Journal of Computational and Applied Mathematics 259 (2014) 512–521

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

On the security of RFID anti-counting security protocol (ACSP)

Masoumeh Safkhani [a], Nasour Bagheri [b,*], Ali Mahani [c]

[a] Electrical Engineering Department, Iran University of Science and Technology, Tehran, Iran
[b] Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran, 16788-15811, Iran
[c] Department of Electrical and Electronic Engineering, Shahid Bahonar University of Kerman, Iran

ARTICLE INFO

Article history:
Received 15 February 2013
Received in revised form 12 June 2013

Keywords:
RFID
Authentication
Counting attack
Tag impersonation attack
De-synchronization attack
Traceability attack

ABSTRACT

Recently Qian et al. (2012) [26] have proposed a new attack for RFID systems, called counting attack, where the attacker just aims to estimate the number of tagged objects instead of steal the tags' private information. They have stated that most of the existing RFID mutual authentication protocols are vulnerable to this attack. To defend against counting attack, they proposed a novel Anti-Counting Security Protocol called ACSP. The designers of ACSP have claimed that their protocol is resistant against counting attack and also the other known RFID security threats. However in this paper we present the following efficient attacks against this protocol:

- Two tag impersonation attack: the success probability of each attack is "1" while the complexity is at most three runs of the protocol.
- Two single tag de-synchronization attacks, the success probability of both attacks is "1" while the complexity is at most two runs of the protocol.
- Group of tags de-synchronization attack: this attack, which can de-synchronize all tags in the range at once, has a success probability of "1" while its complexity is one run of the protocol.
- Traceability attack: the adversary's advantage in this attack is almost the maximum of possible advantages for an adversary in the same model, i.e., $\frac{1}{2}$. The complexity of this attack is three runs of the protocol.

To counteract such flaws, we improve the ACSP protocol by applying some modifications so that it provides the desired security.

1. Introduction

Radio Frequency IDentification (RFID) systems consist of a reading device called reader and one or more tags. The reader is a powerful device which can read/modify the tag's information. Tags are very constraint devices that range from passive tags, which respond only at reader commands, to active tags, which have an on-board power supply.

Designing secure authentication protocols for low-cost RFID tags has received the attention of a lot of researchers, though many protocols have been published lately [1–19]. However, most of them have not satisfied the claimed security goals [20–25]. The security of an RFID protocol can be analyzed in several directions. For example, an adversary may be interested in tracking the tag's holder which can compromise the tag's holder privacy or it may try to clone a legitimate tag to pay less. We can classify the main known attacks in the field of RFID as follows:

* Corresponding author. Tel.: +98 21 2297006; fax: +98 21 2297006.
E-mail addresses: M_Safkhani@iust.ac.ir (M. Safkhani), Nbagheri@srttu.edu, na.bagheri@gmail.com (N. Bagheri), AMahani@uk.ac.ir (A. Mahani).

0377-0427/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cam.2013.06.016

- Forgery attacks that include:
  - Tag impersonation.
  - Reader impersonation.
- Secret disclosure attacks that include:
  - Reader's secret values, e.g. shared *key*, disclosure attack.
  - Tag's secret values, e.g. tag's *ID*, disclosure attack.
- De-synchronization attacks.
- Traceability attacks.
- Cloning attacks.
- Replay attacks.

At the moment, the RFID community is short of secure and lightweight protocols for RFID, but designing a secure authentication is not an easy task, many designed and easily broken protocols over the last decade confirm this claim. Security cryptanalysis of protocols is one of the works we can do to aid secure protocol designing. Cryptanalysis of protocols shows the different ways to break the security of protocols and thereby to show some guidelines to design secure protocols and prevent predecessor protocols' weaknesses. So from this viewpoint, cryptanalysis of protocols advances the research of the RFID field.

On the other hand, RFID has potential to be used in many applications. However, any application has its own security concerns and new application may introduce new concerns. For example, Qian et al. have recently proposed a new attack, called a counting attack, in which the adversary's target is to estimate the number of tagged objects. They have stated that most of the existing RFID protocols are vulnerable to counting attack. To clarify why the counting attack and the protocol ASCP is important, consider the following scenarios stated in [26]: in a warehouse, a tag is attached to each piece of merchandise to keep a unique ID which is used for automatic identification. Now, to count the number of tags using a portable reader, a spy may sneak into the warehouse and rapidly get stock information, e.g., the amount of goods. This behavior of the adversary where she counts the number of objects through estimating the cardinality of attached tags is defined as a counting attack. Such a security concern also could exist in vehicular ad-hoc network (VANET) systems. To address counting attack in RFID systems, Qian et al. have proposed a new protocol entitled ACSP [26]. Qian et al. have claimed that ACSP is secure against counting attack and also against the other attacks in the context. Although their claims are too strong, ACSP's security analysis has worth to be considered. So in this paper we consider ACSP security.

Adversary model: In this paper, all attacks are performed using a man-in-the-middle configuration. More precisely, the attacker is an active adversary who is able to intercept, modify and block the ongoing reader-tag communication without being detected.

Paper contributions: In this paper we show three kinds of attacks against ACSP. The first attack is the tag impersonation attack which is able to force the reader to authenticate the adversary as a legitimate tag. The second attack is a de-synchronization attack which is able to de-synchronize the communication between the tag and the reader and the third attack is a traceability attack which is able to trace tags and compromise the privacy of the tag's holder. The success probability of each attack is "1", the complexity of each attack is at most three runs of the protocol and the main computation cost of any attack is at most one calculation of hash function and one calculation of *CRC* function. In fact, this paper shows that ACSP has critical weaknesses and cannot be used in any application which needs to resist against the above mentioned attacks. Finally, this paper introduces an improved version of the protocol called $ACSP^{+}$ that can resist against the above mentioned attacks and also the other known active and passive attacks in the context.

Throughout the paper, we use the notations which are depicted in Table 1. Each message which is sent from the reader to the tag or vice versa includes a message header, $\overline{MSG\_HEADER}$, the message body and the *CRC* of the message. The $\overline{MSG\_HEADER}$ can be $\overline{SELECT}$, $\overline{QUERY}$, $\overline{IDENT}$, $\overline{AUTHEN}$ or $\overline{QUERYREP}$.

Paper organization: The ACSP protocol is briefly described in Section 2. In Section 3, we describe our impersonation attack which can be also considered as de-synchronization attack. We present another de-synchronization attack which de-synchronizes a single tag and the reader in Section 4. In Section 5, we present another de-synchronization attack which de-synchronizes all tags in the range. A traceability attack is described in Section 6. Section 7 presents an improved version of the ACSP protocol, $ACSP^{+}$, and its security analysis. Finally, we conclude the paper in Section 8.

