Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.

Volume 34, Issue 3, 15 March 2011 ISSN 0140-3664
ELSEVIER
computer
communications
Available online at www.sciencedirect.com
ScienceDirect
Special Issue:
Special Issue of Computer Communications on Information and Future Communication Security
Guest Editors: Jong Hyuk Park
Sheikh Iqbal Ahamed
Willy Susilo
Michela Taufer

This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

http://www.elsevier.com/copyright

Author's personal copy

Computer Communications 34 (2011) 384–390

Contents lists available at ScienceDirect

Computer Communications

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

Defending RFID authentication protocols against DoS attacks

Dang Nguyen Duc [*], Kwangjo Kim

Auto-ID Lab Korea, Department of Information and Communications Engineering, KAIST, 119 Munjiro, Yuseong-gu, Daejeon 305-732, Republic of Korea

A R T I C L E I N F O

Article history:
Available online 22 June 2010

Keywords:
RFID security
Denial-of-service attack
Forward security

A B S T R A C T

In this paper, we present a security weakness of a forward secure authentication protocol proposed by Tri Van Le et al. called O-FRAP which stands for *Optimistic Forward secure RFID Authentication Protocol*. In particular, we point out that in the O-FRAP protocol, the server can be subject to a denial-of-service attack due to a flaw in the database querying procedure. Our attack also applies to a simplified version of O-FRAP called O-RAP (*Optimistic RFID Authentication Protocol*) which is essentially O-FRAP but without a secret key updating procedure (and thus forward security). We then propose two improved protocols called O-FRAP$^+$ and O-RAP$^+$ which prevent the said denial-of-service attack. In addition, the O-FRAP$^+$ protocol also addresses two security weaknesses of O-FRAP pointed out earlier by Khaled and Raphael. In terms of performance, comparing to O-FRAP, O-FRAP$^+$ requires a few more computational steps but much less storage at the back-end server.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Radio Frequency Identification (RFID) is an emerging technology which promises automatic item tracking. The key idea is to attach each and every item with an RFID tag which can be read by RFID readers via radio communication. Each RFID tag is a cheap device capable of emitting a unique number which will be served as the identification information of an RFID-tagged item in a database at a back-end server.

Unfortunately, a widespread adoption of RFID is uncertain because of its inherent security weaknesses which includes tag cloning and privacy violation. It turns out that these two security weaknesses come from the very basic operation of an RFID tag, that is to communicate the identification of an RFID-tagged item (hereafter referred to as *Electronic Product Code* or EPC for short). This is an inherent security risk since we depend on the EPC number to recognize a product as genuine or fake. An attacker equipped with a compatible reader can scan many RFID tags to harvest a large number of EPC numbers. He then can produce RFID tags which emit exactly the same EPCs he has collected. This kind of tags are called *cloned tags*. The cloned tags can be attached to counterfeited items which should be recognized as genuine items. The core functionality of an RFID tag also raises privacy concern. As each EPC number is unique, an attacker with a compatible reader can recognize and track RFID tags which leads to privacy violation of a person carrying tagged items.

To deal with the security problems of RFID, the use of cryptographic protocols is required. However, designing cryptographic protocols for RFID tags is challenging as an RFID tag is a low-cost device with limited computational power. It is infeasible to implement public key cryptographic primitives and block ciphers. As a result, a new approach to design cryptographic protocols for RFID tags which employ only *lightweight* primitives is required. The most popular lightweight primitive used in designing cryptographic protocols for RFID is hash function.

We now describe desirable security properties of a cryptographic protocol for RFID as follows [1–5].

- *Mutual authentication between tag and reader/back-end server*: In order to prevent EPC numbers from being harvested by malicious parties, reader-to-tag authentication must be provided. In addition, the server should not waste its computational resource on verifying and identifying fake tags. Therefore, RFID readers should also authenticate tags before forwarding legitimate tags to the server for identification.
- *Privacy-preserving authentication:* In order to prevent a tag from being tracked by malicious parties, it is not sufficient to avoid communicating the tag's EPC number in clear text. Indeed, the information exchanged during different authentication sessions should not help a malicious party to trace a tag. We call such a property *unlinkability*. We refer to a protocol that provides both secure authentication and unlinkability as *a privacy-preserving authentication protocol*. A common approach to provide privacy-preserving authentication is to use pseudonym. More specifically, for each authentication session, a tag uses a different *temporary identity* called pseudonym to communicate with an RFID reader.
- *Forward security*: As an RFID tag is generally not a tamper-proof device, it can be easily stolen and dissected to reveal secret information stored in the memory of the tag. Many

* Corresponding author. Tel.: +82 42 866 6236; fax: +82 42 866 6273.
E-mail addresses: nguyenduc@icu.ac.kr (D.N. Duc), kkj@cs.kaist.ac.kr (K. Kim).

