}$) and ($IDS_i, k_{1\_i}, k_{2\_i}$) at T_j side, and (IDS_i, k_{1_i}, k_{2_i}) at server side.

The second phase (session $i+1$):

Let Ad monitor T_j's communication channel until a new session (i.e., $i+1$) of Gossamer protocol is successfully held by another reader R'' and T_j. In this phase, T_j utilizes the old record, i.e., IDS_i, k_{1_i}, k_{2_i}, to communicate with the reader as the IDS stored in the backend server is the old one. After that, the key values at the tag side are ($IDS_{i+2}, k_{1\_i+2}, k_{2\_i+2}$) and ($IDS_i, k_{1\_i}, k_{2\_i}$), and the key value at the server side is ($IDS_{i+2}, k_{1\_i+2}, k_{2\_i+2}$).

The third phase (session $i+2$):

Let Ad perform the following attack procedures.

- InitReader( ): Ad selects the target tag T_j, and uses the oracle query O_1 to invoke a reader R to start a new session $i+2$ of Gossamer protocol. Ad then queries T_j which first replies IDS_{i+2} and then sends IDS_i, once Ad pretends that he/she cannot find the IDS_{i+2} in the backend server. This step consumes two oracle queries O_2.

- Send$(T_j, i+2, A\|B\|C)$: Ad uses the oracle query O_2 to send $A\|B\|C$ to T_j, where $A\|B\|C$ are involved with $(IDS_i, k_{1\_i}, k_{2\_i})$. Hence, the legitimacy of $A\|B\|C$ will be passed at T_j side. Next, T_j updates the key values, i.e., $(IDS_{i+3}, k_{1\_i+3}, k_{2\_i+3})$ and $(IDS_i, k_{1\_i}, k_{2\_i})$.

Now the secrets shared between the server and the tag T_j are out-of-synchronization as Gossamer protocol adopts *key independent update* mechanism. Note that the key values at T_j side are $(IDS_{i+3}, k_{1\_i+3}, k_{2\_i+3})$ and $(IDS_i, k_{1\_i}, k_{2\_i})$, and the key value at the server side is $(IDS_{i+2}, k_{1\_i+2}, k_{2\_i+2})$. Finally, Ad finishes the experiment and outputs a bit b' as its conjecture of the value of b from $Exp_{Ad}^{Availability}$. It is obvious that Ad will always make a correct guess of b with the above attack steps in which only 1, 3, 0, 1, 1 and 0 execution times of the oracle O_1, O_2, O_3, O_4, O_5 and O_6, respectively, are performed. As the probability that $Ad(\varepsilon, t, 1, 3, 0, 1, 1, 0)$-break the availability of Gossamer protocol is significant, i.e., $Adv(\varepsilon, t, 3, 1, 0, 1, 1, 0) = |Pr[Ad's\ guess\ is\ correct] - 50\%| = 50\%$, and the running time of Ad is polynomial, we have proved that Gossamer protocol cannot guarantee $(\varepsilon, t, 1, 3, 0, 1, 1, 0)$-availability.

Example 3.4. ***The YW09 Scheme [56] Is Vulnerable to De-synchronization Attack.***

- **Review of YW09 Scheme [56]**

Every tag T_j is assigned with eight data records, i.e., ID, IDS_{old}, IDS_{new}, K_{1_old}, K_{1_new}, K_{2_old}, K_{2_new} and RID, which are stored in T_j's internal memory. Note that the currently involved records $[ID, IDS_{new}, K_{1\_new}, K_{2\_new}, RID]$ and the last successfully verified records $[ID, IDS_{old}, K_{1\_old}, K_{2\_old}, RID]$ are maintained simultaneously. For each T_j, the reader R (and the server S) maintains an entry $[ID, IDS, K_1, K_2, RID]$. At the system initialization, S generates IDS, K_1, K_2 for each tag T_j and sets the T_j's values such as $IDS_{old} = IDS_{new} = IDS$, $K_{1\_new} = K_{1\_old} = K_1$, $K_{2\_new} = K_{1\_old} = K_2$, $R_1 = R_1$ and $RID = RID$. The normal process of YW09 is as follows.

(1) Initially, the reader R sends a request message *Hello* to the tag T_j.

