The Study of RFID Authentication Protocols and Security of Some Popular RFID Tags

Hung-Yu Chien
Department of Information Management, National Chi Nan University, Taiwan, R.O.C.

1. Introduction

A radio frequency identification (RFID) system consists of three components: radio frequency (RF) tags (or transponders), RF readers (or transceivers), and a backend server. Tag readers *inquire* tags of their contents by broadcasting an RF signal, without physical contact, at a rate of several hundred tags per second and from a range of several meters. The advancements of Silicon manufacturing also result in great cost reduction for RFID tags compared to barcodes, not to mention that the tags can carry more data and are more resistant to dust and twisting. Thanks to these excellent features, the world has seen many RFID systems already put to use by manufacturers and businesses of all kinds of goods for supply management and inventory control and such; in addition, many public facilities and parking lots have also brought in RFID systems to help them offer faster, easier and more user-friendly services. As a matter of fact, potential applications are everywhere [57]. Such features as great convenience, low cost, and wide applicability will soon make RFID systems the most pervasive microchips in history [57].
However, the wide distribution of RFID systems into modern society may very much likely get the security of both businesses and consumers exposed to threats and risks. For example, businesses may have malicious competitors on the market that collect unprotected RFIDs to gather information illegally, spread false tags to provide wrong information, or even launch denial of service (DOS) attacks against them. On the other hand, as a consumer, it is naturally preferred that the information of the purchase of RFID-tagged products be kept private from outsiders; however, a tag reader at a fixed location can read the content of an un-protected tag, tracing the RFID-tagged product or/and even identifying the person carrying the tagged product. Correlating data collected from multiple tag readers such as their locations and so on can also possibly be used to spy on an individual and track down his/her social interactions. Besides passive eavesdropping and tracking, a thief might use counterfeit tags to fool automated checkout or security systems into accepting wrong information like price, proof of presence or other information.

RFID authentication protocols

To protect the private information on the RFID tags, some special devices (such as a blocker tag [26]) can be used here to deter the reader from accessing the tags, or tag authenticates the reader before its access. An RFID authentication protocol is a cryptographic protocol that

Source: Development and Implementation of RFID Technology, Book edited by: Cristina TURCU,
ISBN 978-3-902613-54-7, pp. 554, February 2009, I-Tech, Vienna, Austria

allows a reader and a tag to authenticate each other, and the protocol is especially suitable for cases where resource-limited RFID tags are involved. In fact, although there are high-cost RFID tags like [25] available on the market that can support conventional symmetric key computations or even public key computations, the mainstream tags targeted at the majority of consumers are low-cost and can only support simple computations and very limited storage [50]. For example, for such tags as Gen 2 [16, 58] or iso 15693, conventional authentication protocols that require symmetric key computations or even public key computations are not applicable. Therefore, most of the efforts both the businesses concerned and the academic community have made so far are focused on the research and development of low-cost tags with higher security levels. Therefore, the topic of the next section is authentication protocols that are designed for low-cost RFIDs. Please also note that since well-designed conventional cryptographic protocols can be effectively implemented on resource-abundant backend servers and readers, it is usually assumed that the channels between backend servers and readers are secure; however, now that the focus is on RFID authentication protocols, this study has to assume that the channel between tags and readers is insecure. Figure 1 shows the components of an RFID system.

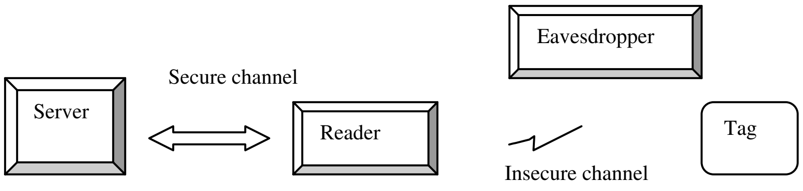

Fig. 1. Components of RFID systems

In addition, there are two special situations where the authentication of RFID tags is required to be done on extra conditions. To begin with, yoking proof protocols like [4, 7, 23, 24, 48, 53, 60] require the proof of simultaneous presence of two (or more) tags, and RFID distance bounding protocols like [5, 39, 56], on the other hand, not only authenticate the tags but also ensure that the authenticated tags are within a pre-assumed distance from the verifiers (the readers) so that the system is immune to message relay attacks like those brought up by [56]. In the following paragraphs, we shall briefly introduce yoking proof protocols and RFID distance bounding protocols. For detailed information, please refer to [4, 5, 7, 23, 24, 39, 48, 53, 56, 60].

Yoking proof

In 2004, Juels introduced an interesting RFID yoking proof protocol [23], which allows a verifier to prove the simultaneous presence of two tags in the communication range of a specific reader. Juels proposed several possible yoking proof protocol applications [23]. Let us take one example. Suppose a hard disk manufacturer wishes to ship each hard disk with its information leaflet. In such a case, each hard disk and each leaflet can be labeled with a different tag so that the yoking protocol can be applied to prove the simultaneous presence of the tagged products before shipping. In fact, the yoking proof protocol is a variant of the cryptographic authentication protocol, and it additionally requires the evidence of the simultaneous presence of two tags (or more tags).

RFID distance bounding protocols

Due to the short communication range, an authenticated RFID tag is deemed to be in proximity by its verifier (for example, an RFID reader), and the security of many RFID applications depends on this proximity assumption. However, this belief of proximity could be maliciously manipulated and thus become misleading when relay attacks like [56] are launched. For example, the access control system of a building would allow the access only when an authenticated tag is in the proximity. However, a specific kind of relay attack named the mafia attack, introduced by Desmedt [14], could cheat the system where an attacker sets up a rogue tag (say $\hat{A}$) and a rogue reader (say $\hat{B}$) sitting between the real reader and the real tag, and $\hat{A}$ and $\hat{B}$ cooperatively relay the messages between the real tag and the real reader so that the real reader wrongly believes that the tag is in its proximity (but it is not). A distance bounding protocol is a cryptographic mechanism that can prevent relay attacks from working. It is executed by a tag A and a reader B, and the tag A can convince the reader B of A's identity and A's physical proximity to B.

2. RFID authentication protocols

An RFID authentication protocol provides mutual authentication between the reader and the tag, and should resist potential security threats and attacks like the replay attack, man-in-the middle attack, etc. In addition to mutual authentication, anonymity and forward secrecy are also desirable properties for RFIDs. The point of ensuring the system's anonymity is to protect the privacy of the tags' identities such that un-authorized readers cannot identify or track a specific tag. Forward secrecy property, on the other hand, aims to protect the past communications where a tag is involved even if we assume that an attacker may have the power to compromise the tag some time later [50].

Just like tags of variant kinds currently available on the market, RFID authentication protocols can be quite different from one another, and the differences may come from the distinct resources required or the varied mechanisms adopted. Accordingly, we can classify these protocols and specify the features each kind has. Following the classification brought up by [52], for example, a protocol can be either a single-round design or a multi-round system. The former allows the reader and the tag to authenticate each other after a single round of operation of the protocol, while the latter has to run multiple rounds to do the job. Generally speaking, a single round protocol is more efficient than a multi-round protocol in terms of the number of interactions. Another classification, proposed by Chien [11], is based on the resources demanded by the protocols. This classification is very practical, because as we said earlier, on the market there are varieties of tags, of which most are resource-limited, and the resources required by these protocols can be very different. Under such circumstances, of course we will have a better view of the whole market if we classify the protocols and tags according to what kinds of resources are required. A third classification is based on the kind of cryptographic approach adopted, for the approach decides how well the protocol performs. Section 2.1 classifies the protocols as either single-round methods or multi-round methods, reviews the protocols and discusses the security properties. In Section 2.2, according to the required resources, we classify the protocols into four classes and introduce their corresponding applications. Finally, based on the cryptographic approaches, Section 2.3 classifies the protocols and discusses their performance.