0140-3664/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.06.014

Author's personal copy

authentication protocols for RFID including [6] have taken this threat into account by providing a security property called *forward security*. In the case of a privacy-preserving authentication protocol, forward security guarantees that all of authentication sessions of a tag happened before the tag's secret is revealed remain unlinkable. In other words, the privacy of the tag is protected up to the point of the loss of the secret information. A well-known method to achieve forward security is to update the secret key frequently (say, after every authentication session). Once a secret key is revealed, the previous authentication sessions that are associated with old and unknown secret keys are unlinkable. Updating secret keys regularly might also have positive impact on providing privacy-preserving as a tag possesses different keys during different authentication sessions. Unfortunately, updating the secret key interactively between a reader and a tag is often subject to de-synchronization of secret, i.e., the attacker can cause the reader and the tag to posses different keys which makes future communication impossible.

In this paper, we analyze the security of a forward secure RFID authentication protocol called O-FRAP [6] which stands for *Optimistic Forward secure RFID Authentication Protocol*. O-FRAP is shown to achieve privacy-preserving mutual authentication and forward security under a security model called *Universal Composable Security Framework* (UC framework for short) [7]. The UC framework guarantees not only the security of a protocol running in isolation, but also the security of the same protocol running as a component of a bigger system. Our analysis of O-FRAP shows that it is vulnerable to a denial-of-service attack (DoS for short). Our attack can be summarized as follows:

- First, an attacker queries a large number of tags but terminates each protocol session prematurely. The goal of this phase is to cause each tag to possess a different version of pseudonym from the one stored in the server's database.
- Then, the server's computational resources can be abused by querying tags with de-synchronized pseudonyms. The attacker can also attack the server directly by rendering itself as a tag but sending invalid pseudonyms to the server.

Note that, in [9], the authors of O-FRAP presented a simplified version of O-FRAP called *Optimistic RFID Authentication Protocol* (O-RAP for short) which does not provide forward security. However, O-RAP is also vulnerable to the DoS attack presented in this paper. We then propose an enhanced protocol called O-FRAP$^+$ to counter our attack. Comparing to O-FRAP, O-FRAP$^+$ introduces insignificant computational overhead but requires much less memory storage at the server side. Our improved design can also apply to the case of O-RAP protocol to make O-RAP resistant against the DoS attack presented in this paper.

This paper is organized as follows: we first review O-FRAP and O-RAP in Section 2. Then, we present the vulnerability of O-FRAP and O-RAP against DoS attack in Section 3. The improved protocol O-FRAP$^+$ (and the enhanced version of O-RAP which we call O-RAP$^+$) is presented in Section 4, followed by security analysis and comparison in Section 5. Finally, we end with conclusion in Section 6.

2. Related works

2.1. Notation

Throughout this paper, we will use the notations summarized in Table 1.

Table 1
Summary of notations.

Notation	Description
D	Database of tags kept by server
$D.query(.)$	Database querying procedure
$D.update(.)$	Key updating procedure at database
n	Number of tags in D
k_{tag}	Shared secret between tag and server
r_{tag}	Tag pseudonym
l	Bit length of k_{tag} and r_{tag}
$Prev_i$	(k_{tag}, r_{tag}) of tag i stored in D which was used in one of previous sessions
Cur_i	(k_{tag}, r_{tag}) of tag i stored in D which is currently used
$instance(i)$	A pair of the form (k_{tag}, r_{tag}) of tag i
k_S	Fixed key shared between tag and server
$f(.)$	Pseudorandom function

2.2. O-FRAP and O-RAP protocols

O-FRAP is an authentication protocol for RFID in which a back-end server authenticates and identifies RFID tags. Each tag is numbered from 1 to n and all of tag information is stored in a database D at the back-end server. Note that, the RFID reader is omitted in the description of O-FRAP as it essentially just plays the role of an intermediate party who relays messages exchanged between a tag and the server. A detailed description of O-FRAP is given in Fig. 1.

We now discuss how O-FRAP achieves unlinkability and forward security. Each tag shares with the server a secret key denoted by k_{tag}. To protect a tag against malicious tracking, for each authentication session, a tag uses a randomly chosen number r_{tag} as its pseudonym. The tag pseudonym is stored in both the memory of the tag and the server's database D. The goal is to use the pseudonym to index D and quickly look up information on a tag given its pseudonym (the $D.query(.)$ procedure).