(2) Once T_j receives the *Hello* message, it first computes R_1 and then calculates $(IDS_{new} \| IDS_{new}) \oplus (RID \| R_1)$ and $RID + R_1$, and sends these two results to R, where $R_1 = (K_{1\_new} \oplus K_{2\_old}) + ((K_{2\_new} \oplus K_{1\_old}) \vee R_1)$. After receiving T_j's response, the reader R utilizes the RID retrieved from the server S (with its database) to derive values R_1 and IDS. Note that if the reader R can probe the matched record at S side, it steps to the following authentication procedures; otherwise, it interrogates T_j again and, after that, T_j will responds with $(IDS_{old} \| IDS_{old}) \oplus (RID \| R_1)$ and $RID + R_1$.

(3) The reader R then exploits the matched IDS and two newly generated random numbers n_1 and n_2 to calculate the values as follows. Next, the reader R sends $(A\|B\|C) \oplus (R_1\|R_1\|R_1)$ to T_j.

$$A = IDS \oplus K_1 \oplus n_1, \quad B = (IDS \vee K_2) + n_2, \quad K_1' = (K_1 \oplus n_2) \lll K_1,$$
$$K_2' = (K_2 \oplus n_1) \lll K_2 \text{ and } C = (K_1 \oplus K_2') + (K_1' \oplus K_2)$$

(4) Upon getting the message from R, T_j first XORs $(R_1\|R_1\|R_1)$ with the received value $(A\|B\|C) \oplus (R_1\|R_1\|R_1)$ to get $(A\|B\|C)$, and then extracts n_1 from A and n_2 from B. After that, T_j computes $K_1' = (K_1 \oplus n_2) \lll K_1$, $K_2' = (K_2 \oplus n_1) \lll K_2$ and $C' = (K_1 \oplus K_2') + (K_1' \oplus K_2)$. If C' does not match with the received value C, the session is terminated; otherwise, the reader R is authenticated and T_j calculates $D = (K_2' + ID) \oplus ((K_1 \oplus K_2) \vee K_1')$ which is soon transmitted to R. Meanwhile, T_j performs the updates: $IDS_{old} = IDS$, $IDS_{new} = (IDS + ID) \oplus (n_2 \oplus K_1')$, $K_{1\_old} =$

K_1, $K_{1\_new} = K_1'$, $K_{2\_old} = K_2$, $K_{2\_new} = K_2'$. After obtaining D, the reader R uses the secret values stored at S side to compute. $D' = (K_2' + ID) \oplus ((K_1 \oplus K_2) \vee K_1')$ and D' with D. If both them are identical, S updates $IDS = (IDS + ID) \oplus (n_2 \oplus K_1')$, $K_1 = K_1'$ and $K_2 = K_2'$; otherwise, the protocol is terminated.

• **De-synchronization Attack on YW09 Scheme [56]**

Given YW09 scheme and its relevant security parameter *sp*, the adversary Ad performs the following attack steps. Note that in session $i-1$ the secrets shared between T_j and S are synchronized, i.e., the secret at S side is $(IDS, K_1, K_2) = (IDS, K_1, K_2)_i$, and the secrets in T_j are $(IDS, K_1, K_2)_{old} = (IDS, K_1, K_2)_{i-1}$ and $(IDS, K_1, K_2)_{new} = (IDS, K_1, K_2)_i$.

The first phase (session i):

The adversary Ad first exploits the oracle query O_6 to compromise an arbitrary tag T_j and obtains the shared secret RID, where $l \neq j$. Ad then monitors the channel involved with the target tag T_j until a normal operation process of YW09 scheme between the reader R' and T_j is held. During the authentication procedure, Ad records $\{(A\|B\|C) \oplus (R_1\|R_1\|R_1)\}_i$ with the oracle query O_4 and intercept the message $\{D\}_i$ via the oracle query O_5. Now the secret at S side is $(IDS, K_1, K_2) = (IDS, K_1, K_2)_i$, and the secrets in T_j are $(IDS, K_1, K_2)_{old} = (IDS, K_1, K_2)_i$ and $(IDS, K_1, K_2)_{new} = (IDS, K_1, K_2)_{i+1}$.

The second phase (session $i+1$):

The adversary Ad monitors T_j's communication channel until a whole authentication session of YW09 scheme between another reader R'' and T_j is completed. Note that in this step (i.e., session $i+1$), $\{(A\|B\|C) \oplus (R_1\|R_1\|R_1)\}_{i+1}$ and $\{D\}_{i+1}$ are produced and based on $(IDS, K_1, K_2)_i$. As n_1 and n_2 are fresh at each session, $\{(A\|B\|C) \oplus (R_1\|R_1\|R_1), D\}_{i+1}$ is different from $\{(A\|B\|C\|) \oplus (R_1\|R_1\|R_1), D\}_i$. Since $(IDS)_{i+1}$ cannot be found in S side, the old tag pseudonym $(IDS)_i$ and corresponding record $(K_1, K_2)_i$ will be used to pass the legitimacy examination at R' side. Thus, the tag T_j will update its secrets $(IDS, K_1, K_2)_{old} = (IDS, K_1, K_2)_i$ and $(IDS, K_1, K_2)_{new} = (IDS, K_1, K_2)_{i+2}$ while the server S will update the shared secret $(IDS, K_1, K_2) = (IDS, K_1, K_2)_{i+2}$.

