S^{3}PR: Secure Serverless Search Protocols for RFID

Sheikh I. Ahamed[1], Farzana Rahman[1], Endadul Hoque[1], Fahim Kawsar[2], and Tatsuo Nakajima[2]

[1]*Ubicomp lab, MSCS Dept., Marquette University, USA*
[2]*Dept. of Computer Science, Waseda University, Japan*

iq@mscs.mu.edu, {bondhons084, endadulhoque }@yahoo.com
{fahim,tatsuo}@dcl.info.waseda.ac.jp

Abstract

For RFID tags to proliferate in our day to day life, they will have to offer practical, low cost and secured mechanisms for tag authentication which has been in the midst of researcher's interest for almost a decade. One extension of RFID authentication is RFID tag searching, which has not been given much attention so far. But we firmly believe that in near future tag searching will be a significant issue. In this paper we propose a lightweight and serverless RFID tag searching protocol. This protocol can search a particular tag efficiently without server's intervention. Furthermore they are secured against major security threats.

Keywords: RFID security, serverless search protocol

1. Introduction

RFID systems (referring to Radio Frequency IDentification) embrace one important development track in the framework of ubiquitous or pervasive computing. It is an automatic identification system, which relies on storing and remotely retrieving data on objects by using RFID tags. Tags equipped within objects have unique identification information and is applicable in various fields such as supply chain management, employee identification, product maintenance etc.

In all such practical implementations, often a reader needs to determine whether a particular tag exists within a group of tags. This is referred to as RFID searching. In fact RFID searching is an extension of RFID authentication. By authenticating every tag within a group, we can find out the desired tag. But this is very inefficient. Tag searching with the help of central database will not be a challenging issue. But without the help of server, the reader has to search a tag entirely by itself. This is a critical task because it is vulnerable to privacy and security threats [3]. For example, through the broadcast of a search query, a reader in a warehouse wants to search for a tag which belongs to a precious object. Now if the tag exists, it will reply and an adversary will become sure that a valuable object exists around him. Such security threats are very common while searching. So introducing prevailing, secure and practical RFID searching is one of the major goals of researchers now a day.

So far serverless searching is discussed only in [5]. In serverless system, reader has to search, authenticate as well as provide security without server's intervention. This departure from server based system will also reduce cost for RFID system deployment in many areas where tag searching is done frequently like supply chain management and E-passport.

In this paper, we tried to find solutions to the following questions: a) how readers can search a particular tag without the help of server? b) how a tag identifies that the communicating reader is legitimate? Here, we propose a low cost, secured, serverless search protocol that provides solutions to the preceding questions. And all these characteristics are ensured without a back end server which makes our proposal suitable for various application areas.

1.1 Our major contribution

I. In this paper we are proposing serverless, forward secure, anonymous searching protocols for RFID tags.

II. We have considered all the major attacks and our search protocols are secure against those attacks. We considered security of both tags and readers as both can be attacked by adversaries

III. In this paper, we discussed some real life application challenges and their solutions using our proposed serverless search protocols.

The remainder of the paper is organized as follows. The next section presents related work. Some major security requirements for RFID search protocols are reflected in section 3. Section 4.1 provides some

preliminaries for the rest of the paper. Section 4.2 provides search protocols and their security analysis is discussed in section 4.3. Some real life challenges and solutions are discussed in section 5. And finally in section 6 some concluding remarks are reported.

2. Related works

The assortment of research literature on RFID searching is inadequate although it is a major issue in its real life implementation. We stated in section 1 that RFID searching is an extension of RFID authentication. So we will go through some relevant literatures on RFID authentication. But we will mainly concentrate on the single serverless searching protocols proposed so far [5].