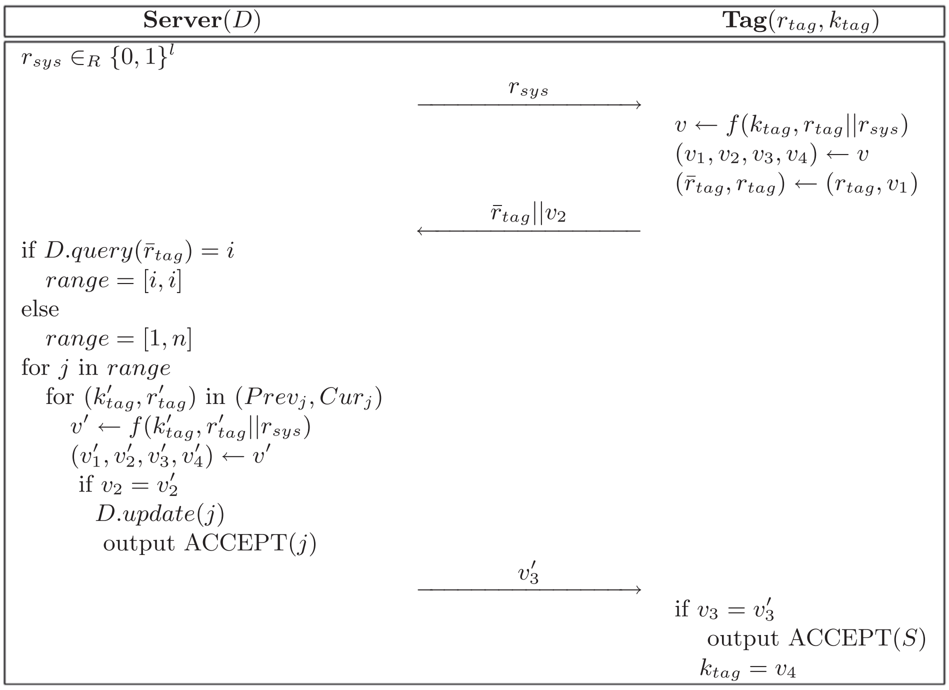
To achieve forward security, k_{tag} is updated after every successful authentication session, both at the tag and the server sides. In O-FRAP, the tag updates its key in the last round only if it successfully verifies the server. Therefore, an active attacker can intercept and modify the server's authentication token causing the tag fails to verify the server and not to update its key. To prevent de-synchronization attack, the server keeps two versions of secret key for each tag in its database, a previously-used key (denoted by k_{tag}^{prev}) and a currently-used one (denoted by k_{tag}^{cur}). The server updates the two keys in the $D.update(.)$ procedure as follows:

- If the tag is authenticated with k_{tag}^{cur}, the server does: $k_{tag}^{prev} = k_{tag}^{cur}$ and $k_{tag}^{cur} = k_{tag}^{new}$ where k_{tag}^{new} is a newly generated key.
- If the tag is authenticated with k_{tag}^{prev}, the server preserves k_{tag}^{prev} and lets $k_{tag}^{cur} = k_{tag}^{new}$. The server does not update k_{tag}^{prev} because an active attacker can cause de-synchronization of secret by modifying the server's authentication token ν_3' in two consecutive sessions.

The server also maintains two versions of the pseudonym for each tag. Each entry in D which corresponds to one tag is indexed by two pseudonyms. The two pseudonyms are also updated in the same fashion as the secret keys are. We denote $Prev_i$ and Cur_i as two instances of tag information, each of the form (Secret Key, Pseudonym), for a tag numbered i in D.

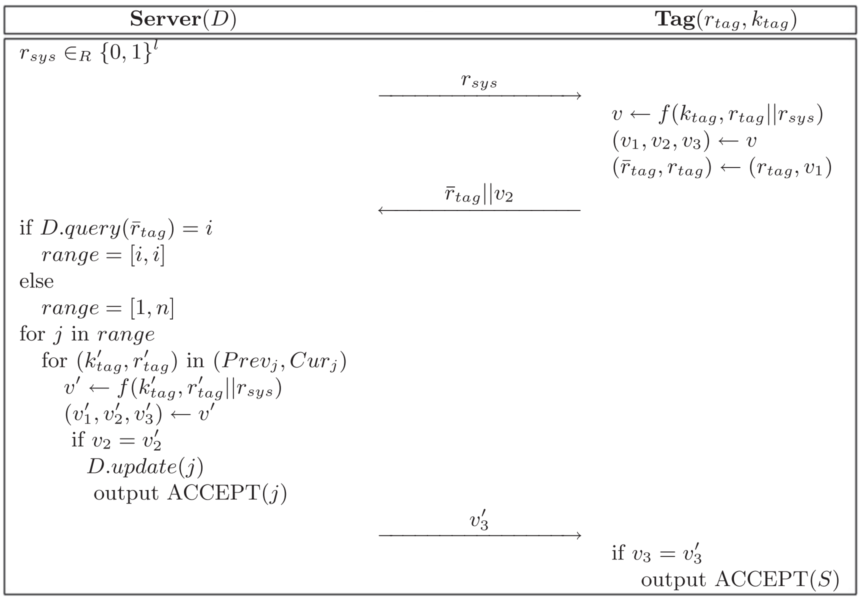
O-RAP which stands for *Optimistic RFID Authentication Protocol* is a simplified version of O-FRAP which appeared in [9]. O-RAP is essentially O-FRAP but without a key updating procedure. As a result, O-RAP does not provide forward security and the back-end server does not need to store two versions of the shared secret key for each tag. A detailed description of O-RAP is given in Fig. 2.