The third phase (session $i+2$):

Once the second step is done, the adversary Ad immediately selects the target tag T_j and invokes oracle query O_1 to obtain a session identifier $i+2$, a state information *st* and the challenge *Hello* message. The adversary Ad executes twice oracle O_2 operations to send *Hello* to T_j, and T_j responds $\{(IDS_i\|IDS_i) \oplus (RID\|R_{1\_i+2}), RID + R_{1\_i+2}\}$. In that case, Ad can derive the values R_{1_i+2} and $\{(A\|B\|C)_i \oplus (R_{1\_i+2}\|R_{1\_i+2}\|R_{1\_i+2})\}$ according to the values RID and R_1 obtained in step 1.

Ad then uses the oracle query O_2 to send $\{(A\|B\|C)_i \oplus (R_{1\_i+2}\|R_{1\_i+2}\|R_{1\_i+2})\}$ to T_j. Since $\{(A\|B\|C)_i \oplus (R_{1\_i+2}\|R_{1\_i+2}\|R_{1\_i+2})\}$ are involved with record $(IDS, K_1, K_2)_i$ and fresh pseudo random number R_{1_i+2}, $\{(A\|B\|C)_i \oplus (R_{1\_i+2}\|R_{1_i+2}\|R_{1_i+2})\}$ will be verified successfully by T_j. Now the secrets at T_j side are $(IDS, K_1, K_2)_{old} = (IDS, K_1, K_2)_i$ and $(IDS, K_1, K_2)_{new} = (IDS, K_1, K_2)_{i+3}$; however, the secret at S side is still $(IDS, K_1, K_2)_{old} = (IDS, K_1, K_2)_{i+2}$. As YW09 scheme adopts *key independent update*, the secrets shared between T_j and S is out of synchronization now.

Finally, Ad finishes the experiment and outputs a bit b' as its conjecture of the value of b from $Exp_{Ad}^{Availability}$. With the above procedures, Ad does make a correct guess of b, where 1, 3, 1, 1 and 1 execution times of O_1, O_2, O_4, O_5 and O_6 are required. The probability that $Ad(\varepsilon, t, 1, 3, 0, 1, 1, 1)$-break the availability of YW09 scheme is significant, i.e., $Adv(\varepsilon, t, 1, 3, 0, 1, 1, 1) = |Pr[Ad's\ guess\ is\ correct] - 50\%| = 50\%$, and the running time of Ad is polynomial. The insecurity of YW09 scheme is demonstrated.

3.4. **Important remarks.**

Remark 3.1. *As RFID authentication protocols [4,12,23,35-37,48,53] do not possess secret/key update mechanism, the forward/backward security cannot be guaranteed. Once the target tag T_j was compromised, the revealed secrets contained in T_j can be exploited by adversary to trace T_j's (previously involved and future) events or trajectories.*

Remark 3.2. *The RFID authentication schemes [10,13,16,26,27,39,41-43,45] possess the key update mechanism, but all of them lack the prevention scheme for de-synchronization attack. Malicious attacker can easily break the synchronization of secrets shared between the server and the tags via simple message interception.*

Remark 3.3. *The type III protocols [8,11,54,55] cannot guarantee the backward security as the updated key is always dependent on the currently involved key value. Even if a new session is invoked, the same updated key value will be derived.*

4. **Conclusion.**

Based on the proposed attack models, our two theorems have proved RFID authentication protocols involving with *key independent update* and *key redundancy design* cannot defend against de-synchronization attack. In addition, protocols categorized in type III or those being analyzed by other references [10,13,19,22,23,29,30, 35,38,40,46,52,55], cannot guarantee forward/backward security. In summary, our work shows that most existing authentication protocols cannot simultaneously provide forward/ backward security and resist de-synchronization attack in real world scenarios.