RFID systems are severely vulnerable to many security and privacy threats. That is why numbers of techniques have been proposed for ensuring RFID security and the assortment of authentication protocols is quite extensive [3]. Most of the authentication protocols proposed so far is backed by central database. One such famous authentication protocol is YA-TRAP [6] which is not secured against DOS attack. Another hash chain based RFID identification protocol is RIPP-FS [2], which shares a private symmetric key with server. Another famous lightweight authentication protocol is OSK [4], which suffers from the problem of desynchronization. In [1], Avoine and Oechslin modified OSK which removed the scalability problem. Serverless authentication protocols are proposed for the first time in [5]. In this paper, Chiu et al. proposed a challenge response based mutual authentication protocol. But the reader has to do lot of computation to find out id of the required tag. And their protocol 2 is not purely and strongly anonymous.

Serverless RFID searching protocols were also proposed in [5] for the first time. According to this protocol, a reader wishes to find out whether a specific tag is within its vicinity by broadcasting $h(f(r_i, t_j)||n_r) \oplus id_j, n_r$ and r_i . Based on this search query, only the intended tag, if exists, reply with its encrypted id. Other tags within the reader's vicinity reply a random number based on certain probability. Tags authenticate the reader based on the search query and reader authenticates tags based on the reply "string". Both valid query and valid replies are generated by legitimate parties.

3. Security requirements

A number of research literatures has dealt with several privacy and security issues of RFID. Some of which are discussed in section 2. RFID searching should also be secured because an adversary may want to find out whether precious objects exist by querying tags. So we point out that the following security goals must be guaranteed by our protocols: protection against tracking, eavesdropping and cloning.

4. Search protocols

In practical implementation of RFID, a reader often wants to find out whether a particular tag exists around him within a group of tags. One solution can be to perform authentication protocol for each of the tags of that group. But this is an inefficient approach as the number of tags within a system is likely to be huge. So another solution for RFID searching can be: reader will search for a tag and only that particular tag, if it exists, will reply in return. So the objective of secure RFID searching should be: the reader will search a specific RFID tag which he is authorized to access. And tags will reply with valid answers only if the reader is legitimate.

In this paper, we present different search protocols. According to the protocols, tag identifier is not passed to the reader in response to a reader's query. Whereas the tag sends certifying information to the reader in such a way that only the authorized reader is able to find out whether this is the desired tag. In this way, the reader can become sure about the existence of the tag that he is searching for.

4.1. Notation and assumption

We refer an RFID reader denoted as R. Each R has a unique identifier r and a contact list $\mathcal{L}$. We will describe the contents of $\mathcal{L}$ a little later. R obtains r and $\mathcal{L}$ from a trusted center, TC, after authenticating itself. The TC is a trusted party who deploys all the RFID tags and authorizes any RFID reader. For the sake of simplicity we assume that R and TC communicate through a secure channel.

According to our proposal, Each RFID tag T contains a unique value id, a unique secret t in its nonvolatile memory. All readers and tags also have knowledge of a pseudorandom number generator $\mathcal{P}(.)$ which takes a $seed$ as an argument and outputs a pseudorandom number according to its distribution. After generating a pseudorandom number, $\mathcal{P}(.)$ makes use of a function $\mathcal{M}(.)$ that generates next $seed$ of the pseudorandom number generator. For each authorized tag, the current $seed$ is stored in the reader in its nonvolatile memory. And in case of tag, a current $seed$ is stored for the authenticated reader in its nonvolatile memory. The initial $seed$ is computed by TC and stored in the tag and the reader by TC. The $seed$ stored

in both the reader and tag, is defined in the following manner:

$$seed_i^0 = f(r, t) = h(r \parallel t)$$
$$seed_i^{k+1} = \mathcal{M}(seed_i^k)$$

where, $h(.)$ is a one way hash function and i represents i^{th} tag or reader. Superscript k is used to represent the $seed$ after generating $(k-1)^{st}$ pseudorandom number from the distribution of pseudorandom number generator. $seed_i^k$ is used to generate k^{th} number according to its distribution. From now on, we will refer to k as the step k. In fact, both the $seed$s in tag and reader become same after each authentication and searching.