Author's personal copy



```
Server(D)                                                    Tag(r_tag, k_tag)
$r_{sys} \in_R \{0,1\}^l$
                         $\xrightarrow{r_{sys}}$
                                                     $v \leftarrow f(k_{tag}, r_{tag}||r_{sys})$
                                                     $(v_1, v_2, v_3, v_4) \leftarrow v$
                                                     $(\bar{r}_{tag}, r_{tag}) \leftarrow (r_{tag}, v_1)$
                         $\xleftarrow{\bar{r}_{tag}||v_2}$
if $D.query(\bar{r}_{tag}) = i$
  $range = [i, i]$
else
  $range = [1, n]$
for $j$ in $range$
  for $(k'_{tag}, r'_{tag})$ in $(Prev_j, Cur_j)$
    $v' \leftarrow f(k'_{tag}, r'_{tag}||r_{sys})$
    $(v'_1, v'_2, v'_3, v'_4) \leftarrow v'$
    if $v_2 = v'_2$
      $D.update(j)$
      output ACCEPT$(j)$
                         $\xrightarrow{v'_3}$
                                                     if $v_3 = v'_3$
                                                       output ACCEPT$(S)$
                                                       $k_{tag} = v_4$
```

Fig. 1. The O-FRAP protocol.



```
Server(D)                                                    Tag(r_tag, k_tag)
$r_{sys} \in_R \{0,1\}^l$
                         $\xrightarrow{r_{sys}}$
                                                     $v \leftarrow f(k_{tag}, r_{tag}||r_{sys})$
                                                     $(v_1, v_2, v_3) \leftarrow v$
                                                     $(\bar{r}_{tag}, r_{tag}) \leftarrow (r_{tag}, v_1)$
                         $\xleftarrow{\bar{r}_{tag}||v_2}$
if $D.query(\bar{r}_{tag}) = i$
  $range = [i, i]$
else
  $range = [1, n]$
for $j$ in $range$
  for $(k'_{tag}, r'_{tag})$ in $(Prev_j, Cur_j)$
    $v' \leftarrow f(k'_{tag}, r'_{tag}||r_{sys})$
    $(v'_1, v'_2, v'_3) \leftarrow v'$
    if $v_2 = v'_2$
      $D.update(j)$
      output ACCEPT$(j)$
                         $\xrightarrow{v'_3}$
                                                     if $v_3 = v'_3$
                                                       output ACCEPT$(S)$
```

Fig. 2. The O-RAP protocol.

Note that, Khalil and Raphael pointed out that the forward security of O-FRAP$^+$ can be violated because the tag outputs ACCEPT before updating its secret key. Therefore, if an attacker corrupts the tag just before the tag's secret key is updated, the immediate previous authentication session can be linked to the current session. We also would like to remark that in O-FRAP, the tag updates its secret key only if the server is authenticated successfully. This potentially defeats forward security because an attacker can modify v'_3 to cause the server authentication to fail (which leads to the tag not to update its secret key). Another weakness of O-FRAP noticed in [8] is that the privacy-preserving property can be violated. More specifically, an attacker can trick a tag into updating its pseudonym but not its secret key in one session and then he will be able to trace the tag in the immediate following session. Note that, the two attacks presented in [8] are not quite practical as only two consecutive sessions can be linked.

3. Denial-of-service attack on O-FRAP and O-RAP

We now present a DoS attack on O-FRAP. The attack also works on O-RAP as the two protocols share the same design. In O-FRAP, the server will scan through the whole tag population in D if it fails to single out one tag in D given a pseudonym of a tag, $\bar{r}_{tag}$. An attacker can exploit this property by sending a bogus $\bar{r}_{tag}$, say $\bar{r}^*_{tag}$

Author's personal copy

to the server and thus abuses the computational resources of the server. A widespread presence of fake tags can make the problem even more serious. One may argue that if the server fails to locate a single tag in *D*, it means that an attack is detected. In addition, the fact that the server tries to match a tag with all the available tags in its database is simply to make the protocol complete. However, O-FRAP and O-RAP were designed to function like that for a different reason which was not mentioned in [6,9]. The actual reason is that it is straightforward to cause de-synchronization of pseudonym between a tag and the server. As a tag always updates its pseudonym regardless of being queried by a legitimate or malicious reader, the pseudonym can be easily de-synchronized by an attacker who sends arbitrary query requests to the tag. In order to accommodate an RFID tag whose pseudonym has been de-synchronized, the server needs to match that tag with each and every entry in its database. The attack is illustrated in Fig. 3 where an attacker queries the tag with r^*_{sys} to cause de-synchronization of pseudonym. Then, both the tag and the attacker can cause the server to scan through the whole database *D*. We also want to remark that the de-synchronization of pseudonym always implies that the pseudonym at the tag is ahead of the pseudonym at the server. Therefore, keeping a pseudonym used in one of previous session and indexing *D* with this value are useless.

There are several ways to prevent the above attack as follows:

- An RFID tag should avoid updating its pseudonym and terminate an authentication session if it is queried by an unknown server. Note that, in order to provide unlinkability, the pseudonym of a tag should not be sent before the server is authenticated.
- If an attacker attempts to send an invalid pseudonym, the server should be able to detect it and take appropriate measures, e.g., stop the authentication session and examine the tag in a physically secure location. In case of O-FRAP, the server cannot distinguish whether a tag in question suffers from de-synchronization of pseudonym or its pseudonym has been actively modified by the attacker.