2. ACSP description

ACSP is composed of two phases, i.e., SID Update and Tag Identification, that are described separately below:

SID update phase: This phase of the protocol, which is depicted in Fig. 1, accomplishes as below:

1. The reader generates two random numbers R_1 and R_2 and sends $(\overline{UPDSID}, R_1, R_2, H(R_1, SID_{\text{cur}}), CRC)$ to the tag.
2. Upon receipt of the message, each tag checks the values of $H(R_1, SID_{\text{cur}})$ and *CRC* to verify whether the command is valid. If it is valid, the tag updates local *SID* as $SID_{\text{new}} = H(R_1 \oplus R_2, SID_{\text{cur}})$ and sends $(\overline{UPDACK}, R_2, CRC)$ to the reader.
3. Once the reader receives the message, he checks the values of R_2 and *CRC* to verify whether the command is valid. If it is valid, it updates local *SID* as $SID_{\text{new}} = H(R_1 \oplus R_2, SID_{\text{cur}})$.

Table 1
Notation.

Notation	Description
$\mathbb{R}$:	RFID reader
$\mathbb{T}$:	RFID tag
SID:	The session identifier
SID_{cur}:	The current session identifier
SID_{new}:	The new session identifier used in the following communication session
TID:	The current unique identifier of tag
TID_{new}:	The next identifier of tag used in the following communication session
R_i, $1 \le i \le 5$:	Pseudo random numbers that are generated by the reader or the tag
$H(x_1, x_2)$:	One way hash function with variables x_1 and x_2
CRC:	Cyclic Redundancy Code
$\oplus$:	Exclusive-or operation
$\mathcal{A}$:	Adversary
$\overline{MSG\_HEADER}$:	Header of message which indicates the type of the message, including Select, Query, Identification, Authentication and End commands
$\overline{SELECT}$:	Select command
$\overline{QUERY}$:	Query command
$\overline{IDENT}$:	Identification message from tag, containing *TID* information
$\overline{AUTHEN}$:	Authentication message
$\overline{UPDSID}$:	SID update command
$\overline{UPDACK}$:	Update acknowledge message from tag
$\overline{QUERYREP}$:	Command used for ending current slot
MASKVAL:	Bit mask contained in Select command used for selecting target tag

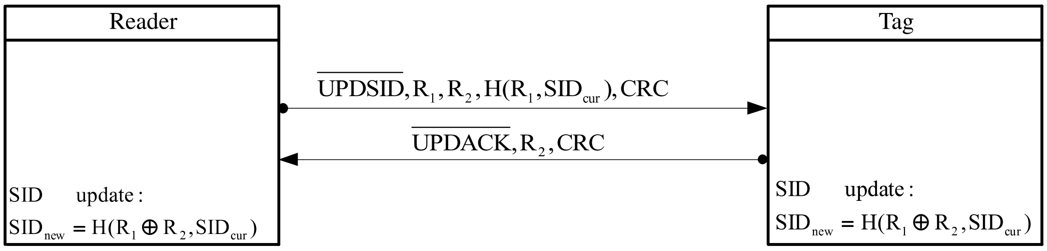
Reader
Tag
$\overline{UPDSID}, R_1, R_2, H(R_1, SID_{cur}), CRC$
$\overline{UPDACK}, R_2, CRC$
SID update :
$SID_{new} = H(R_1 \oplus R_2, SID_{cur})$
SID update :
$SID_{new} = H(R_1 \oplus R_2, SID_{cur})$

Fig. 1. The ACSP's SID update phase.

Tag identification phase: This phase of the protocol, which is depicted in Fig. 2, works as follows:

1. The reader $\mathbb{R}$ generates a random number R_3 and sends $(\overline{SELECT}, R_3, H(R_3, SID), (MASKVAL \oplus SID), CRC)$ to the target tag.
2. Upon receipt of the message, any tag $\mathbb{T}_j$ checks the included *CRC*, R_3 and corresponding $H(R_3, SID)$. If all are correct, the tag extracts *MASKVAL* by calculating $(MASKVAL \oplus SID) \oplus SID$. If the tag's identifier *TID* matches *MASKVAL*, the tag gets ready to respond, otherwise the tag keeps silent until receiving the next Select command.
3. $\mathbb{R}$ generates a random number R_4 and sends $(\overline{QUERY}, R_4, H(R_4, SID), CRC)$.
4. Upon receipt of the message, the ready tag $\mathbb{T}_i$ checks the received *CRC* and $(R_4, H(R_4, SID))$ in the Query command. If they are correct, it generates a random number R_5 and responds with $(\overline{IDENT}, R_5, H(R_4, TID), CRC)$ as an identification message.
5. Upon receipt of the identification message without collision, $\mathbb{R}$ proceeds as follows:
   (a) It searches for the tag's *TID* in its database according to R_4 and $H(R_4, TID)$.
   (b) If $\mathbb{R}$ finds the *TID* of the tag it does as follows:
   - Updates tag's *TID* as $TID_{\text{new}} = H(R_4 \oplus R_5, TID)$.
   - Sends $(\overline{AUTHEN}, H(R_5, TID), CRC)$ to the tag.

   (c) Else, the reader sends $(\overline{QUERYREP}, R_p, H(R_p, SID), CRC)$ to end this slot and starts a new one.
6. The tag receives $(\overline{AUTHEN}, H(R_5, TID), CRC)$ and upon receipt of these values, it checks $H(R_5, TID)$ and proceeds as follows:
   (a) If it is correct, it updates its identifier as $TID_{\text{new}} = H(R_4 \oplus R_5, TID)$.
   (b) Else, it does nothing.

3. Tag impersonation attack

The tag impersonation attack is a forgery attack in which the reader accepts a spoofed tag as a legitimate tag. Any secure RFID authentication protocol must resist against all kind of forgery attacks, including the tag impersonation attack. In this

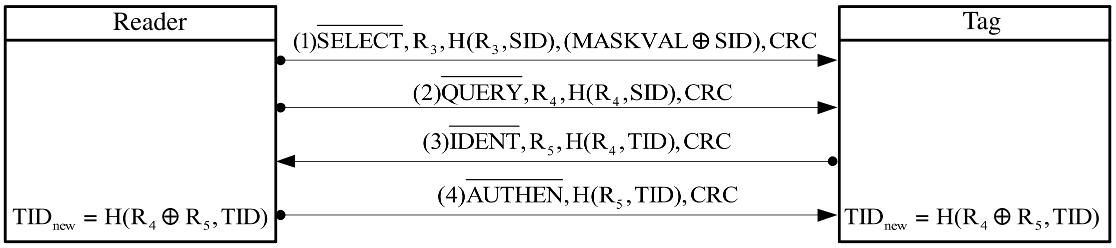
Reader
Tag
(1)$\overline{\text{SELECT}}$, R$_3$, H(R$_3$,SID), (MASKVAL ⊕ SID), CRC
(2)$\overline{\text{QUERY}}$, R$_4$, H(R$_4$,SID), CRC
(3)$\overline{\text{IDENT}}$, R$_5$, H(R$_4$,TID), CRC
(4)$\overline{\text{AUTHEN}}$, H(R$_5$,TID), CRC
TID$_{new}$ = H(R$_4$ ⊕ R$_5$,TID)
TID$_{new}$ = H(R$_4$ ⊕ R$_5$,TID)