2.1 RFID authentication protocols

Some single-round protocols are introduced in Section 2.1.1~2.1.6, while multi-round protocols are introduced in Section 2.1.7. Even though tags' data and keys are stored in the backend server in most of the cases, we do not differentiate the role of backend sever and the reader to simplify the description in the following sections. The notations used are introduced as follows.

r, r_T, r_R : l-bit random numbers.

ID_T, ID_R : the identity of tag *T*, the identity of reader *R*.

k_i : the secret key shared between tag T_i and the reader *R*.

$h()$, $g()$: secure one-way hash function; $h()$, $g()$: $\{0,1\}^* \rightarrow \{0,1\}^l$.

$CRC()$: cyclic redundancy code.

$f()$: a pseudo random number generator (PRNG function).

2.1.1 Weis et al.'s schemes

Weis et al. proposed a series of RFID authentication protocols [63, 64], and we review their hash-based access control protocol and the randomized access control.

Hash-based access control: Each hash-enabled tag T_i in this design will have a portion of memory reserved for a temporary $metaID_i$ and will operate in either a locked state or an unlocked state. Initially, a tag owner stores the hash of a random key, $metaID_i \leftarrow h(k_i)$, in the tag through either the RF channel or a physical contact to lock the tag. The owner also stores both the key and the $metaID_i$ in a backend server. Upon receipt of a $metaID_i$ value, the tag enters its locked state, and responds to all queries with only its $metaID_i$ and offers no other functionality. To unlock a tag, the owner inquires the tag, looks up the appropriate key in the back-end database and finally transmits the key to the tag. The tag hashes the received key and compares it to the stored $metaID_i$. If the values match, the tag unlocks itself and offers its full functionality to any nearby readers. The protocol is depicted in Figure 2.

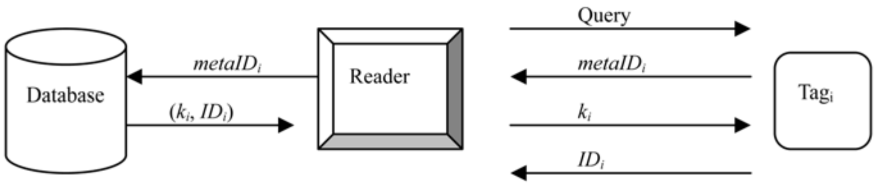

Fig. 2. Weis et al.'s Hash-based scheme: unlocking protocol

Randomized access control: In the previous scheme, a tag always responds with its $metaID_i$ to the queries, which allows any party to track an individual. So, Weis et al. proposed their randomized access control schemes where a tag will not respond predictably to queries by unauthorized users, but must still identifiable by only legitimate readers. The randomized access control schemes require tags equipped with a random number generator, in addition to the one-way hash function. Upon receiving a query from the reader, a tag responds with the values $(r, h(ID_i \| r))$, where r is a randomly chosen number. A legitimate reader identifies one of its tags by performing a brute-force search of its known IDs, hashing each of them concatenated with *r* until it finds a match. This mode is only feasible for owners of a relatively small number of tags. The protocol is depicted in Fig. 3.

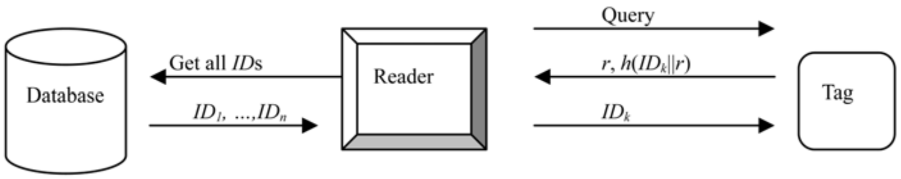

Fig. 3. Weis et al.'s randomized access control

Weakness of the hash-based scheme: In Figure 2, the reader broadcasts the tag's key in the forward channel. Since the signal in forward channel is strong enough for an adversary to monitor the transmission without being detected, this will allow an adversary easily eavesdrop the key and spoof a legal reader later.

Weaknesses of the random-access scheme: The Random-access scheme was designed to protect the *metaID* in the hash-based scheme to avoid individual tracking. However, it has poor scalability: it cannot support a large volume of tags because it has to perform the brute-force search to find a matched ID. It also gives the adversary (who resides in the range of the backward channel) a very high probability to find the matched tag, since he also searches only a small database of possible IDs. What makes it worse: the legal reader will broadcast the matched ID in the forward channel. So, an adversary might record the eavesdropped data ($r, h(ID_k \| r)$) and then easily spoofs the tags later.

2.1.2 Ohkubo et al.'s scheme [43]

The reader and each tag T_x initially shares a distinct hash seed s_{1_x}. T_x updates $s_{i+1\_x} = h(s_{i\_x})$ for $i \geq 1$ and responds with $a_{i\_x} = g(s_{i\_x})$ in the *i*-th authentication, where $h()/g()$ are two different hash functions. The reader can follow the hashing chains to authenticate the tag. The protocol is depicted in Fig. 4.

This scheme provides only one-way authentication of the tag, but it owns the forward secrecy property; that is, even assuming a tag is compromised some day in the future, the past communications from the same tag can not be traced. However, Ohkubo et al.'s original version cannot resist the replay attack [1]- a simple replay of old message can cheat the reader into accepting a forged tag. The scheme has the poor scalability problem [2, 3] – the computational cost to identify a tag is O(*nm*), where *n* is the number of potential tags and *m* is the maximum length of the hash chain. Avoine et al. [1] discussed the techniques to conquer the replay attack, and Avoine et al. [1, 2] also proposed their improvements to reduce the time complexity at the cost of extra memory.

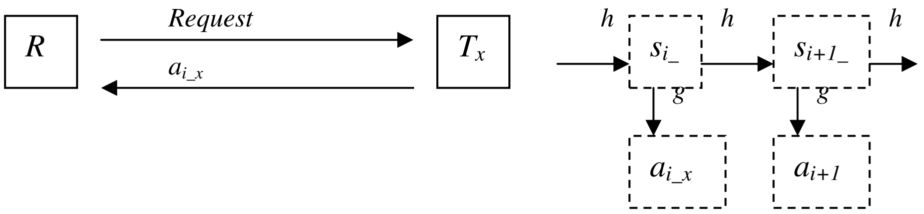

Fig. 4. Ohkubo et al.'s scheme

2.1.3 Karthikeyan-Nesterenko's scheme [28]

Karthikeyanand and Nesterenko, based on simple XOR operation, $\oplus$, and matrix operation, designed an efficient tag identification and reader authentication scheme. Initially, two matrices M_1 and M_2^{-1} are stored on each tag, and two matrices M_2 and M_1^{-1} are stored on the reader, where all the matrices are of size $p \times p$, and M_1^{-1} and M_2^{-1} are the inverses of M_1 and M_2 respectively. The tag and the reader also store a key *K* which is a vector of size *q*, where *q*=*rp*. That is, *K* can be represented as $K=[K_1, K_2, \ldots, K_r]$, where $K_i, i = 1,2...,r$ are vectors of size p. As a slight abuse of notation, the notation *X*=*KM*, where *K* is a vector of size *q* and *M* is a $p \times p$ matrix, denotes a component-wise multiplication of *K* and *M*. That is, $X=[X_1, \ldots, X_r]=[K_1M, \ldots, K_rM]$.

When the reader inquires a tag, the tag computes $X = KM_1$, and sends back X to the reader. The reader then forwards the message to the backend server, where the server will search its database to find a match. If it can find a match, then the tag is identified, and the server performs the following operations to authenticate itself to the tag and renew the key. The server first computes $Y = (K_1 \oplus K_2 \oplus ... \oplus K_r)M_2$, randomly selects a vector X_{new} of size *q*, computes $K_{new} = X_{new}M_1^{-1}$ and $Z = K_{new}M_2$, and finally sends (Y, Z) to the reader, which forwards (Y, Z) to the tag. Upon receiving the response from the reader, the tag verifies whether the equation $YM_2^{-1} \overset{?}{=} (K_1 \oplus K_2 \oplus ... \oplus K_r)$ holds; if so, the tag updates the key as $K_{new} = ZM_2^{-1}$. The scheme is depicted in Fig. 5.