As we will see below, the two enhanced protocols, O-FRAP$^+$ and O-RAP$^+$, take both of the above approaches into account.

4. O-FRAP$^+$ and O-RAP$^+$ protocols

We now present the first improved protocol, O-FRAP$^+$, which aims to solve the security issues with O-FRAP mentioned above. First, we shall discuss how to address those security issues and then describe the construction of O-FRAP$^+$.

4.1. Main idea

In order to prevent an RFID tag from updating its pseudonym accidentally, the server needs to be authenticated first. This cannot be done in O-FRAP as the server has to look up a secret key shared with an unknown tag before it can compute the authentication token ν'_3. Note that, there is no loss of security if a tag uses one key to authenticate the server and another key to prove its identity. Therefore, we can use another common and fixed key to authenticate the server. This key can be common for all tags or a local group of tags (e.g., the tag database is partitioned and distributed) so that the server does not have to search for this key first. Let's call this key k_S. Authenticating the server first has another benefit as the tag can now update its secret key before the server. The de-synchronization attack can still be possible but in this case the problem is much easier to handle without the need for storing two version of keys for each tag. To detect whether a pseudonym has been tampered with before reaching the server, a tag can attach to its pseudonym an integrity-checking message with the secret key k_S. In other words, the tag is authenticated by the server in two steps: first, the server verifies that the tag is in its database with k_S; then the tag is identified with its own secret key k_{tag}. In practice, a reader can use the secret key k_S to filter out fake tags. Only tags that pass the first authentication step using k_S (i.e., tags that are actually in the database) can be forwarded to the back-end server. Then, the back-end server will use the second authentication step to authenticate and identify the tags.

4.2. Construction

In O-FRAP$^+$, each tag shares with the server two keys, a common secret key k_S and a private secret key k_{tag}. The database *D* is indexed with only currently-used pseudonyms of tags. The protocol consists of four rounds roughly described as follows:

- Round 1: The server broadcasts its querying request r_{sys}.
- Round 2: The tag then challenges the server with t_{sys}.
- Round 3: The server sends its response which will be verified by the tag.
- Round 4: After authenticating the server, the tag updates its pseudonym and secret key. Then it sends its old pseudonym and its authentication token to the server so that the server can authenticate and identify the tag. Note that, in response to the attack in [8], the tag should updates its secret key and pseudonym before accepting the server.

A detail description of O-FRAP$^+$ is given in Fig. 4 where a C language convention *return* statement is used instead of *output* used in the description of O-FRAP. Note that, O-FRAP$^+$ is a 4-round protocol for a purely practical reason. In practice, an RFID reader is usually the one to initiate an authentication session.

We now discuss the key updating procedure at the server side. After successfully verifying that the tag is in *D*, it is likely that the *D.query*(.) will succeed and return one entry in *D*. Let *instance*(*i*) be

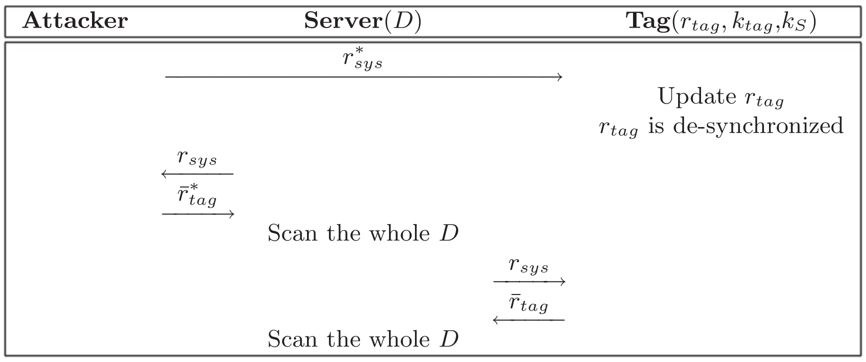
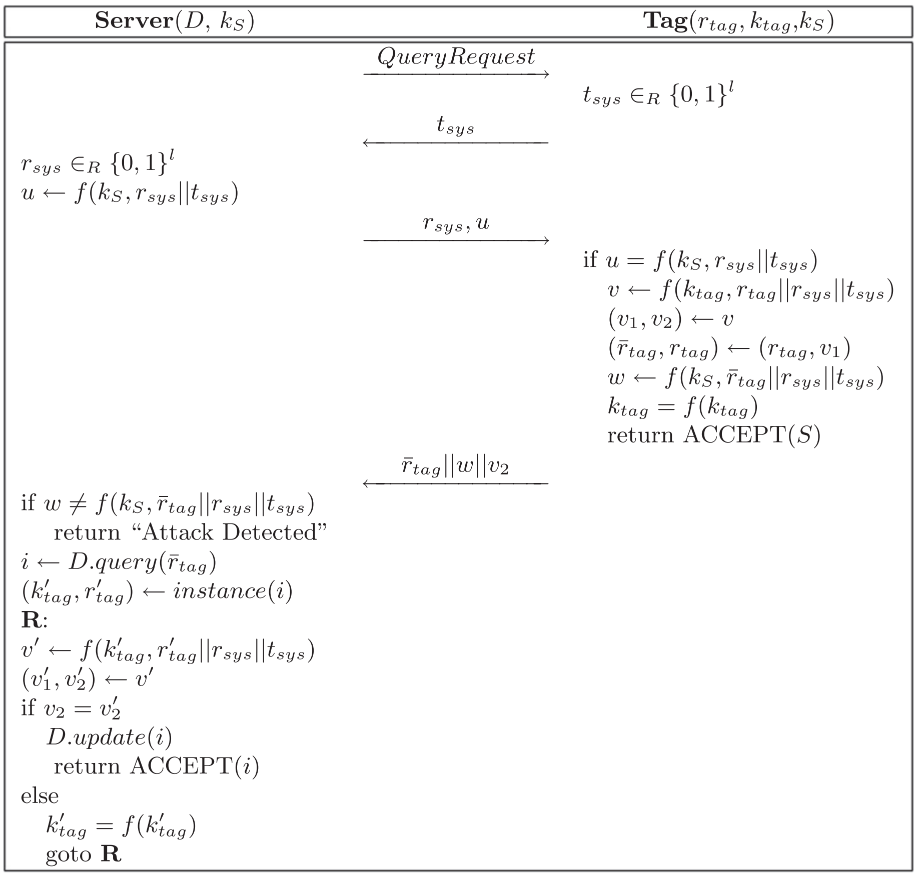