Fig. 2. The ACSP's tag identification phase.

section, we prove that ACSP is vulnerable to the tag impersonation attack. In our tag impersonation attack, to impersonate the tag $\mathbb{T}_i$ the adversary ($\mathcal{A}$) can follow the steps below:

Phase 1: *Retrieving TID*$_{\text{new}}$

1. The reader $\mathbb{R}$ generates a random number R_3 and sends $(\overline{SELECT}, R_3, H(R_3, SID), (MASKVAL \oplus SID), CRC)$ to the target tag $\mathbb{T}_i$. $\mathcal{A}$ does not change the transferred message of this step.
2. Upon receipt of the message, any tag $\mathbb{T}_j$ checks the included CRC, R_3 and corresponding $H(R_3, SID)$. If all are correct, $\mathbb{T}_i$ extracts $MASKVAL$ by calculating $(MASKVAL \oplus SID) \oplus SID$. If the tag's identifier TID matches $MASKVAL$, $\mathbb{T}_i$ gets ready to respond, otherwise it keeps silent until receiving the next Select command.
3. $\mathbb{R}$ generates a random number R_4 and sends $(\overline{QUERY}, R_4, H(R_4, SID), CRC)$. $\mathcal{A}$ does not change the transferred message of this step.
4. Upon receipt of the message, a ready tag $\mathbb{T}_i$ checks CRC and the correctness of $(R_4, H(R_4, SID))$ in the Query command. If they are correct, it generates a random number R_5 and responds with $(\overline{IDENT}, R_5, H(R_4, TID), CRC)$ as an identification message.
5. $\mathcal{A}$ intercepts $(\overline{IDENT}, R_5, H(R_4, TID), CRC)$, changes the R_5 value to zero, computes the corresponding CRC of $(\overline{IDENT}, 0, H(R_4, TID))$(denoted by CRC') and sends $(\overline{IDENT}, 0, H(R_4, TID), CRC')$ to the reader.
6. Upon receipt of the identification message without collision, $\mathbb{R}$ proceeds as follows:
   (a) It searches for the tag's identifier TID in its database according to R_4 and $H(R_4, TID)$.
   (b) If $\mathbb{R}$ finds the TID of $\mathbb{T}_i$, it continues as follows:
   - Updates $\mathbb{T}_i$'s identifier TID as $TID_{\text{new}} = H(R_4 \oplus R_5, TID) = H(R_4 \oplus 0, TID) = H(R_4, TID)$.
   - Sends $(\overline{AUTHEN}, H(R_5, TID), CRC)$ to $\mathbb{T}_i$.
7. $\mathcal{A}$ blocks this message.

Hence, following the above procedure, the adversary knows $TID_{\text{new}} = H(R_4, TID)$ and the secret TID of $\mathbb{T}_i$ stored in the reader database.

Phase 2: *Tag impersonation*: to impersonate the tag, $\mathcal{A}$ waits until the reader initiates a new session to identify any tag $\mathbb{T}_j$ where:

1. $\mathbb{R}$ generates a random number R_3 and sends $(\overline{SELECT}, R_3, H(R_3, SID), (MASKVAL \oplus SID), CRC)$.
2. Upon receipt of the message, $\mathcal{A}$ gets ready to respond.
3. $\mathbb{R}$ generates a random number R_4 and sends $(\overline{QUERY}, R_4, H(R_4, SID), CRC)$.
4. Upon receipt of the message, $\mathcal{A}$ blocks any response from other tags towards $\mathbb{R}$ and generates a random number R_5 and responds with $(\overline{IDENT}, R_5, H(R_4, TID), CRC)$ as identification message, where TID has been retrieved from *Phase* 1 of attack.
5. Upon receipt of the identification message without collision, $\mathbb{R}$ proceeds as follows:
   (a) It searches for the tag's TID in its database according to R_4 and $H(R_4, TID)$.
   (b) It finds TID of $\mathbb{T}_i$ and continues as follows:
   - Updates $\mathbb{T}_i$'s identifier TID as $TID_{\text{new}} = H(R_4 \oplus R_5, TID) = H(R_4 \oplus 0, TID) = H(R_4, TID)$.
   - Sends $(\overline{AUTHEN}, H(R_5, TID), CRC)$ to $\mathbb{T}_i$.
6. $\mathcal{A}$ updates its record of TID to $TID_{\text{new}} = H(R_4 \oplus R_5, TID)$.

So the reader authenticates the adversary as a legitimate tag and updates tag's identifier TID. The success probability of the above attack is "1" and the complexity is only two successive runs of the protocol. We emphasize that, as has been indicated in the attack, in *Phase* 2 of the attack the adversary does not need to wait for a special session of the protocol between $\mathbb{R}$ and the target tag $\mathbb{T}_i$ to impersonate $\mathbb{T}_i$, but it can impersonate $\mathbb{T}_i$ on any run of the protocol after *Phase* 1 of the attack, where the adversary retrieves TID of $\mathbb{T}_i$.

However, the above presented attack is based on a very weak aspect of ACSP implementation. More precisely, the attacker sends a zero value instead of a random number. This fact is easy to solve by ignoring any session which includes $R_5 = 0$. Thus, this attack may not be very realistic. However, even in that case it is possible to impersonate the tag as follows:

Phase 1: *Learning*

1. The reader $\mathbb{R}$ generates a random number R_3 and sends $(\overline{SELECT}, R_3, H(R_3, SID), (MASKVAL \oplus SID), CRC)$ to the target tag $\mathbb{T}_i$.
2. Upon receipt of the message, any tag $\mathbb{T}_j$ checks the included CRC, R_3 and corresponding $H(R_3, SID)$. If all are correct, $\mathbb{T}_i$ extracts $MASKVAL$ by calculating $(MASKVAL \oplus SID) \oplus SID$. If the tag's identifier TID matches $MASKVAL$, $\mathbb{T}_i$ gets ready to respond, otherwise it keeps silent until receiving the next Select command.
3. $\mathbb{R}$ generates a random number R_4 and sends $(\overline{QUERY}, R_4, H(R_4, SID), CRC)$.
4. Upon receipt of the message, a ready tag $\mathbb{T}_i$ checks CRC and correctness of $(R_4, H(R_4, SID))$ in the Query command. If they are correct, it generates a random number R_5 and responds with $(\overline{IDENT}, R_5, H(R_4, TID), CRC)$ as identification message.
5. $\mathcal{A}$ blocks the received $(\overline{IDENT}, R_5, H(R_4, TID), CRC)$ but stores them for the later usage.
6. The protocol is ended because the reader does not receive the desired response.