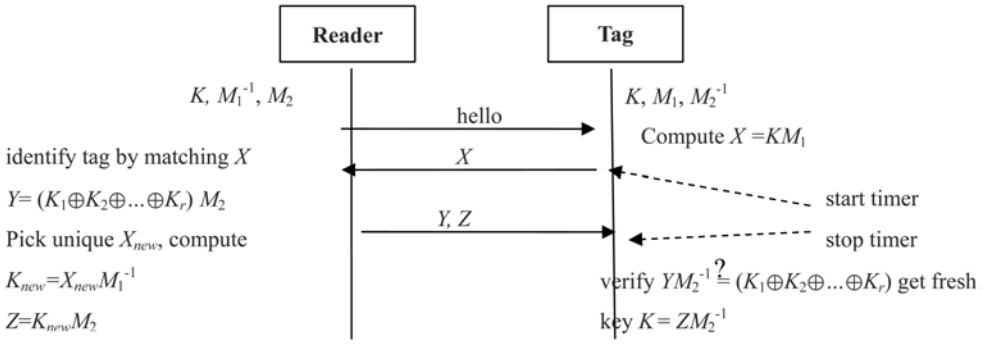

Fig. 5. Karthikeyan-Nesterenko's scheme

Weaknesses of Karthikeyan-Nesterenko's scheme

The scheme cannot resist the following attacks and threats- Denial of Services attack (DOS), replay attack and individual tracing.

In Karthikeyan-Nesterenko's scheme, the tag does not authenticate the received value *Z* when updating the key. Therefore, an attacker can replace the transmitted *Z* with an old one $\overline{Z}$ or any random value *Z** without being noticed; Upon receiving a valid *Y* and the fake *Z**, the tag will authenticate the *Y* successfully and then will update the key as $K^* = M_2^{-1} \cdot Z^*$. So, the legitimate reader and the tag cannot authenticate each other any more since the key is wrongly updated.

If the attacker replaces the *Z* with an old one $\overline{Z}$ (assuming $\overline{Y}$ and $\overline{Z}$ are previously sent in the *i*th legal session) in the above mentioned attack, then the attacker can replay the $\overline{Y}$ in the next session to cheat the tag in wrongly accepting the request and access the tag accordingly. He can even record the transmitted data from several sessions, and then launches the above attack several times. This will allow the attacker to trace the tag. Therefore, the anonymity property is violated.

2.1.4 Duc et al.'s scheme [15]

Duc et al.'s scheme was designed for improving the security of EPCglobal Class-1 Generation-2 tag (which is called Gen-2 for short later). Initially, each tag and the backend server share the tag's EPC code (the identity of the tag), the tag's *access* PIN, and an initial key K_0 (this key will be updated after each successful authentication, and K_i denotes the key after *i*th authentication). The steps of (*i*+1)th authentication are described as follows, where "Reader → tag: *M*" denotes the reader sends the tag a message *M*.

1. Reader → tag: *Query request*.
2. Tag → reader → server: M_1, r, C.

The tag selects a random number r, computes $M_1 = CRC(EPC \| r) \oplus K_i$ and $C = CRC(M_1 \oplus r)$, and sends back (M_1, r, C) to the reader, where the reader will forward (M_1, r, C) to the backend server.

3. Server → reader: the tag's info or "*failure*".

For each tuple (EPC, K_i) in its database, the server verifies whether the equations $M_1 \oplus K_i \overset{?}{=} CRC(EPC \| r)$ and $C \overset{?}{=} CRC(M_1 \oplus r)$ hold. If it can find a match, then the tag is successfully identified and authenticated, and the server will forward the tag's information to the reader and proceed to the next step; otherwise, it stops the process with failure.

4. Server → Reader → tag: M_2

To authenticate itself to the tag and update the information on the tag, the server computes $M_2 = CRC(EPC \| PIN \| r) \oplus K_i$ and sends M_2 to the tag through the reader. Upon receiving M_2, the tag uses its local values to verify the received M_2. If the verification succeeds, the tag will accept the "end session" command in the next step.

5. Reader → tag: "end session"
   Reader → server: "end session".

- Upon receiving the "end session" command, both the server and the tag update their shared key as $K_{i+1} = f(K_i)$.

The weaknesses

Duc et al.'s scheme cannot resist the DOS attack against tags and readers, cannot detect the disguise of tags, and cannot provide forward secrecy.

(1) In the last step of Duc et al.'s scheme, the reader sends the "end session" commands to both the tag and the backend server to update the key. If one of the "end session" commands is intercepted, then the shared key between the tag and the server will be out of synchronization. Thus, the tag and the reader cannot authenticate each other any more. The DOS attack succeeds. (2) If it is the "end session" command to the server is intercepted, then the server will hold the old key; therefore, a counterfeit tag can replay the old data (M_1, r, C) to disguise as a legitimate tag. So, the scheme fails to detect a disguised tag. (3) The scheme cannot provide forward secrecy. Suppose a tag is compromised, then the attacker would get the values (EPC, PIN, K_i) of the tag ; So, from the eavesdropped data (M_1, M_2, r) of the

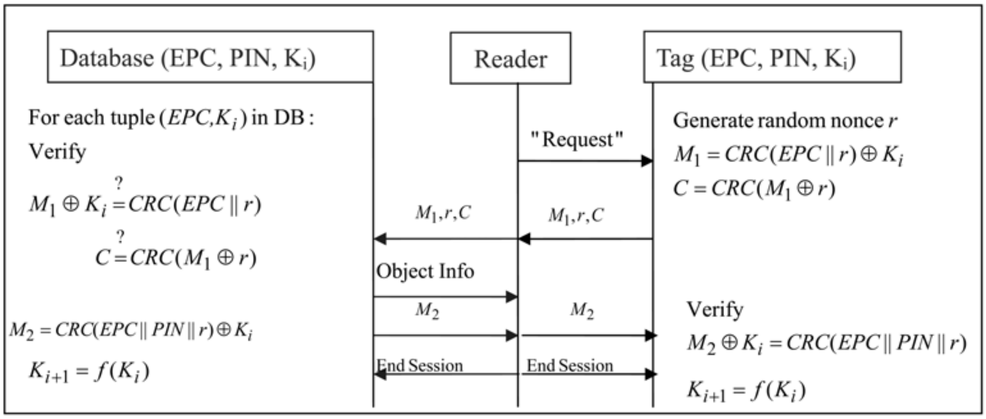
Database (EPC, PIN, K_i)
Reader
Tag (EPC, PIN, K_i)
For each tuple (EPC, K_i) in DB:
Verify
$M_1 \oplus K_i \overset{?}{=} CRC(EPC \| r)$
$C \overset{?}{=} CRC(M_1 \oplus r)$
$M_2 = CRC(EPC \| PIN \| r) \oplus K_i$
$K_{i+1} = f(K_i)$
"Request"
M_1, r, C
M_1, r, C
Object Info
M_2
M_2
End Session
End Session
Generate random nonce r
$M_1 = CRC(EPC \| r) \oplus K_i$
$C = CRC(M_1 \oplus r)$
Verify
$M_2 \oplus K_i = CRC(EPC \| PIN \| r)$
$K_{i+1} = f(K_i)$

Fig. 6. Due et al. scheme

past communications, the attacker can verify whether a communication comes from the same tag by performing the following checking. For each eavesdropped communication (M_1, M_2, r), he computes $M_1 \oplus M_2$ to derive the value $CRC(EPC \oplus r) \oplus CRC(EPC \| PIN \| r)$, and then, using the compromised values (EPC, PIN, K_i) and the eavesdropped r, he can do the same computation to verify whether it came from the same tag. So, the past communications of a compromised tag can be traced.