Fig. 3. Denial-of-service attack on O-FRAP and O-RAP.

Author's personal copy



```
Server(D, k_S)                                              Tag(r_tag, k_tag, k_S)
                          QueryRequest →
                                                            $t_{sys} \in_R \{0,1\}^l$
                          ← $t_{sys}$
$r_{sys} \in_R \{0,1\}^l$
$u \leftarrow f(k_S, r_{sys}||t_{sys})$
                          $r_{sys}, u$ →
                                                            if $u = f(k_S, r_{sys}||t_{sys})$
                                                              $v \leftarrow f(k_{tag}, r_{tag}||r_{sys}||t_{sys})$
                                                              $(v_1, v_2) \leftarrow v$
                                                              $(\bar{r}_{tag}, r_{tag}) \leftarrow (r_{tag}, v_1)$
                                                              $w \leftarrow f(k_S, \bar{r}_{tag}||r_{sys}||t_{sys})$
                                                              $k_{tag} = f(k_{tag})$
                                                              return ACCEPT($S$)
                          ← $\bar{r}_{tag}||w||v_2$
if $w \neq f(k_S, \bar{r}_{tag}||r_{sys}||t_{sys})$
    return "Attack Detected"
$i \leftarrow D.query(\bar{r}_{tag})$
$(k'_{tag}, r'_{tag}) \leftarrow instance(i)$
R:
$v' \leftarrow f(k'_{tag}, r'_{tag}||r_{sys}||t_{sys})$
$(v'_1, v'_2) \leftarrow v'$
if $v_2 = v'_2$
   $D.update(i)$
    return ACCEPT($i$)
else
   $k'_{tag} = f(k'_{tag})$
   goto R
```

Fig. 4. The O-FRAP⁺ protocol.

a (Secret Key, Pseudonym)=(k_{tag},r_{tag}) pair of the tag *i* stored in *D*. Then, the server can authenticate the tag with the key k_{tag}. However, it is still possible to cause de-synchronization of k_{tag} in O-FRAP$^+$. By modifying v_2, an attacker can cause tag authentication with k_{tag} to fail which results in the server not to update its version of k_{tag}. Note that, the de-synchronization of secret in O-FRAP$^+$ is very different from the problem in O-FRAP. In O-FRAP, the de-synchronization of k_{tag} means the tag still keeps a key used in one of previous authentication sessions while the server keeps the currently-used key. Whereas, in O-FRAP$^+$, if de-synchronization of k_{tag} occurs then the server keeps one of the old keys while the tag has the latest key. Furthermore, even though k_{tag} is inconsistent between the server and the tag, the server can still locate the candidate tag in its database. It is clearly not the case in O-FRAP. This is why we do not need to store two versions of secret key for each tag in *D*. To accommodate a tag whose k_{tag} has been de-synchronized, we update k_{tag} in a chaining fashion, i.e., $k_{tag}^{new} = f(k_{tag}^{cur})$.[1] The server can try a new $k_{tag} = f(k_{tag})$ to re-authenticate the tag once it fails to authenticate the tag with its current version of k_{tag}. The number of times the server tries in this scenario is up to a specific deployment of O-FRAP$^+$.

Using the same approach described above, we can also secure O-RAP against the denial-of-service attack presented in this paper. We call O-RAP$^+$ as the secure version of O-RAP. Note that, O-RAP does not have a key updating procedure. Therefore, there is no key updating procedure as well as *goto* statement to handle the de-synchronization of secret problem in O-RAP$^+$. O-RAP$^+$ is illustrated in Fig. 5.