Hence, following the above procedure, the adversary knows the value of R_4 and $H(R_4, TID)$.

Phase 2: *Retrieving* TID_{new}

1. The reader $\mathbb{R}$ generates a random number R'_3 and sends $(\overline{SELECT}, R'_3, H(R'_3, SID), (MASKVAL \oplus SID), CRC)$ to the target tag $\mathbb{T}_i$. $\mathcal{A}$ does not change the transferred message of this step.
2. Upon receipt of the message, any tag $\mathbb{T}_j$ checks the included CRC, R'_3 and corresponding $H(R'_3, SID)$. If all are correct, $\mathbb{T}_i$ extracts $MASKVAL$ by calculating $(MASKVAL \oplus SID) \oplus SID$. If the tag's identifier TID matches $MASKVAL$, $\mathbb{T}_i$ gets ready to respond, otherwise it keeps silent until receiving the next Select command.
3. $\mathbb{R}$ generates a random number R'_4 and sends $(\overline{QUERY}, R'_4, H(R'_4, SID), CRC)$. $\mathcal{A}$ does not change the transferred message of this step.
4. Upon receipt of the message, a ready tag $\mathbb{T}_i$ checks CRC and correctness of $(R'_4, H(R'_4, SID))$ in the Query command. If they are correct, it generates a random number R'_5 and responds with $(\overline{IDENT}, R'_5, H(R'_4, TID), CRC)$ as identification message.
5. $\mathcal{A}$ intercepts $(\overline{IDENT}, R'_5, H(R'_4, TID), CRC)$, changes R_5 value to $R''_5 = R'_4 \oplus R_4$, computes the corresponding CRC of $(\overline{IDENT}, R''_5, H(R'_4, TID))$ and sends $(\overline{IDENT}, R''_5, H(R'_4, TID), CRC')$ to the reader.
6. Upon receipt of the identification message without collision, $\mathbb{R}$ proceeds as follows:
   (a) It searches for the tag's identifier TID in its database according to R'_4 and $H(R'_4, TID)$.
   (b) If $\mathbb{R}$ finds the TID of $\mathbb{T}_i$, which will find, it does as follows:
   - Updates $\mathbb{T}_i$'s identifier TID as $TID_{\text{new}} = H(R'_4 \oplus R''_5, TID) = H(R'_4 \oplus R'_4 \oplus R_4, TID) = H(R_4, TID)$.
   - Sends $(\overline{AUTHEN}, H(R''_5, TID), CRC)$ to $\mathbb{T}_i$.
7. $\mathcal{A}$ blocks this message.

Hence, following the above procedure, the adversary knows the $TID_{\text{new}} = H(R_4, TID)$ which is the secret TID of $\mathbb{T}_i$ stored in the reader database.

Phase 3: *Tag impersonation*: This Phase of attack is identical to Phase 2 of the previous attack.

So the reader authenticates the adversary as a legitimate tag and updates tag's identifier TID. The success probability of the above attack is "1" and the complexity is only three successive runs of the protocol.

Remark 1. After the successful run of the given tag impersonation attacks, the reader authenticates the adversary as a legitimate tag and updates the tag's identifier TID, while the legitimate tag has not updated its TID. Hence, the reader and the legitimate tag will not authenticate each other anymore in the following transactions. Therefore, the given impersonation attacks lead to de-synchronization attacks.

Remark 2. To overcome the potential de-synchronization attacks because of the message lost problem occurring in the tag identification procedure, designers suggested [26, Section 6] that the reader preserves a copy of TID used in the last successful identification for each tag. However, this modification also does not improve the security of the protocol against the proposed attacks.

4. Single tag de-synchronization attack

In a de-synchronization attack, the adversary forces the tag and the reader to update their common values to different values from each other. If the adversary can succeed in forcing the tag and the reader to do so, they will not authenticate each other in the further transactions. Qian et al. [26] have stated that it is possible to de-synchronize the tag and the reader in ACSP if the adversary blocks the transferred message from the reader to the tag in step 5b, and to solve the problem, as an enhanced protocol, they suggested [26, Section 6] that the reader preserves a copy of TID used in the last successful identification for each tag. However, in Section 3 we have described a powerful tag impersonation attack which also de-synchronizes the target tag and the reader in their original and enhanced protocols.

In this section, we present a different de-synchronization attack against the original protocol which is not following Qian et al.'s suggestion to attack the protocol. Our de-synchronization attack is rather simple and it is based on this fact that in step 3 of the ACSP protocol the random number R_5 does not affect any part of the transferred message, except the *CRC* value. Hence, if an adversary $\mathcal{A}$ changes R_5 and *CRC* properly, there is no way for the tag to understand the modification. We use this observation in our de-synchronization attack against ACSP. In this attack, $\mathcal{A}$ does as follows:

- $\mathcal{A}$ lets steps 1 and 2 of the protocol be run without any change.
- $\mathcal{A}$ intercepts the message of step 3 of the protocol and changes the value of R_5 to some arbitrary value, i.e. $R_5 \oplus \Delta$, and computes the corresponding *CRC* of $(\overline{IDENT}, R_5 \oplus \Delta, H(R_4, TID))$ denoted by CRC'. Then it sends $(\overline{IDENT}, R_5 \oplus \Delta, H(R_4, TID), CRC')$ to the reader.
- The reader authenticates the adversary as a legitimate tag. Then $\mathbb{R}$ updates tag's identifier *TID* as $TID_{\text{new}} = H(R_4 \oplus R_5 \oplus \Delta, TID)$.

It must be noted that the reader updates the tag's identifier *TID* by $R_5 \oplus \Delta$ while the tag does not update its *TID* so they cannot authenticate each other in the following transactions. The success probability of the above attack is "1" and the complexity is only one run of the protocol.

5. Group de-synchronization attack

In this section we introduce an attack which de-synchronizes all tags in the range. It must be noted that in ACSP the reader and tags share the session identifier, *SID*. This value gets updated periodically following the SID Update Phase 2 of the protocol. However, if an adversary forces the tag to update its SID_{new} to a different value compared to the value in the reader side, then the Tag Identification Phase 2 of the protocol cannot be run properly and the tag and the reader will be de-synchronized. In addition, in the SID update phase of the protocol, all tags must update their *SID*, otherwise they will lose their synchronization with the reader. On the other hand, to update *SID*, the reader generates two random numbers R_1 and R_2 and sends $(\overline{UPDSID}, R_1, R_2, H(R_1, SID_{\text{cur}}), CRC)$ to the tags. Each tag, upon receipt of the message, checks the values of $H(R_1, SID_{\text{cur}})$ and *CRC* and if it is valid it updates local *SID* as $SID_{\text{new}} = H(R_1 \oplus R_2, SID_{\text{cur}})$. It can be seen that the adversary $\mathcal{A}$ can proceed as follows:

1. $\mathcal{A}$ intercepts the message,
2. changes R_2 to $R_2' \neq R_2$,
3. calculates the $CRC' = CRC(\overline{UPDSID}, R_1, R_2', H(R_1, SID_{\text{cur}}))$,
4. and broadcasts $(\overline{UPDSID}, R_1, R_2', H(R_1, SID_{\text{cur}}), CRC')$ to all tags in the range.