2.1.5 Peris-Lopez et al.'s protocols [45-47]

Peris-Lopez et al. proposed a series of ultra-lightweight RFID authentication protocols [45-47] which were designed for very low-cost tags. Their schemes were very efficient: they require about 300 gates only and involve only simple bitwise operations. We review the LAMP protocol [45], which is one of Peris-Lopez et al.'s ultra-lightweight protocols.

LMAP involves only simple bitwise operations- bitwise XOR ($\oplus$), bitwise AND ($\wedge$), bitwise OR ($\vee$), and addition mod 2^m (+). The random number generator is only required on the reader. To protect the anonymity of tags, they adopt the technique of pseudonyms (*IDS*s), which is 96-bit length and is updated per successful authentication. Each tag shares an *IDS* and four keys (called *K1, K2*, *K3*, and *K4*, each with 96 bits) with readers, and they update the *IDS* and the keys after successful authentication. It needs 480 bits of rewritable memory and 96 bits for static identification number (*ID*).

The protocols consist of three stages- tag identification phase, mutual authentication phase, and pseudonym updating and key updating phase. In the following, ID_i denotes the static identification of Tag$_i$, IDS_i^n denotes the pseudonym of Tag$_i$ at the *n*-th run, and $K1_i^n / K2_i^n / K3_i^n / K4_i^n$ denote the four keys of Tag$_i$ at the *n*-th run. LMAP is depicted in Fig. 7.

Tag identification: Initially, the reader sends "*hello*" to probe Tag$_i$, which responds with its current IDS_i^n.

Mutual authentication phase: the reader uses IDS_i^n to find the corresponding four keys in its database, via the help of the backend server. It then randomly selects two integers *n1* and *n2*, and computes the values *A*, *B*, and *C* (the calculation equations are specified in Fig. 7). From *A*||*B*||*C*, Tag$_i$ first extracts *n1* from *A*, and then verifies the value of *B*. If the verification succeeds, then it extracts *n2* from *C*, and computes the response value *D*. Upon receiving *D*, the reader verify the data *D* to authenticate the tag.

Pseudonym updating and key updating: After the reader and the tag authenticated each other, they update their local pseudonym and keys as specified in Fig. 7.

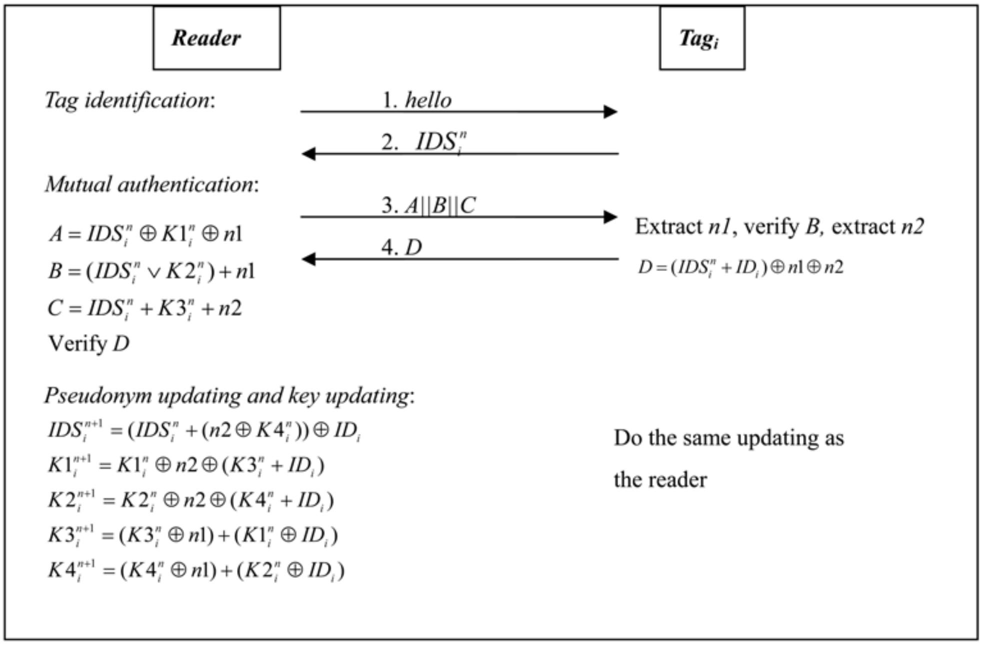
Reader
Tag$_i$
Tag identification:
1. hello
2. IDS_i^n
Mutual authentication:
3. $A||B||C$
4. D
$A = IDS_i^n \oplus K1_i^n \oplus n1$
$B = (IDS_i^n \vee K2_i^n) + n1$
$C = IDS_i^n + K3_i^n + n2$
Verify D
Extract $n1$, verify B, extract $n2$
$D = (IDS_i^n + ID_i) \oplus n1 \oplus n2$
Pseudonym updating and key updating:
$IDS_i^{n+1} = (IDS_i^n + (n2 \oplus K4_i^n)) \oplus ID_i$
$K1_i^{n+1} = K1_i^n \oplus n2 \oplus (K3_i^n + ID_i)$
$K2_i^{n+1} = K2_i^n \oplus n2 \oplus (K4_i^n + ID_i)$
$K3_i^{n+1} = (K3_i^n \oplus n1) + (K1_i^n \oplus ID_i)$
$K4_i^{n+1} = (K4_i^n \oplus n1) + (K2_i^n \oplus ID_i)$
Do the same updating as the reader

Fig. 7. LMAP

The weaknesses

The authentication of reader and tag in LMAP depends on the synchronization of pseudonym and keys. However, it is very easy to de-synchronize these values by intercepting the data in Step 4. In addition to the DOS attack, one can fully disclose the secrets of tags as follows.

We assume that an attacker can intercept, modify, and replay message between reader and tag in a reasonable time, and there is a completion message to indicate the completion of successful authentication. The attack scenario consists of five phases, but our attack is much more efficient than Li-Wang's work [34]. The whole scenario is depicted in Fig. 8.

In the attack scenario in Fig. 8, we omit the superscript *n* and the subscript *i* of pseudonym and of keys without causing ambiguity, since we are attacking the same tag within a successful session. In Phase 1, an attacker impersonates a reader and acquires the current *IDS* of a tag, and then the attacker (now impersonating the tag) uses the *IDS* to get a valid message *A*||*B*||*C* from the reader in Phase 2.

In Phase 3, the attacker iteratively inverts the *j*-th (for $1 \le j \le 96$) bit of *A*, modifies *B*, and sends $A_j^{'} \| B_j^{'} \| C$ to the tag. From the tag's response (which is either a message *D* or an error message), the attacker can derive the *j*-th bit of *n1*. After deriving the value of *n1*, it further derives the values of *K1* and *K2* from *A*, *B*, *IDS* and *n1*. The detail of deriving the *j*-th bit is as follows. Let $A_j^{'}$ denotes the value by inverting the *j*-th bit of *A*. If the tag receives $A_j^{'}$, then it will derive $n_1^{'}$, which is equal to either $n1+2^{j-1}$ or $n1-2^{j-1}$, and each of the cases is with probability 1/2. So, the attacker can assume $n_1{'}=n_1+2^{j-1}$, computes $B_j^{'}=B+2^{i-1}$, and sends $A_j^{'} \| B_j^{'} \| C$ to the tag. After receiving $A_j^{'} \| B_j^{'} \| C$, the tag extracts $n_1^{'}$ from $A_j^{'}$, verifies $B_j^{'}$, and then responds with either a message D_j or an error message. If a proper D_j is returned, the attacker can conclude that $n_1{'}=n_1+2^{j-1}$ and $n1[j]=0$ ($n1[j]$ denotes the *j*-th bit of *n1*); otherwise, it concludes that $n1[j]=1$. With this technique, the attacker launches 96 runs to derive all the bits of n_1, and then solves the values of *K1* and *K2* accordingly. Now the rest is to derive the values of *n2*, *K3*, *K4* and *ID*.