5. Security analysis and comparison

5.1. Secure mutual authentication

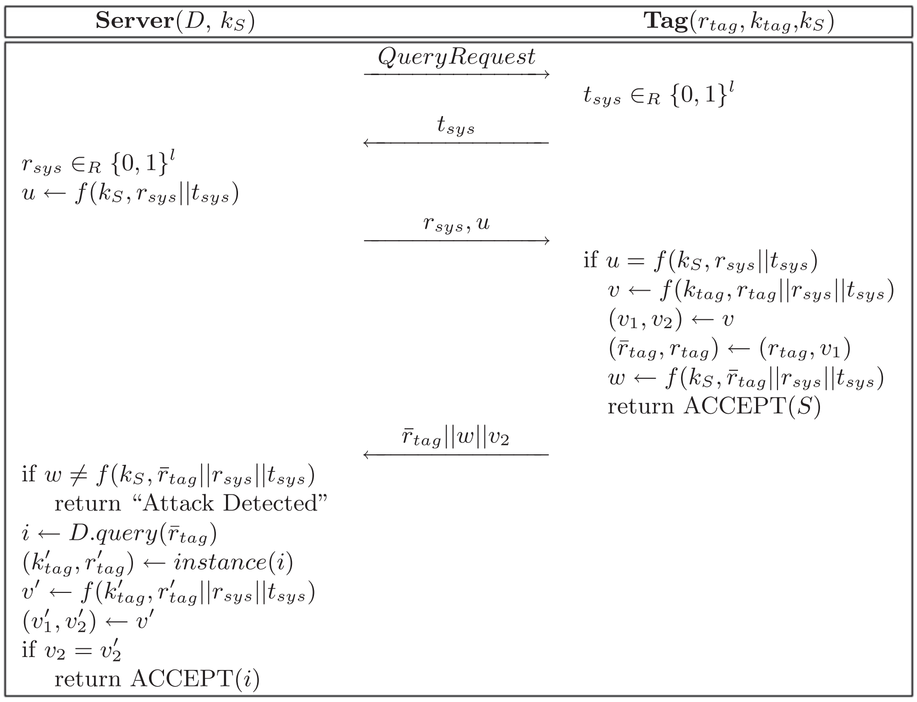
We can see that the mechanisms to provide mutual authentication in O-FRAP and O-FRAP$^+$ are essentially the same (this is also true for O-RAP and O-RAP$^+$). More specifically, an authentication token is composed of a random nonce and the output from *F*(.) with a shared secret key and the random nonce as the input. The only difference is that in O-FRAP$^+$ and O-RAP$^+$, the keys to authenticate the server and a tag are different. Therefore, if an attacker can violate the security of mutual authentication in the O-FRAP$^+$ and O-RAP$^+$ protocols, it can do the same for the O-FRAP and O-RAP protocols.

5.2. Privacy-preserving

Both O-FRAP$^+$ and O-RAP$^+$ are secure against the tracing attack presented in [8]. The reason is that the tag pseudonym and secret key are updated at the same time. Therefore, an attacker cannot cause a tag to update its pseudonym but not its secret key. Note that, the tag pseudonym is emitted only after the server is verified so not updating the pseudonym after every session does not make a tag vulnerable to tracing attack. In addition, the fact that a tag stops the protocol prematurely (i.e., the server is not successfully authenticated) might have positive impact on privacy protection in practice because it limits the ability of an attacker to detect

[1] If the output length of *f*(.) is longer than *l*, we can take the first *l bits the output of f*(.).

Author's personal copy



```
Server($D$, $k_S$)                                    Tag($r_{tag}$, $k_{tag}$, $k_S$)
                        $\xrightarrow{QueryRequest}$
                                                      $t_{sys} \in_R \{0,1\}^l$
                        $\xleftarrow{t_{sys}}$
$r_{sys} \in_R \{0,1\}^l$
$u \leftarrow f(k_S, r_{sys}||t_{sys})$
                        $\xrightarrow{r_{sys}, u}$
                                                      if $u = f(k_S, r_{sys}||t_{sys})$
                                                          $v \leftarrow f(k_{tag}, r_{tag}||r_{sys}||t_{sys})$
                                                          $(v_1, v_2) \leftarrow v$
                                                          $(\bar{r}_{tag}, r_{tag}) \leftarrow (r_{tag}, v_1)$
                                                          $w \leftarrow f(k_S, \bar{r}_{tag}||r_{sys}||t_{sys})$
                                                          return ACCEPT($S$)
                        $\xleftarrow{\bar{r}_{tag}||w||v_2}$
if $w \neq f(k_S, \bar{r}_{tag}||r_{sys}||t_{sys})$
    return "Attack Detected"
$i \leftarrow D.query(\bar{r}_{tag})$
$(k'_{tag}, r'_{tag}) \leftarrow instance(i)$
$v' \leftarrow f(k'_{tag}, r'_{tag}||r_{sys}||t_{sys})$
$(v'_1, v'_2) \leftarrow v'$
if $v_2 = v'_2$
    return ACCEPT($i$)
```

Fig. 5. The O-RAP$^+$ protocol.