Obviously, the modified message will pass the tags verification test and all tags in the range will update their *SID* to $SID_{\text{new}} = H(R_1 \oplus R_2', SID_{\text{cur}})$. Since with a high probability $H(R_1 \oplus R_2, SID_{\text{cur}}) \neq H(R_1 \oplus R_2', SID_{\text{cur}})$ (the exact probability is $1 - 2^{-L}$ where L is the length of *SID* in bits), all tags are de-synchronized from the reader. The success probability of the above attack is almost "1" and the complexity is only one run of protocol.

Remark 3. It must be noted that the designers have discussed [26, Section 6] that the tag and the reader will be de-synchronized if the tag does not receive the $\overline{UPDSID}$ command. To overcome this problem they suggested any tag to send an update acknowledge command as $(\overline{UPDACK}, R_2, CRC)$ to the reader. However, this message can be generated by the adversary and sent to the reader because it does not include any secret parameter. Therefore, the enhanced protocol is also vulnerable to the given attack.

6. Traceability attack

In this section we show how ACSP puts at stake the location privacy of the tags' holders because tags can be tracked with a high probability. Specifically, a target tag $\mathbb{T}_i$ which is supposed to trace is given to the adversary. Later, a random tag $\mathbb{T}_j$ is given to $\mathcal{A}$ and the adversary should verify whether it is $\mathbb{T}_i$. She outputs its decision as a single bit $\widetilde{b} \in \{0, 1\}$, 0 for $T_j \neq T_i$ and 1 for $T_j = T_i$ cases. The adversary's success in winning the traceability game $\mathcal{G}$ is equivalent to the success of breaking the untraceability property offered by the protocol. So the advantage of $\mathcal{A}$ in distinguishing whether the messages correspond to T_i or not is defined as below:

$$\text{Adv}_{\mathcal{A}}^{\text{UNT}}(q, kr) = \left| \Pr[\widetilde{b} \text{ is correct}] - \frac{1}{2} \right|$$

where q is a security parameter (i.e. the bit length of the key shared between the tag and the reader) and kr is the number of times $\mathcal{A}$ runs the protocol. Our traceability attack is described as below:

Phase 1: *Learning*, in this phase of attack the adversary de-synchronizes $\mathbb{T}_i$ and $\mathbb{R}$. In addition, through the de-synchronization attack she can achieve the required information to trace $\mathbb{T}_i$ later.

1. $\mathcal{A}$ de-synchronizes $\mathbb{T}_i$ and $\mathbb{R}$ following the given tag impersonation attack which also leads to de-synchronization (refer to Remark 1) and stores the following messages transferred between $\mathbb{R}$ and $\mathbb{T}_i$, include:
   - $(\overline{SELECT}, R_3, H(R_3, SID), (MASKVAL \oplus SID), CRC)$, from $\mathbb{R}$ to $\mathbb{T}_i$.
   - $(\overline{QUERY}, R_4, H(R_4, SID), CRC)$, from $\mathbb{R}$ to $\mathbb{T}_i$.
   - $(\overline{IDENT}, R_5, H(R_4, TID), CRC)$, from $\mathbb{T}_i$ to $\mathbb{R}$.

Phase 2: *Challenging*, in this phase of protocol the adversary does as follows:

1. $\mathcal{A}$ sends the eavesdropped $(\overline{SELECT}, R_3, H(R_3, SID), (MASKVAL \oplus SID), CRC)$ to the target tag $\mathbb{T}_j$.
2. Upon receipt of the message, $\mathbb{T}_j$ checks the included CRC, R_3 and corresponding $H(R_3, SID)$. If $T_j = T_i$ then all are correct and $\mathbb{T}_j$ extracts $MASKVAL$ by calculating $(MASKVAL \oplus SID) \oplus SID$. The tag's identifier TID matches $MASKVAL$ and $\mathbb{T}_j$ gets ready to respond. If $T_j \neq T_i$ it will keep silent.
3. $\mathcal{A}$ sends the eavesdropped $(\overline{QUERY}, R_4, H(R_4, SID), CRC)$ to $\mathbb{T}_j$.
4. Upon receipt of the message, if $\mathbb{T}_j$ is ready, it checks CRC and the correctness of $(R_4, H(R_4, SID))$ in the Query command. If they are correct then it generates a random number R'_5 and responds with $(\overline{IDENT}, R'_5, H(R_4, TID'), CRC')$ as an identification message.
5. $\mathcal{A}$ eavesdrops $(\overline{IDENT}, R'_5, H(R_4, TID'), CRC')$ and goes to the next phase of attack.

Phase 3: *Guessing*, $\mathcal{A}$ finishes $\mathcal{G}$ and outputs a bit $\widetilde{b}$ as its conjecture of the value b. In particular, $\mathcal{A}$ utilizes the following simple decision rule in generating the decision bit $\widetilde{b}$:

$$\begin{cases} \text{if } H(R_4, TID) == H(R_4, TID') & \widetilde{b} = 1 \\ \text{if } H(R_4, TID) \neq H(R_4, TID') & \widetilde{b} = 0. \end{cases} \tag{1}$$

In the given attack, the adversary de-synchronizes the reader and the target tag $\mathbb{T}_i$ at the first phase of attack. Hence, $\mathbb{T}_i$ will not update its secret parameters including SID and TID. On the other hand, as long as SID and TID are fixed, the eavesdropped values in the Learning phase of attack pass the tags verification tests in the protocol and the adversary will expect the same $H(R_4, TID)$ from the tag to be included in its $\overline{IDENT}$ command. Hence, if $T_j = T_i$ then with the probability of "1" we have $H(R_4, TID) == H(R_4, TID')$. However, if $T_j \neq T_i$ then with a negligible probability $H(R_4, TID) == H(R_4, TID')$. This probability is less than 2^{-L} where L is the output length of hash function, $H(.)$. Therefore, the success probability of adversary to output the correct $\widetilde{b}$ is lower bounded by $1 - 2^{-L}$. Hence, the adversary's advantage in winning the game is as follows, which is almost the maximum of possible advantages:

$$\mathrm{Adv}_A^{\mathsf{UNT}}(q, 1) = \left|1 - 2^{-L} - \frac{1}{2}\right| = \left|\frac{1}{2} - 2^{-L}\right|.$$

The total complexity of the attack is "3" runs of protocol, two runs in the Learning phase and one run in the Challenging phase of attack.