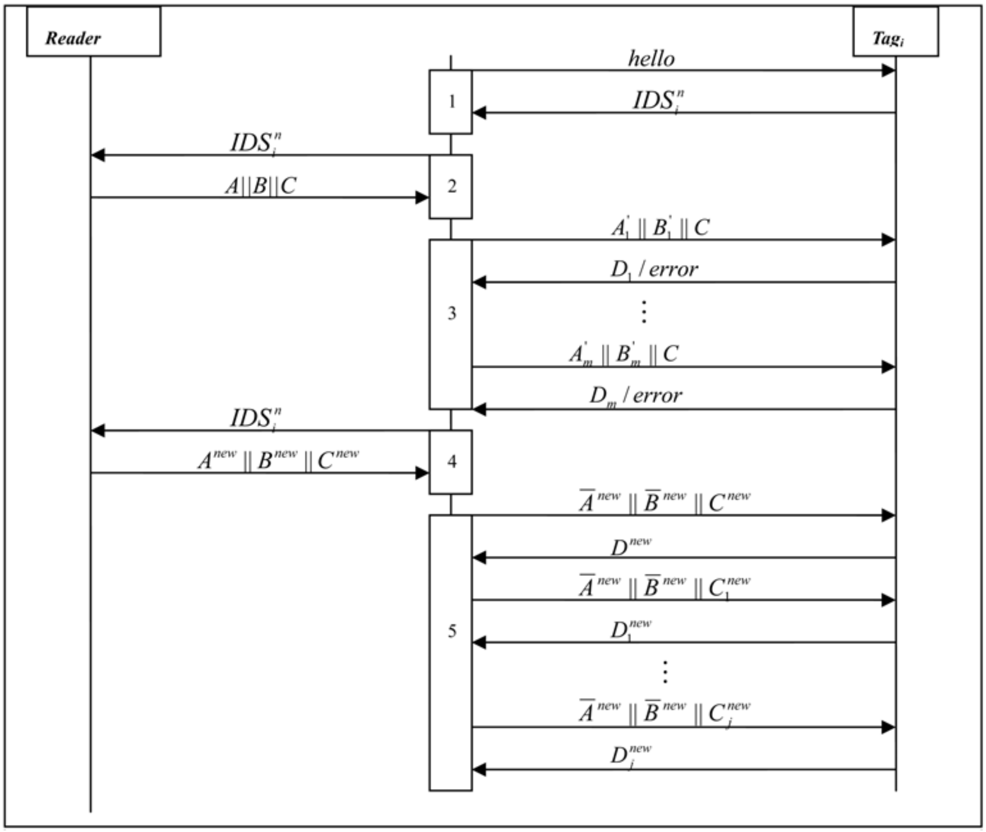
Reader
Tag$_i$
hello
1
IDS_i^n
IDS_i^n
$A\|B\|C$
2
$A_1^{'} \| B_1^{'} \| C$
D_1 / error
3
$A_m^{'} \| B_m^{'} \| C$
D_m / error
IDS_i^n
$A^{new} \| B^{new} \| C^{new}$
4
$\overline{A}^{new} \| \overline{B}^{new} \| C^{new}$
D^{new}
$\overline{A}^{new} \| \overline{B}^{new} \| C_1^{new}$
D_1^{new}
5
$\overline{A}^{new} \| \overline{B}^{new} \| C_j^{new}$
D_j^{new}

Fig. 8. Full-disclosure attack on LMAP

In Phase 4, the attacker impersonates the tag to the reader to get a new response $A^{new} \| B^{new} \| C^{new}$.

In phase 5, since the values of *IDS*, *K1*, and *K2* are already known, the attacker first sets $\overline{n1}^{new}$=0 to have $\overline{A}^{new} = IDS \oplus K1$ and $\overline{B}^{new} = IDS \vee K2$, and sends $\overline{A}^{new} \| \overline{B}^{new} \| C^{new}$. So, the tag will respond with $D^{new} = (IDS + ID) \oplus n2$. Next, the attacker sets $C_1^{new} = C^{new} + 1$, and sends $\overline{A}^{new} \| \overline{B}^{new} \| C_1^{new}$ to the tag, which will extract $n2+1$ and will respond with $D_1^{new} = (IDS + ID) \oplus (n2+1)$. Now we have the equation $n2 \oplus (n2+1) = D^{new} \oplus D_1^{new}$. The possible values of $D^{new} \oplus D_1^{new}$ are summarized in Table 1. From Table 1, we can see that (1) if $D^{new} \oplus D_1^{new}$ has the form 0…01, then *n*2[1]=0; (2) if $D^{new} \oplus D_1^{new}$ has *i*+1 1s on the right and the rest are 0s, then *n2* has *i* 1s from the right and followed by a zero. So, two simple interactions with the tag, the attacker can determine *i*+1 ($i \in [0,95]$) bits of *n2*. Following that, the attacker sets $C_{i+2}^{new} = C^{new} + 2^{i+1}$, and sends $\overline{A}^{new} \| \overline{B}^{new} \| C_{i+2}^{new}$. After getting the response D_{i+2}^{new}, the attacker computes $D^{new} \oplus D_{i+2}^{new}$ to determine the next few bits. It repeats this process until all the 96 bits of *n2* are solved. This phase takes 2 interactions in the best case and 96 interactions in the worst case. After deriving *n2*, the attacker can further solve *K3* and *ID* from the data *C* and *D*. With two successive pseudonyms IDS_i^n and IDS_i^{n+1}, the attacker further derives *K4* .

If *n*2[1]=0 (that is, *n*2 has the form xxxx….x0)*	then $n2 \oplus (n2+1) = 000...01$
If *n*2[1]=1 and *n*2 has the form xxx01…1 (that is, *n*2 has *i* 1s from the right followed by a 0)	then $n2 \oplus (n2+1) = 0...01...1$ (that is, $n2 \oplus (n2+1)$ has *i*+1 1s on the right and the rest are 0s)

*x denote the bit value is either 0 or 1.

Table 1. The possible values of $D^{new} \oplus D_1^{new}$

For more details of weaknesses of Peris-Lopez et al.'s ultra-lightweight protocols [45-47], one can refer to [13, 33-35].

2.1.6 Chien's SASI protocol [11]

Chien's SASI was designed for very low-cost RFID tags. Each tag has a static identification (*ID*), and pre-shares a pseudonym (*IDS*) and two keys *K1*/*K2* with the backend server. The length of each of *ID*/*IDS*/*K1*/*K2* is 96 bits. To resist the possible de-synchronization attack, each tag actually keeps two entries of (*IDS*, *K1*, *K2*): one is for the *old* values and the other is for the *potential next* values. The protocol consist of three stages- tag identification phase, mutual authentication phase, and pseudonym updating and key updating phase. In each protocol instance, the reader may probe the tag twice or once in the tag identification phase, depending on the tag's *IDS* is found or not. The reader first sends "*hello*" message to the tag, and the tag will respond with its *potential* next *IDS*. The reader uses the tag's response *IDS* to find a matched entry in the database, and goes to the mutual authentication phase if a matched entry is found; otherwise, it probes again and the tag responds with its old *IDS*. In the mutual authentication phase, the reader and the tag authenticate each other, and they respectively update their local pseudonym and the keys after successful authentication. After successful authentication, the tag stores the matched values to the entry ($IDS_{old} \| K1_{old} \| K2_{old}$) and stores the updated values to the entry ($IDS_{next} \| K1_{next} \| K2_{next}$). The random number generator is required on the reader only, and the tags only involve simple bit-wise operations like bitwise XOR ($\oplus$), bitwise OR ($\vee$), bitwise AND ($\wedge$), addition mod

2^m (+), and left rotate ($Rot(x,y)$). $Rot(x,y)$ is defined to left rotate the value of x with y bits. The protocol procedures are described as follows.