In this paper, we have introduced a formal definition of *authentication availability* and its relevant adversarial experiment. According to the definition, we have demonstrated that protocols categorized as types I and II are vulnerable to de-synchronization attack, and argue that most existing RFID authentication schemes cannot provide forward/backward security and defend against de-synchronization attack at the same time. We are the first one to introduce formal attack models analyzing RFID authentication protocols against de-synchronization attack. Our analyses indicate that *key independent update* and *key redundancy design* (i.e., to store both new and old secret values in database or tag) makes these RFID authentication schemes themselves difficult to support *authentication availability*. Any future extension of these protocols without modification on either *key independent update* or *key redundancy design* will incur the same identified authentication flaw. In the future, we plan to develop a robust framework with strong security and privacy to evaluate existing RFID authentication schemes, and propose a practical RFID authentication scheme with formal security proofs.

Acknowledgment. The authors gratefully acknowledge the support from Taiwan Information Security Center (TWISC) and National Science Council, Taiwan, under the Grants No. NSC 100-2219-E-011-002, NSC 100-2218-E-011-005 and NSC 100-2218-E-259-004-MY2. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] M. Akgun, M. U. Caglayan and E. Anarim, A new RFID authentication protocol with resistance to server impersonation, *IEEE International Symposium on Parallel & Distributed Processing*, pp.1-8, 2009.

[2] G. Avoine, E. Dysli and P. Oechslin, Reducing time complexity in RFID systems, *The 12th Annual Workshop on Selected Areas in Cryptography*, 2005.

[3] J. Ayoade, Security implications in RFID and authentication processing framework, *Computers & Security*, vol.25, no.3, pp.207-212, 2006.

[4] J. Bringer, H. Chabanne and E. Dottax, HB++: A lightweight authentication protocol secure against some attacks, *The 2nd International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing*, pp.28-33, 2006.
[5] M. Burmester and B. de Medeiros, The security of EPC Gen2 compliant RFID protocols, *The 6th International Conference of Applied Cryptography and Network Security, LNCS*, vol.5037, pp.490-506, 2008.
[6] S. Cai, Y. Li, T. Li and R. H. Deng, Attacks and improvements to an RFID mutual authentication protocol and its extensions, *The 2nd ACM Conference on Wireless Network Security*, Zurich, Switzerland, 2009.
[7] T. Cao, E. Bertino and H. Lei, Security analysis of the SASI protocol, *IEEE Transactions on Dependable and Secure Computing*, vol.6, pp.73-77, 2008.
[8] Y. Chen, J.-S. Chou and H.-M. Sun, A novel mutual authentication scheme based on quadratic residues for RFID systems, *Computer Networks*, vol.52, no.12, pp.2373-2380, 2008.
[9] H.-Y. Chien, SASI: A new ultralightweight RFID authentication protocol providing strong authentication and strong integrity, *IEEE Trans. on Dependable and Secure Computing*, vol.4, no.4, pp.337-340, 2007.
[10] H.-Y. Chien and C.-W. Huang, Security of ultra-lightweight RFID authentication protocols and its improvements, *ACM SIGOPS Operating System Review*, vol.41, pp.83-86, 2007.
[11] H.-Y. Chien and C.-H. Chen, Mutual authentication protocol for RFID conforming to EPC class 1 generation 2 standards, *Computer Standards & Interfaces*, vol.29, no.2, pp.254-259, 2007.
[12] E. Y. Choi, D. H. Lee and J. I. Lim, Anti-cloning protocol suitable to EPCglobal class-1 generation-2 RFID systems, *Computer Standards & Interfaces*, vol.31, pp.1124-1130, 2009.
[13] P. D'Arco and A. De Santis, From weaknesses to secret disclosure in a recent ultra-lightweight RFID authentication protocol, *Cryptology ePrint Archive*, 2008.
[14] I. Damgård and Ø. M. Pedersen, RFID security: Tradeoffs between security and efficiency, *Topics in Cryptology CT-RSA, LNCS*, vol.4964, pp.318-332, 2008.
[15] R. H. Deng, Y. Li, A. C. Yao, M. Yung and Y. Zhao, A new framework for RFID privacy, *Cryptology ePrint Archive, Report 2010/059*, 2010.
[16] T. Dimitriou, A lightweight RFID protocol to protect against traceability and cloning attacks, *SecureComm*, 2005.
[17] D. N. Duc, J. Park, H. Lee and K. Kim, Enhancing security of EPCglobal GEN-2 RFID tag against traceability and cloning, *Symposium on Cryptography and Information Security*, 2006.
[18] T. van Duersen and S. Radomirović, On a new formal proof model for RFID location privacy, *Cryptology ePrint Archive, Report 2008/477*, 2008.
[19] I. Erguler and E. Anarim, Scalability and security conflict for RFID authentication protocols, *IACR ePrint*, 2010.
[20] H. Gilbert, M. Robshaw and H. Sibert, An active attack against HB+ – A provably secure lightweight authentication protocol, *Cryptology ePrint Archive*, 2005.
[21] J. H. Ha, S. J. Moon, J. Zhou and J. C. Ha, A new formal proof model for RFID location privacy, *ESORICS, LNCS*, vol.5283, pp.267-281, 2008.
[22] D. Han and D. Kwon, Vulnerability of an RFID authentication protocol conforming to EPC class 1 generation 2 standards, *Computer Standards & Interfaces*, vol.31, no.4, pp.648-652, 2009.
[23] J. C. Hernandex-Castro, J. M. Estevex-Tapiador, P. Peris-Lopez and J.-J. Quisquater, Cryptanalysis of the SASI ultralightweight RFID authentication protocol, *ePrint arXiv: 0811.4257*, 2008.
[24] A. Juel and S. Weis, Defining strong privacy for RFID, *Cryptology ePrint Archive, Report 2006/137*, 2006.
[25] A. Juels and S. A. Weis, Authenticating pervasive devices with human protocols, *CRYPTO, LNCS*, vol.3621, pp.293-308, 2005.
[26] S. Karthikeyan and M. Nesterenko, RFID security without extensive cryptography, *The 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks*, pp.63-67, 2005.
[27] T. V. Le, M. Burmester and B. de Medeiros, Universally composable and forward-secure RFID authentication and authenticated key exchange, *The 2nd Asian ACM Symposium on Information, Computer and Communications Security*, pp.242-252, 2007.
[28] S. Lee, T. Asano and K. Kim, RFID mutual authentication scheme based on synchronized secret information, *Symposium on Cryptography and Information Security*, 2006.
[29] T. Li and R. H. Deng, Vulnerability analysis of EMAP – An efficient RFID mutual authentication protocol, *The 2nd International Conference on Availability, Reliability and Security*, pp.238-245, 2007.