7. Improved protocol

The main drawback of the ACSP protocol is the use of CRC to provide the integrity of the transferred messages, and it is well known that an active adversary can change a part of message and also calculate the related CRC adaptively to easily pass this checking. To fix this problem, it is possible to use hash based message authentication codes (MAC) to provide the message integrity. For example, in the SID Update Phase 2 of protocol, instead of $(\overline{UPDSID}, R_1, R_2, H(R_1, SID_{\mathrm{cur}}), CRC)$ which can be manipulated easily by an active adversary, to de-synchronize the tag, one can use $(\overline{UPDSID}, R_1, R_2, H(\overline{UPDSID}, R_1, R_2, SID_{\mathrm{cur}}))$ for which any adversary has a negligible chance to manipulate without the knowledge of SID_{cur}. Although this suggestion fixes the problem of integrity, it does not rule out the replay attack. To fix this problem, any message, which is used to decide whether to update a parameter, should be randomized by nonces contributed by both parties. Hence, to design the new protocol, called $ACSP^+$, we follow this approach.

7.1. $ACSP^+$ description

Similar to the ACSP protocol, $ACSP^+$ is composed of two phases, i.e., SID Update and Tag Identification described seperately below:

SID update phase: This phase of protocol, which is depicted in Fig. 3, accomplishes as below:

1. The reader generates a random number R_1 and sends $(\overline{UPDSID}, R_1, H(\overline{UPDSID}, R_1, SID_{\mathrm{cur}}))$ to the tag.
2. Upon receipt of the message, each tag checks the values of $\overline{UPDSID}$, R_1 and $H(\overline{UPDSID}, R_1, SID_x)$ to verify whether the command is valid, where SID_x could be either of SID_{old} or SID_{cur}. If it is valid, the tag generates a random number R_2, updates local SID as $SID_{\mathrm{new}} = H(SID_{\mathrm{cur}})$ and sends $(\overline{UPDACK}, R_2, H(\overline{UPDACK}, R_2, R_1, SID_{\mathrm{cur}}))$ to the reader. The tag also keeps a history of SID_{cur} as SID_{old} and assigns SID_{new} to SID_{cur}.

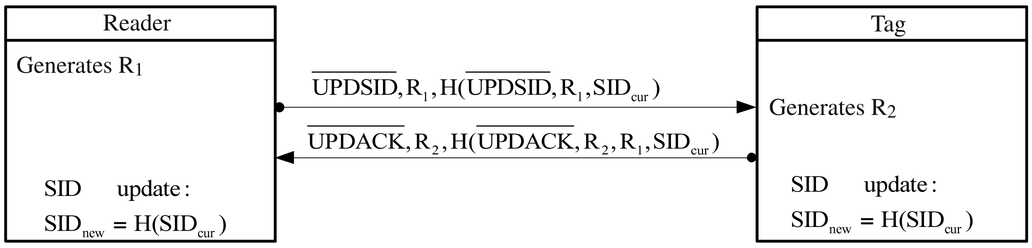
Reader
Generates R1
SID update:
$SID_{new} = H(SID_{cur})$
$\overline{UPDSID}, R_1, H(\overline{UPDSID}, R_1, SID_{cur})$
$\overline{UPDACK}, R_2, H(\overline{UPDACK}, R_2, R_1, SID_{cur})$
Tag
Generates R2
SID update:
$SID_{new} = H(SID_{cur})$

Fig. 3. The $ACSP^{+}$'s SID update phase.

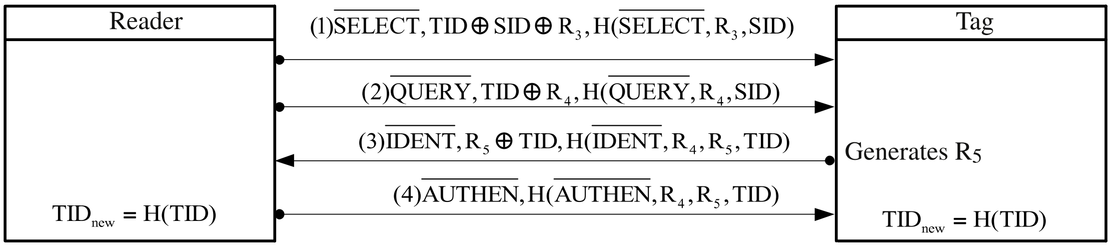
Reader
$TID_{new} = H(TID)$
$(1)\overline{SELECT}, TID \oplus SID \oplus R_3, H(\overline{SELECT}, R_3, SID)$
$(2)\overline{QUERY}, TID \oplus R_4, H(\overline{QUERY}, R_4, SID)$
$(3)\overline{IDENT}, R_5 \oplus TID, H(\overline{IDENT}, R_4, R_5, TID)$
$(4)\overline{AUTHEN}, H(\overline{AUTHEN}, R_4, R_5, TID)$
Tag
Generates R5
$TID_{new} = H(TID)$

Fig. 4. The $ACSP^{+}$'s tag identification phase.

3. Upon receipt of the message, the reader checks whether the command is valid. If it is valid, the reader updates local SID as $SID_{\text{new}} = H(SID_{\text{cur}})$.

Tag identification phase: This phase of protocol, which is depicted in Fig. 4, works as below:

1. The reader $\mathbb{R}$ generates a random number R_3 and sends $(\overline{SELECT}, TID \oplus SID \oplus R_3, H(\overline{SELECT}, R_3, SID))$ to the target tag.
2. Upon receipt of the message, any tag $\mathbb{T}_j$ extracts R'_3 from the received $TID \oplus SID \oplus R_3$ and verifies whether $H(\overline{SELECT}, R'_3, SID) \overset{?}{=} H(\overline{SELECT}, R_3, SID)$. If all are correct, the tag gets ready to respond; otherwise the tag keeps silent until receiving the next Select command.
3. $\mathbb{R}$ generates a random number R_4 and sends $\overline{QUERY}$, $TID \oplus R_4$ and $H(\overline{QUERY}, R_4, SID)$ to the target tag.
4. Upon receipt of the message, the ready tag $\mathbb{T}_i$ extracts R_4 from the received $TID \oplus R_4$ and checks the received $H(\overline{QUERY}, R_4, SID)$ in the Query command. If they are correct, it generates a random number R_5 and responds with $(\overline{IDENT}, TID \oplus R_5, H(\overline{IDENT}, R_4, R_5))$ as an identification message.
5. Upon receipt of the identification message without collision, $\mathbb{R}$ proceeds as follows:
   (a) It extracts R_5 from the received $TID \oplus R_5$ and checks the received $H(\overline{IDENT}, R_4, R_5)$ to authenticate the tag. If $\mathbb{R}$ authenticates the tag, it continues as follows:
   - Updates tag's identifier TID as $TID_{\text{new}} = H(TID)$. The reader also keeps a copy of TID_{old}.
   - Sends $(\overline{AUTHEN}, H(\overline{AUTHEN}, R_4, R_5, TID))$ to the tag.