Tag identification: Initially, the reader sends "*hello*" to the tag, which first responds with its potential next *IDS* . If the reader could find a matched entry in the database, it steps into the mutual authentication phase; otherwise, it probes again and the tag responds with its old *IDS*.

Mutual authentication phase: the reader uses *IDS* to find a matched record in the database. It could be the potential next *IDS* or the old *IDS* of the tag. It then uses the matched values and two generated random integers *n1* and *n2* to compute the values *A*, *B*, and *C* (the calculation equations are specified in Fig. 9). From *A*||*B*||*C*, the tag first extracts *n1* from *A*, extracts $n2$ from *B*, computes $\bar{K}1$ and $\bar{K}2$ and then verifies the value of *C*. If the verification succeeds, then it computes the response value *D*. Upon receiving *D*, the reader uses its local values to verify *D*.

Pseudonym updating and key updating: After the reader and the tag authenticated each other, they update their local pseudonym and keys as specified in Fig. 9. The scheme also provides confirmation of the synchronization values ($\bar{K}1, \bar{K}2$) when the reader and the tag successfully authenticate each other.

The weaknesses

Sun et al. [62] had noticed that SASI is still vulnerable to DOS attacks. One attack scenario is described as follows. Assume that there is a synchronized tag T_x in which ($IDS_{next}, K1_{next}, K2_{next}$) equals to ($IDS_1, K1_1, K2_1$) stored in the database. Now, suppose the reader probes the tag, and sends out (A', B', C') , which is eavesdropped by the attacker. At the end of the protocol, the attacker interrupts the message *D* so that the reader will not update its variables. However, the tag will update its variables as follows: a) ($IDS_{old}, K1_{old}, K2_{old}$)=($IDS_1, K1_1, K2_1$), b) ($IDS_{next}, K1_{next}, K2_{next}$) =($IDS_2, K1_2, K2_2$).

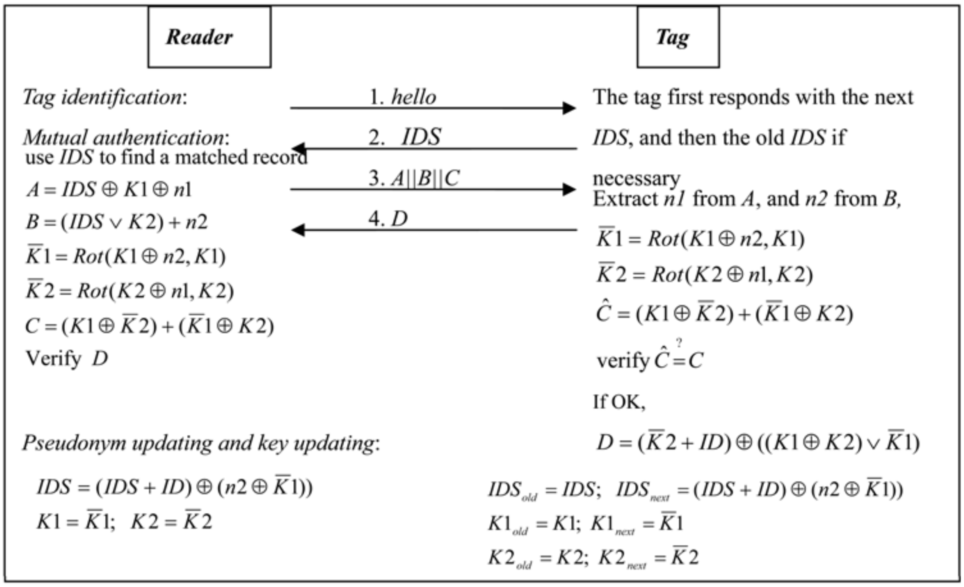
Reader
Tag
Tag identification:
1. hello
The tag first responds with the next IDS, and then the old IDS if necessary
Mutual authentication:
use IDS to find a matched record
2. IDS
3. A||B||C
4. D
$A = IDS \oplus K1 \oplus n1$
$B = (IDS \vee K2) + n2$
$\bar{K}1 = Rot(K1 \oplus n2, K1)$
$\bar{K}2 = Rot(K2 \oplus n1, K2)$
$C = (K1 \oplus \bar{K}2) + (\bar{K}1 \oplus K2)$
Verify D
Extract n1 from A, and n2 from B,
$\bar{K}1 = Rot(K1 \oplus n2, K1)$
$\bar{K}2 = Rot(K2 \oplus n1, K2)$
$\hat{C} = (K1 \oplus \bar{K}2) + (\bar{K}1 \oplus K2)$
verify $\hat{C} \overset{?}{=} C$
If OK,
$D = (\bar{K}2 + ID) \oplus ((K1 \oplus K2) \vee \bar{K}1)$
Pseudonym updating and key updating:
$IDS = (IDS + ID) \oplus (n2 \oplus \bar{K}1))$
$K1 = \bar{K}1;\ K2 = \bar{K}2$
$IDS_{old} = IDS;\ IDS_{next} = (IDS + ID) \oplus (n2 \oplus \bar{K}1))$
$K1_{old} = K1;\ K1_{next} = \bar{K}1$
$K2_{old} = K2;\ K2_{next} = \bar{K}2$

Fig. 9. SASI protocol

Next, the attacker allows the reader and the tag to run the protocol again without intervening. Because IDS_2 is not found in the database, both the reader and the tag use IDS_1 to complete the authentication. Thus, the database will update its variable list to ($IDS_3, K1_3, K2_3$), but the tag would own the values ($IDS_1, K1_1, K2_1$) and ($IDS_3, K1_3, K2_3$).

Finally, the attacker imitates as a valid reader to probe the tag. The tag first replies $IDS_{next} = IDS_3$, the attacker ignores this reply, which triggers the tag to reply IDS_1. The attacker now replays the recorded message (A', B', C'), which is valid and the tag would update its values a) ($IDS_{old}, K1_{old}, K2_{old}$)=($IDS_1, K1_1, K2_1$), b) ($IDS_{next}, K1_{next}, K2_{next}$) =($IDS_2, K1_2, K2_2$). Now, the genuine reader and the tag are out-of-synchronization. Sun et al. had shown another attack scenario, and other researchers like [8, 35, 49] had further shown passive attack to disclose the secrets of tags.

2.1.7 Multi-round authentication protocols- HB$^+$ [27]

The HB protocol [20, 21], proposed by Hopper and Blum, is a multi-round protocol and is based on the hardness of the LPN (Learning parity with noise) problem. However, the HB protocol is only secure to passive attacks, and successive improvements like [6, 18, 29, 40, 51] tried, but in vein, to protect from active attacks. In the following, we introduce the LPN problem and HB$^+$ protocol [27]. Interested readers are referred to [6, 18, 29, 40, 51] for other HB-related works.