Table 2
Comparison of O-FRAP, O-FRAP$^+$ and O-RAP$^+$.

	O-FRAP	O-FRAP$^+$	O-RAP$^+$
No. of $f(.)$ evaluation	Tag: 1 Server: 1	Tag: 4 Server: 4	Tag: 3 Server: 3
Key length	Tag: $2l$ bits Server: $4ln$ bits	Tag: $3l$ bits Server: $(2n+1)l$ bits	Tag: $3l$ bits Server: $(2n+1)l$ bits
Mutual authentication	Yes	Yes	Yes
Privacy protection	Weak	Strong	Strong
Resistant against DoS	No	Yes	Yes

the presence of RFID tags. For instance, a malicious party may attempt to look for a particular RFID tag embedded in a passport to determine the nationality of the passport holder. However, because the tag terminates the protocol early, the malicious party might not have enough information on the protocol signature to decide the presence or origin of the tag.

5.3. Forward security

O-FRAP$^+$ prevents the attack in [8] by requiring a tag to update its secret key before accepting the server as authenticated. Furthermore, in O-FRAP$^+$, an attacker cannot cause a tag not to update its secret key even in case k_S is corrupted. It is different from O-FRAP where an attacker can modify the server authentication token v'_3 so that a tag will not update its secret key. As updating secret is required to achieve forward security, O-FRAP$^+$ is at least as secure as O-FRAP.

5.4. Resistant against DoS attack

O-FRAP$^+$ and O-RAP$^+$ prevent fake tags from abusing the server's computational resources by verifying the integrity of pseudonyms. In addition, an attacker cannot cause a tag to update its pseudonym leading to inconsistent pseudonyms between tags and the server (which makes the server to scan through the whole tag database when querying tags in O-FRAP and O-RAP protocols).

5.5. Comparison

The comparison in terms of required computational resource and security features of O-FRAP, O-FRAP$^+$ and O-RAP$^+$ is given in Table 2.

6. Conclusion

In this paper, we show that the provably secure RFID authentication protocol called O-FRAP and its simplified version O-RAP are vulnerable to DoS attack. We pointed out that it is trivial to abuse the server's computational resources in O-FRAP and O-RAP. We then presented our improved protocols of O-FRAP and O-RAP which we call O-FRAP$^+$ and O-RAP$^+$, respectively. The most important point in the design of O-FRAP$^+$ and O-RAP$^+$ is that the server should be authenticated first by using a fixed secret key. By doing so, we not only defend O-FRAP$^+$ against DoS attack but also remove the need for storing two versions of secret key and pseudonym for each tag in the server's database.

We think our approach of two-phase authentication can be applied to many other RFID authentication protocols which use a similar design to O-FRAP (that is the tag is authenticated and identified first). In some applications where mutual authentication between tag and server/reader is not needed, using two-phase authentication can still help a reader to prevent unwanted information to travel to the back-end server.

Acknowledgement

This research was supported by the ICT Standardization program of MKE (The Ministry of Knowledge Economy). The authors also thank anonymous reviewers of this paper for providing useful comments.

Author's personal copy

References

[1] Stephen Weis, Security and privacy in radio frequency identification devices, Master's Thesis. Available at <http://theory.lcs.mit.edu/sweis/masters.pdf>, May 2003.
[2] Miyako Ohkubo, Koutarou Suzuki, Shingo Kinoshita, Efficient hash-chain based RFID privacy protection scheme, in: the Proceedings of International Conference on Ubiquitous Computing, Workshop Privacy, September 2004.
[3] Ari Juels, Stephen Weis, Authenticating pervasive devices with human protocols, in: the Proceedings of CRYPTO'05, in: Victor Shoup (Ed.), LNCS, 3261, Springer-Verlag, 2005, pp. 293–308.
[4] Ari Juels, Strenthening EPC tag against cloning, in: M. Jakobsson, R. Poovendran (Eds.), The Proceedings of ACM Workshop on Wireless Security (WiSe), 2005, pp. 67–76.
[5] Ari Juels, RFID security and privacy: a research survey, Journal of Selected Areas in Communication (J-SAC) 24 (2) (2006) 381–395.
[6] Tri Van Le, Mike Burnmester, Breno de Medeiros, Universally composable and forward secure RFID authentication and authenticated key exchange, in: The Proceedings of the Second ACM Symposium on Information, Computer and Communications Security, March 2007, pp. 242–252.
[7] Ran Canetti, Obtaining universally composable security: towards the bare bones of trust. Available at <http://eprint.iacr.org/2007/475>.
[8] Khaled Ouafi, Raphael C.-W. Phan, Traceable privacy of recent provably-secure RFID protocols, in: the Proceedings of ACNS 2008, LNCS, 5037, Springer-Verlag, 2008, pp. 479–489.
[9] Mike Burnmester, Tri Van Le, Brene De Medeiros, Gene Tsudik, Universally composable RFID identification and authentication protocols, ACM Transactions on Information and Systems Security 12 (4) (2009) (Article 21).