   (b) Else, the reader sends $(\overline{QUERYREP})$ to end this slot and start a new one.
6. The tag receives $(\overline{AUTHEN}, H(\overline{AUTHEN}, R_4, R_5, TID))$ and upon receipt of these values, it checks $H(\overline{AUTHEN}, R_4, R_5, TID)$ and proceeds as follows:
   (a) If the received $H(\overline{AUTHEN}, R_4, R_5, TID)$ is valid, the tag updates its identifier as $TID_{\text{new}} = H(TID)$.
   (b) Else, it does nothing.

7.2. $ACSP^{+}$ security analysis

In this section, we present a detailed security analysis of $ACSP^{+}$ to show that the protocol meets resistance against the attacks presented in this paper and the other known active and passive attacks in the context.

7.2.1. Resistance against tag impersonation attack

The weakness of ACSP protocol against tag impersonation attack arises from this fact that random number R_5 does not have any effect on the calculation of $H(R_4, TID)$ and also TID_{new} is calculated as $H(R_4 \oplus R_5, TID)$. However, we fix this problem by changing the message of step 4 of ACSP protocol to $(\overline{IDENT}, TID \oplus R_5, H(\overline{IDENT}, R_4, R_5))$ and also updating the tag identifier as $TID_{\text{new}} = H(TID)$. Hence, given that nonces are contributed by both the reader and the tag, without the knowledge of the tag's secrets it is not feasible to impersonate the tag. It must be noted that the hash function can receive any arbitrary length messages and produces a fixed length output usually in an iterative structure (e.g. in Markle–Damgard structure [27,28]). So, a longer input to the hash function only increases the number of calls to its compression function and does not require further hardwares. In ACSP protocol the input length of hash functions is $2L$ while in $ACSP^{+}$ it is almost $4L$ where L is the length of protocol parameters.

Table 2
Performance comparison between ACSP and $ACSP^+$ in the SID Update phase. L denotes the bit length of parameters.

	# of calls to hash	# of $\oplus$	# transferred bits
Reader of ACSP	4	L	$5L$
Reader of $ACSP^+$	8	0	$3L$
Tag of ACSP	4	L	$3L$
Tag of $ACSP^+$	8	0	$3L$

Table 3
Performance comparison between ACSP and $ACSP^+$ in the Tag Identification phase. L denotes the bit length of parameters.

	# of calls to hash	# of $\oplus$	# transferred bits
Reader of ACSP	10	$2L$	$12L$
Reader of $ACSP^+$	15	$4L$	$8L$
Tag of ACSP	10	$2L$	$4L$
Tag of $ACSP^+$	15	$4L$	$3L$

7.2.2. *Resistance against single tag de-synchronization attack*

The single tag de-synchronization attack against ACSP is based on the fact that in step 4 of the ACSP protocol the random number R_5 does not affect any part of the transferred message, except the CRC value. Based on the reasoning on resistance against tag impersonation attack, $ACSP^+$ resists against single tag de-synchronization attack.

7.2.3. *Resistance against group de-synchronization attack*

In the $ACSP^+$ protocol, the random number R_1 has been used in the calculation of step 1 of the SID update phase and its integrity has been guaranteed by the sent $H(\overline{UPDSID}, R_1, SID_{\text{cur}})$ and SID is updated as $SID_{\text{new}} = H(SID_{\text{cur}})$. So if the adversary changes R_1 to R_1' and sends the modified message to tags, they will not accept it because $H(\overline{UPDSID}, R_1', SID_{\text{cur}})$ will not pass the versification with a high probability, i.e., $1 - 2^{-L}$. However, the adversary may block a session and replay the messages later. This approach also does not increase the adversary's advantage due to the updating approach which is used in $ACSP^+$, i.e., $SID_{\text{new}} = H(SID_{\text{cur}})$. Another weakness of the ACSP protocol which leads to a group de-synchronization attack is that in ACSP protocol any tag which updates its SID sends an update acknowledge command as $(\overline{UPDACK}, R_2, CRC)$ to the reader. However, this message can be generated by the adversary and also by the reader because it does not include any secret parameter. To fix this weakness, in $ACSP^+$, the SID update's acknowledgment is calculated as $(\overline{UPDACK}, R_2, H(\overline{UPDACK}, R_2, R_1, SID_{\text{cur}}))$ which includes the secret parameter SID_{cur} and also R_2 is a nonce which is contributed by the tag. Hence $ACSP^+$ provides suitable security against group de-synchronization attack.

7.2.4. *Resistance against traceability attack*

Traceability attack against the ACSP protocol is accomplished based on de-synchronizing of the tag $\mathbb{T}_i$ and the reader where through the de-synchronization attack the adversary achieves the required information to trace $\mathbb{T}_i$ later. Since the $ACSP^+$ protocol resists against single and group de-synchronization attacks it also resists against traceability attack.

7.3. *Performance analysis of $ACSP^+$ protocol*

In Tables 2 and 3 the performance comparison of ACSP and $ACSP^+$ protocols are provided in SID Update and Tag Identification phases respectively. These tables show that the proposed modifications do not increase the computational cost of the protocol extensively while it provides much better security. The only cost is some more iterations of the hash function's compression function.

8. Conclusion

In this paper we considered ACSP, an UHF RFID mutual authentication protocol. Based on its designers' claim this protocol is supposed to resist against counting attack and the other known attacks for RFID systems. However, we presented several efficient attacks against this protocol with a high success probability. Our attacks include tag impersonation attack, single tag de-synchronization attack, group of tags de-synchronization attack and traceability attack. The success probability of each attack is almost "1" and the complexity is at most three runs of the protocol. Hence, ACSP is not a secure protocol for ordinary applications of RFID systems.

In addition, we revised ACSP and entitled it $ACSP^+$, attempting to avoid the security flaws of its predecessor. Our detailed security analysis demonstrated significant security for the new scheme, which may be considered as a significant step toward designing a secure anti-counting RFID protocol.

References

[1] Pedro Peris-Lopez, Julio César Hernández Castro, Juan M. Estévez-Tapiador, Arturo Ribagorda, M^2AP: a minimalist mutual-authentication protocol for low-cost RFID tags, in: Jianhua Ma, Hai Jin, Laurence Tianruo Yang, Jeffrey J.P. Tsai (Eds.), Ubiquitous Intelligence and Computing, Third International Conference, in: Lecture Notes in Computer Science, vol. 4159, Springer, 2006, pp. 912–923.