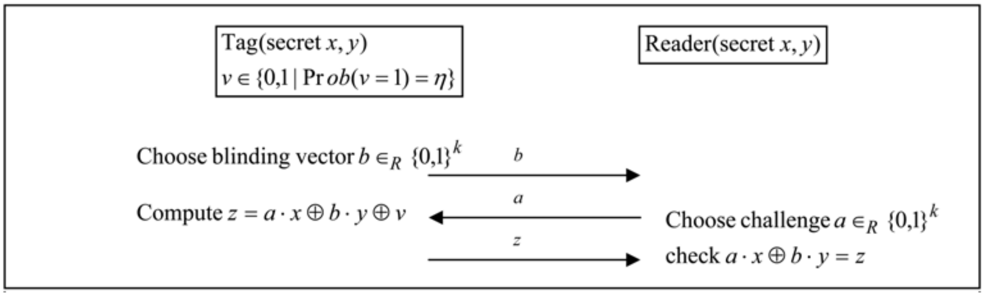
Tag(secret x, y)
$v \in \{0,1 \mid Prob(v=1) = \eta\}$
Reader(secret x, y)
Choose blinding vector $b \in_R \{0,1\}^k$
b
Compute $z = a \cdot x \oplus b \cdot y \oplus v$
a
Choose challenge $a \in_R \{0,1\}^k$
z
check $a \cdot x \oplus b \cdot y = z$

Fig. 10. One round of HB$^+$

The LPN problem: The LPN problem with security parameters q, k, η, with $\eta \in [0, \frac{1}{2}]$ is defined as: given a random $q \times k$ binary matrix A, a random k-bit vector x, a vector v such that $|v| \leq \eta q$, and the product $z = A \cdot x \oplus v$, find a k-bit vector x' such that $|A \cdot x' \oplus z| \leq \eta q$.

HB$^+$: Juels and Weis [27] tried to improve the HB protocol to resist active attacks. There are two k-bits secrets x, y between the reader and the tag. The protocol is composed of q rounds, one of which is depicted in Fig. 10. The tag is successfully authenticated if the check fails at most $q\eta$ times.

Gibert et al. [18] had shown a man-in-the-middle attack on HB$^+$. In their model, they assume that an attacker can learn whether an authentication procedure succeeds or not. One attack scenario is depicted in Fig. 11. The attack consists of two phases. First, the attacker replaces the challenge a sent by the reader with $a' = a \oplus \delta$ in all q rounds of the authentication

process, where δ is a k-bit constant vector. If the authentication succeeds, she can conclude that $\delta \cdot x = 0$ with high probability; otherwise, $\delta \cdot x = 1$ with high probability. The attacker can set only one bit of δ on each time, and repeats the process k times to reveal all the bits of x. In the second phase, the attacker impersonates the tag and sends a well chosen blinding vector b to the reader. After that, she responds to the reader challenge a with $z = x \cdot a$. If the authentication succeeds, she learns that $y \cdot b = 0$ with high probability; otherwise, she concludes that $y \cdot b = 1$ with high probability. After manipulating the bits of b and repeating the process k times, she can learn all the bits of y.
Even though several successive variants of HB$^+$ have been proposed [6, 9, 18, 29, 40, 51], none of them can resist all possible active attacks, and all the variants of HB series did not consider the anonymity and forward secrecy property.

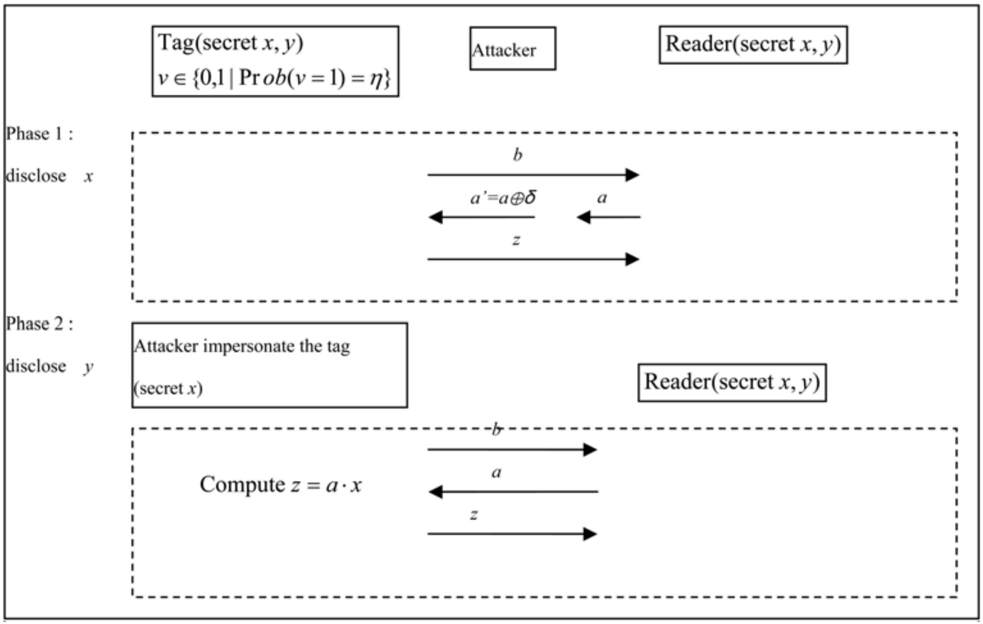

Fig. 11. Attack on HB$^+$

2.2 The resources-based classification of RFID authentication protocols

In Section 2.1, we review and discuss several RFID authentication protocols without discussing their required resources. Actually, there are various RFID tags on the market, and the capacities of these tags are quite varying: some can support public key computations, and some can only support simple bit-wise operations. However, to have large market penetration, the cost of RFID tag plays an important factor, and most of the tags targeted for consumer market would be low-cost or even very low-cost. Even though most of the RFID authentication protocols introduced in Section 2.1 are targeted for such kind of tags, the required resources of these protocols are quite varying.
Based on the required capacity on tags, we roughly classify the RFID authentication protocols into four classes. The first class called "*full-fledged* class" refers to those protocols

(like the schemes [25]) that demand the support of conventional cryptographic functions like symmetric encryption, cryptographic one-way function or even the public key algorithms. One of the main applications of these full-fledged protocols is E-passport and credit card.
The second class called "*simple*" is for those protocols (like the schemes [10, 19, 38. 43, 59, 63, 64, 66, 67]) that should support random number generator and one-way hashing function on tags
The third class called "*lightweight*" protocols refers to those protocols [6, 9, 11, 12, 15, 18, 20, 21, 22, 27-29, 40, 51] that require random number generator and simple functions like Cyclic Redundancy Code (CRC) checksum but not hash function. The EPC Gen 2 tag [17] supports both random number generator and CRC function.
The fourth class called "*ultra-lightweight*" refers to the protocols [8, 11, 13, 33-35, 45-47, 49, 62] that only involve simple bit-wise operations (like XOR, AND, OR, etc) on tags. Peris-Lopez et al. first proposed a series of *ultra-lightweight* authentication protocols [45-47], where the tags involve only simple bit-wise operations like XOR, AND, OR and addition mod 2^m. These schemes are very efficient, and they only require about 300 gates. Unfortunately, Li-Wang [34] and Li-Deng [33] reported the de-synchronization attack and the full-disclosure attack on these protocols, and Chien and Hwang [13] further pointed out the weakness of Li-Wang's improved scheme. The security weaknesses of SASI protocols are explored in [8, 35, 49, 62]. In addition to design ultra-light authentication protocols, other researcher like [41, 65] focused on designing lightweight hash function or encryption functions.