[30] T. Li and G. Wang, Security analysis of two ultra-lightweight RFID authentication protocols, *IFIP International Federation for Information Security*, pp.108-120, 2007.
[31] N. W. Lo and K.-H. Yeh, Hash-based mutual authentication protocol for mobile RFID systems with robust reader-side privacy, *The 1st ACM Workshop on Convergence of RFID and Wireless Sensor Networks and Their Applications*, 2007.
[32] N. W. Lo and K.-H. Yeh, An efficient mutual authentication scheme for EPCglobal class-1 generation-2 RFID system, *The 2nd International Workshop on Trustworthiness, Reliability and Services in Ubiquitous and Sensor Networks, LNCS*, vol.4809, pp.43-56, 2007.
[33] N. W. Lo and K.-H. Yeh, Novel RFID authentication schemes for security enhancement and system efficiency, *The 4th VLDB Workshop on Secure Data Management, LNCS*, vol.4721, pp.203-212, 2007.
[34] N. W. Lo and K.-H. Yeh, Mutual RFID authentication scheme for resource-constrained tags, *Journal of Information Science and Engineering*, 2010.
[35] C. Ma, Y. Li, T. Li and R. H. Deng, RFID privacy: Relation between two notions, minimal condition, and efficient construction, *The 16th ACM Conference on Computer and Communication Security*, Chicago, IL, USA, pp.54-65, 2009.
[36] D. Molnar and D. Wagner, Privacy and security in library RFID: Issues, practices, and architectures, *Conference on Computer and Communications Security*, pp.210-219, 2004.
[37] J. Munilla and A. Peinado, HB-MP: A further step in the HB-family of lightweight authentication protocols, *Computer Networks*, vol.51, no.9, pp.2262-2267, 2007.
[38] C. Y. Ng, W. Susilo, Y. Mu and R. Safavi-Naini, New previacy results on synchronized RFID authentication protocols againse tag tracing, *ESORICS, LNCS*, vol.5789, pp.321-336, 2009.
[39] M. Ohkubo, K. Suzki and S. Kinoshita, Cryptographic approach to privacyfriendly tags, *The RFID Privacy Workshop*, 2003.
[40] K. Oua and R. C.-W. Phan, Privacy of recent RFID authentication protocols, *The 4th International Conference on Information Security Practice and Experience, LNCS*, vol.4991, pp.263-277, 2008.
[41] P. Peris-Lopez, J. C. Hernandex-Castro, J. M. Estevex-Tapiador and A. Ribagorda, LMAP: A real lightweight mutual authentication protocol for low-cost RFID tags, *The 2nd Workshop RFID Security*, 2006.
[42] P. Peris-Lopez, J. C. Hernandex-Castro, J. M. Estevex-Tapiador and A. Ribagorda, EMAP: An efficient mutual authentication protocol for low-cost tags, *OTM Federated Conferences and Workshop: IS Workshop*, 2006.
[43] P. Peris-Lopez, J. C. Hernandex-Castro, J. M. Estevex-Tapiador and A. Ribagorda, M2AP: A minimalist mutual-authentication protocol for low-cost RFID tags, *International Conference on Ubiquitous Intelligence and Computing, LNCS*, vol.4159, pp.912-923, 2006.
[44] P. Peris-Lopez, J. C. Hernandex-Castro, J. M. Estevex-Tapiador and A. Ribagorda, Advances in ultralightweight cryptography for low-cost RFID tags: Gossamer protocol, *The 9th International Workshop of Information Security Applications, LNCS*, vol.5379, pp.56-68, 2008.