[2] Pedro Peris-Lopez, Julio César Hernandez-Castro, Juan M. Estévez-Tapiador, Arturo Ribagorda, EMAP: an efficient mutual authentication protocol for low-cost RFID tags, in: OTM Federated Conferences and Workshop: IS Workshop—IS'06, in: LNCS, vol. 4277, Springer-Verlag, Montpellier, France, 2006, pp. 352–361.

[3] Tieyan Li, Employing lightweight primitives on low-cost RFID tags for authentication, in: VTC Fall, 2008, pp. 1–5.

[4] Chien Hung-Yu, SASI: a new ultralightweight RFID authentication protocol providing strong authentication and strong integrity, IEEE Transactions on Dependable and Secure Computing 4 (4) (2007) 337–340.

[5] Pedro Peris-Lopez, Julio César Hernández Castro, Juan M. Estévez-Tapiador, Arturo Ribagorda, Advances in ultralightweight cryptography for low-cost RFID tags: gossamer protocol, in: WISA, 2008, pp. 56–68.

[6] Mike Burmester, Breno de Medeiros, Jorge Munilla, Alberto Peinado, Secure EPC Gen2 compliant radio frequency identification, in: Pedro M. Ruiz, Jose Joaquin Garcia-Luna-Aceves (Eds.), ADHOC-NOW, in: Lecture Notes in Computer Science, vol. 5793, Springer, 2009, pp. 227–240.

[7] Tzu-Chang Yeh, Yan-Jun Wang, Tsai-Chi Kuo, Sheng-Shih Wang, Securing RFID systems conforming to EPC class 1 generation 2 standard, Expert Systems with Applications 37 (12) (2010) 7678–7683.

[8] Class-1 generation 2 UHF air interface protocol standard version 1.2.0, Gen2, 2008, in: Gen-2 Standard, EPCGlobal, 2008. http://www.epcglobalinc.org/standards/.

[9] EPC tag data standard version 1.4.2008. Yearly report on algorithms and key sizes, Technical Report D.SPA.13Rev.1.0, ICT-2007-216676, in: Gen2 Standard, ECRYPT, 2010. http://www.epcglobalinc.org/standards/.

[10] Eun Young Choi, Dong Hoon Lee, Jong In Lim, Anti-cloning protocol suitable to EPCglobal class-1 generation-2 RFID systems, Computer Standards & Interfaces 31 (6) (2009) 1124–1130.

[11] Nai-Wei Lo, Kuo-Hui Yeh, An efficient mutual authentication scheme for EPCglobal class-1 generation-2 RFID system, in: Mieso K. Denko, Chi-Sheng Shih, Kuan-Ching Li, Shiao-Li Tsao, Qing-An Zeng, Soo-Hyun Park, Young-Bae Ko, Shih-Hao Hung, Jong Hyuk Park (Eds.), EUC Workshops, in: Lecture Notes in Computer Science, vol. 4809, Springer, 2007, pp. 43–56.

[12] Y. Gu, W. Wu, A light-weight mutual authentication protocol for ISO 18000-6B standard RFID system, in: Proceedings of ICCTA 2009, 2009, pp. 21–25.

[13] Hung-Yu Chien, Che-Hao Chen, Mutual authentication protocol for RFID conforming to EPC class 1 generation 2 standards, Computer Standards & Interfaces 29 (2) (2007) 254–259.

[14] Chia-Hui Wei, Min-Shiang Hwang, A.Y. Chin, A mutual authentication protocol for RFID, IT Professional 13 (2) (2011) 20–24.

[15] Jian Shen, Dongmin Choi, Sangman Moh, Ilyong Chung, A novel anonymous RFID authentication protocol providing strong privacy and security, in: 2010 International Conference on Multimedia Information Networking and Security, MINES, 2010, pp. 584–588.

[16] Guang Gong, Yiyuan Luo, Qi Chai, Xuejia Lai, A lightweight stream cipher WG-7 for RFID encryption and authentication, in: IEEE Globecom 2010 Proceedings, 2010, pp. 1–6.

[17] Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, Daniel W. Engels, Security and privacy aspects of low-cost radio frequency identification systems, in: SPC, 2003, pp. 201–212.

[18] Jung-Sik Cho, Sang-Soo Yeo, Sung Kwon Kim, Securing against brute-force attack: A hash-based RFID mutual authentication protocol using a secret value, Computer Communications 34 (3) (2011) 391–397.

[19] Chiu Chiang Tan, Bo Sheng, Qun Li, Secure and serverless RFID authentication and search protocols, IEEE Transactions on Wireless Communications 7 (4) (2008) 1400–1407.

[20] Mihály Bárász, Balázs Boros, Péter Ligeti, Krisztina Lója, Dániel Nagy, Passive attack against the M2AP mutual authentication protocol for RFID tags. in: First International EURASIP Workshop on RFID Technology, Vienna, Austria, September 2007, pp. 1–4.

[21] Tieyan Li, Robert H. Deng, Vulnerability analysis of EMAP-an efficient RFID mutual authentication protocol, in: ARES, IEEE Computer Society, 2007, pp. 238–245.

[22] Tianjie Cao, Elisa Bertino, Hong Lei, Security analysis of the SASI protocol, IEEE Transactions on Dependable and Secure Computing 6 (1) (2009) 73–77.

[23] Pedro Peris-Lopez, Julio C. Hernández-Castro, Juan E. Tapiador, Jan C.A. van der Lubbe, Cryptanalysis of an EPC class-1 generation-2 standard compliant authentication protocol, Engineering Applications of Artificial Intelligence 24 (6) (2011) 1061–1069.

[24] Pedro Peris-Lopez, Tieyan Li, Julio C. Hernández-Castro, Juan E. Tapiador, Practical attacks on a mutual authentication scheme under the EPC class-1 generation-2 standard, Computer Communications 32 (7–10) (2009) 1185–1193.

[25] Masoumeh Safkhani, Nasour Bagheri, Majid Naderi, Yiyuan Luo, Qi Chai, Tag impersonation attack on two RFID mutual authentication protocols, in: ARES, 2011, pp. 581–584.

[26] Zhuzhong Qian, Ce Chen, Ilsun You, Sanglu Lu, ACSP: a novel security protocol against counting attack for UHF RFID systems, Computers & Mathematics with Applications 63 (2) (2012) 492–500.

[27] Ralph Markle, One way hash functions and DES, in: Gilles Brassard (Ed.), CRYPTO'89, in: LNCS, vol. 435, Springer-Verlag, 1990, pp. 428–446.

[28] Ivan Damgård, A design principle for hash functions, in: Gilles Brassard (Ed.), CRYPTO'89, in: LNCS, vol. 435, Springer-Verlag, 1990, pp. 416–427.