2.3 The classification based on cryptographic approaches

Contrary to the authentications in conventional applications where anonymity and un-traceability are usually not necessary properties, anonymity and un-traceability are desirable properties in many RFID applications. Therefore, this section discusses those RFID authentication protocols that consider anonymity and un-traceability, and those protocols like [6, 18, 29, 40, 51, 63, 64] that do not consider or do not well protect anonymity and un-traceability are excluded from the following discussion. Based on the technique a RFID authentication protocol uses to identify a tag while protecting the anonymity, we may classify anonymous RFID authentication protocols into the following different approaches. In describing these approaches, we focus on the techniques to identify tags while preserving the anonymity, without covering the details of the protocols.
Simple challenge-response approach. In this approach, each tag T_i shares a distinct key k_i with the server *S*/ the reader *R*. When the reader *R* probes a tag T_i by sending a random value N_R as a challenge, T_i responds with $h(k_i, N_R)$, where $h()$ denotes a secure one-way function or some function that can output commitment on its inputs while protecting the un-disclosed input k_i. Upon receiving the response $h(k_i, N_R)$, the server computes $h(k_j, N_R)$ for each potential tag T_j in its database to see whether there is a matched tag. This approach allows the server to identify a tag without disclosing the identity to eavesdroppers. Each tag just keeps one secret key, but the server needs to perform the computation for each potential tag to identify the tag. So, the tag's storage space is O(1) but the computational cost for identifying a tag is O(*n*), where *n* the number of possible tags. The previous schemes like [12, 15, 28, 30, 38, 44, 59, 66, 67] adopt this approach.
Tree-walk approach. In this approach, the tags are organized as a tree, where each leaf node in the tree denotes one tag and each edge in the tree is associated with a key. Fig. 12(b) shows

one simple example. In the example, tag *T_1* owns the key *K*1 and *K*3, and tag *T_2* owns the keys *K*1 and *K*4. When a reader probes *T_2* by sending a challenge N_R, *T_2* responds with {$h(K_1, N_R)$, s} on which the server can perform depth-first-search to identify the tag. This approach requires O($\log n$) key space on each tag and demands O($\log n$) computational cost to identify a tag. The key space requirement is a serious burden on low-cost tags. One more serious weakness of this approach is that once a tag is compromised, other tags that share the same keys on the same key paths could be partially traced. The more the number of keys one tag T_i shares with the compromised tag T_j, the more the tag T_i could be identified and traced. The schemes like [31, 32] adopt this approach.
Hash chains approach. One distinguished work of this approach is Ohkubo et al.'s scheme [43]. In this approach, the server and each tag T_x shares a distinct hash seed s_{1_x} initially. T_x updates $s_{i+1\_x} = h(s_{i\_x})$ for $i \geq 1$ and responds with $a_{i\_x} = g(s_{i\_x})$ for each query request, where *h*()/*g*() are two different hash functions. This approach owns the forward secrecy property; that is, even assuming a tag is compromised one day in the future, the past communications from the same tag can not be traced. However, Ohkubo et al.'s original version cannot resist the replay attack [1], and has the poor scalability problem [2, 3] – the computational cost to identify a tag is O(*nm*), where *n* is the number of potential tags and *m* is the maximum length of the hash chain. Avoine et al. [1] discussed the techniques to conquer the replay attack, and Avoine et al. also [1, 2] also proposed their improvements to reduce the time complexity at the cost of extra memory.
Varying Pseudonym (VP) approach. In this approach like [11, 19,45-47], each tag synchronizes its varying identifier and its internal state with the server. Please notice that, even though some challenge-response-based schemes like [2, 12, 15] also synchronize the state between the tags and the server, these schemes do not send a varying pseudonym to facilitate the server perform fast identification; we, therefore, do not count them in this VP approach. The varying identifier is called pseudonym in [11, 45-47], and is called metaID in [19, 31, 32]. Here, we all refer to them the pseudonyms. Upon receiving a challenge request, a tag responds with the current pseudonym and the commitment on the challenge and the secret internal state. Based on the commitment, the server can verify the tag. During the authentication, the tag and the server respectively update their pseudonyms and their internal state. In this approach, the pseudonym not only protects the anonymity of the tag but also facilitates the server to identify the tag in its database with O(1) computational complexity, because the server can directly use the pseudonym to locate the corresponding entry in its database and perform necessary computations for this matched entry only. Further more, each tag only needs constant quantity of internal values- O(1) key storage. It is these excellent features that make it quite attractive than the other approaches. However, due to the synchronization requirement, the VP-based schemes are prone to the de-synchronization attacks (or the denial of service attacks) [8, 13, 33-35, 49, 62], if adversaries can manipulate the communications such that the tag and the server are out of synchronization. Fig. 12 depicts the main ideas of these approaches.

3. Security analysis of the mifare ultralight card and OV-chipkaart

In Section 2, we have examined several RFID authentication protocols published in the literature. In this section and the next, we shall examine the security of some popular tags on the market. Section 3 will discuss the Mifare Ultralight card [36], and Section 4 will cover the EPC Class 1 Generation 2 card [16, 17].

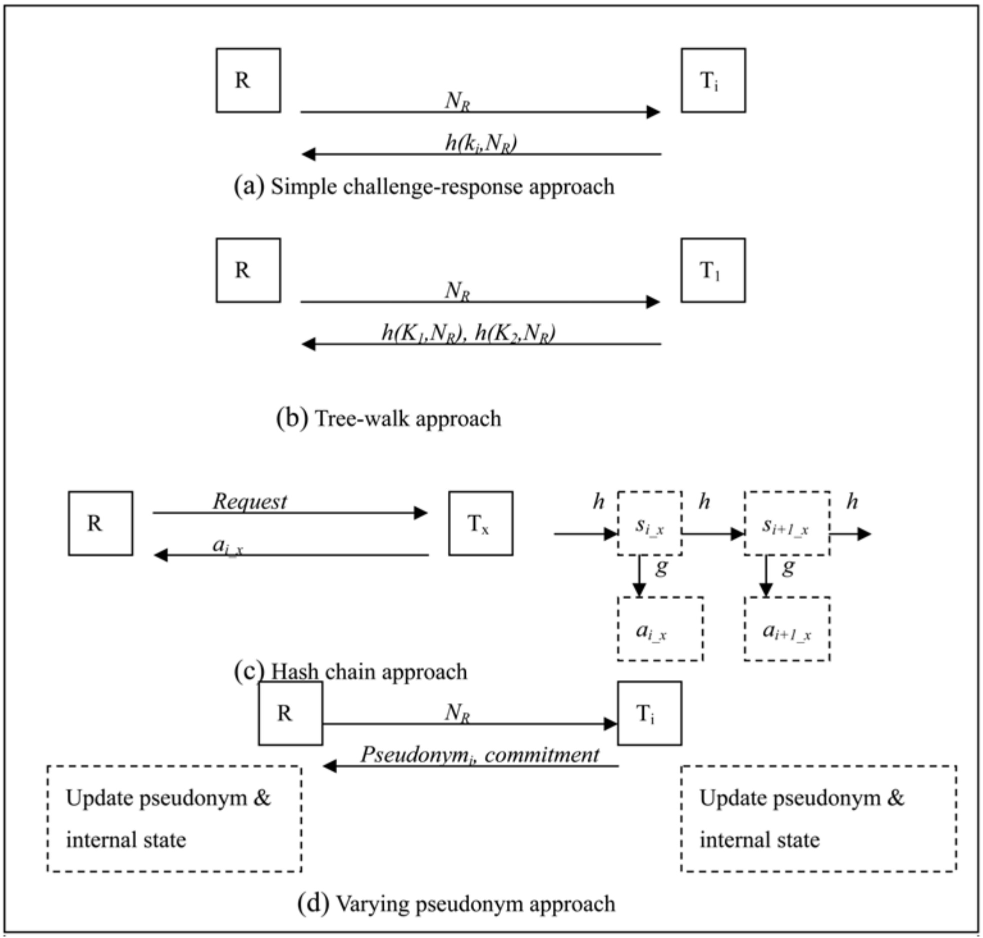

Fig. 12. Approaches to protect RFID tag identity

Mifare is a trademark of contactless RFID products and technologies developed by NXP Semiconductors [42]. Mifare cards have been widely used in many countries, and some of the cards feature a high level of security. However, Mifare Ultralight [42], one of Mifare card series, is focused on supporting faster applications at a cheaper cost. Thus, there is not any security mechanism implemented on the Mifare Ultralight chip. All privileges are fully accessible by anyone on the memory block. Section 3.1 right below will introduce Mifare card series, followed by Section 3.2 that deals with the memory organization of Mifare Ultralight. Then, Section 3.3 will take the OV-chipkaart in the Netherlands, which runs on the basis of the Mifare Ultralight card, as an example to discuss the security weaknesses and possible threats.