[45] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador and A. Ribagorda, An ultra light authentication protocol suitable for resource-limited gen-2 RFID tags, *Journal of Information Science and Engineering*, vol.25, no.1, pp.33-57, 2009.
[46] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. E. Tapiador and J. C. A. van der Lubbe, Security flaws in a recent ultralightweight RFID protocol, *RFIDSec Asia*, 2010.
[47] P. Peris-Lopez, T. Li, J. C. Hernandez-Castro, Lightweight props on the weak security of EPC class-1 generation-2 standard, *IEICE Trans. Inf. & Syst.*, vol.E93-D, no.3, 2010.
[48] K. Rhee, J. Kwak, S. Kim and D. Won, Challenge-response based RFID authentication protocol for distributed database environment, *SPC, LNCS*, vol.3450, 2005.
[49] P. Rizomiliotis, E. Rekleitis and S. Gritzalis, Security analysis of the song-mitchell authentication protocol for low-cost RFID tags, *IEEE Communications Letters*, vol.13, no.4, pp.274-276, 2009.
[50] B. Song and C. J. Mitchell, RFID authentication protocol for low-cost tags, *The 1st ACM Conference on Wireless Network Security*, NY, USA, pp.140-147, 2008.
[51] B. Song and C. Mitchell, Scalable RFID authentication protocol, *Network System Security, IEEE Computer Society*, pp.216-224, 2009.
[52] H.-M. Sun, W.-C. Ting and K.-H. Wang, On the security of Chien's ultralightweight RFID authentication protocol, *Cryptology ePrint Archive, Report 83*, 2008.
[53] S. A. Weis, S. E. Sarma, R. L. Rivest and D. W. Engels, Security and privacy aspects of low-cost radio frequency identification systems, *Security in Pervasive Computing*, pp.201-212, 2003.
[54] J. Yang, J. Park, H. Lee, K. Ren and K. Kim, Mutual authentication protocol for low-cost RFID, *Encrypt Workshop on RFID and Lightweight Crypto*, 2005.

[55] K.-H. Yeh and N. W. Lo, Improvement of two lightweight RFID authentication protocols, *Information Assurance and Security Letters*, vol.1, pp.6-11, 2010.

[56] T. C. Yeh and C. S. Wu, An enhanced ultralightweight RFID authentication protocol, *Joint Conferences on Pervasive Computing*, pp.779-804, 2009.

[57] A. Juels, D. Molner and D. Wagner, Security and privacy issues in EPassports, *The 1st International Conference of Security and Privacy for Emerging Areas in Communication Networks, SecureComm*, 2005.

[58] P. Najera, J. Lopez and R. Roman, Real-time location and inpatient care systems based on passive RFID, *Journal of Network and Computer Applications*, vol.34, pp.980-989, 2011.

[59] C. C. Lo, C. H. Chen, D. Y. Cheng and H. Y. Kung, Ubiquitous healthcare service system with content-awareness capability: Design and implementation, *Expert Systems with Applications*, vol.38, pp.4416-4436, 2011.

[60] K. Ohashi, S. Ota, L. Ohno-Machado and H. Tanaka, Smart medical environment at the point of care: Auto-tracking clinical interventions at the bed side using RFID technology, *Computer in Biology and Medicine*, vol.40, pp.545-554, 2010.