Subscripts are used to describe a particular R or T and their respective variables. Thus a particular RFID reader i will be R_i, with an identifier r_i and contact list $\mathcal{L}_i$. A tag j is T_j and has a secret t_j. The contact list $\mathcal{L}$ contains information about the RFID tags which a particular R has access to. $\mathcal{L}$ has a list of all $seed^0 = f(r,t)$ that TC has authorized R to access. So reader i, R_i authorized to access tags $T_1, \cdots, T_n$ will have $\mathcal{L}_i$ after authenticating itself to TC where,

$$\mathcal{L}_i = \begin{Bmatrix} seed_1^0 : id_1 \\ seed_2^0 : id_2 \\ . \\ . \\ seed_n^0 : id_n \end{Bmatrix}$$

Note that R_i does not know any of the tags secret t. It only knows the outcome of the function $f(r,t)$ as $seed^0$. We assume that the TC cannot be compromised, and that all readers once authenticated by the TC are trusted. We denote an adversary as ϱ .

$seed_i^k$ and $seed_j^k$ represents current $seed$ of pseudo random number distribution of R_i and T_j respectively, where superscript k bears aforementioned meaning. n_j^k is a pseudorandom number generated by i^{th} reader for the j^{th} tag using $seed_j^k$ at step k. Similarly, n_i^k is another pseudorandom number generated by j^{th} tag for the i^{th} reader using $seed_i^k$ at step k.

We can assume $\mathcal{M}(.)$ as an irreversible one way hash function. Therefore a $seed$ can't be linked to the previous $seed$. Although we haven't explicitly shown the use of $\mathcal{M}(.)$ by $\mathcal{P}(.)$ in the protocols, after generating a new pseudorandom number $\mathcal{P}(.)$ executes $\mathcal{M}(.)$ to update the $seed$ which will be stored in a nonvolatile memory of reader or tag. For example, T_j generates a pseudorandom number n_i^k for R_i using $seed_i^k$ stored in $T_j$and at the same time next seed $seed_i^{k+1}$ is also generated. Therefore $\mathcal{P}(.)$ performs like:

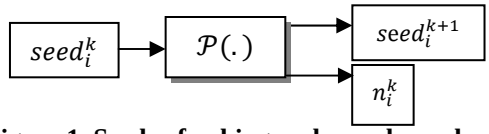

Figure 1. Seed refreshing and pseudorandom number generation mechanism.

Correspondingly, in case of a reader R_i, $\mathcal{P}(.)$ performs in the same way by replacing i with j. Whenever the $seed$ needs to be stored in nonvolatile memory it is explicitly mentioned in the protocols. For example, $seed_j = seed_j^k$ represents that $seed_j^k$ is stored in $seed_j$ along with id_j in contact list, $\mathcal{L}$ of reader.

4.2. Protocols

Suppose, a reader R_i is searching for a tag which we are referring to as $T_{desired}$. One way of searching may be according to Search Protocol 1. Here R_i broadcasts its r_i wishing to find $T_{desired}$. Before getting reply from the tags, R_i computes next random number $(n_{desired}^k)$ for the desired tag using $seed_{desired}^k$. Now, all the tags receiving r_i will reply with next random number (n_i^k) for this particular reader. Reader compares computed random number with those received from the tags. If a match occurs, reader becomes sure that the $T_{desired}$ is present.

Search Protocol 1

$R_i \rightarrow T* : Broadcast\ r_i$
$R_i \quad : Compute\ n_{desired}^k = \mathcal{P}(seed_{desired}^k)$
$T* \quad : n_i^k = \mathcal{P}(seed_i^k)$
$R_i \leftarrow T* : n_i^k$
$R_i \quad : If\ (n_i^k = n_{desired}^k)\ then$
$T_{desired}\ found$
$Else$
$T_{desired}\ not\ found$

One main problem of this protocol is that it is a one side authenticated search protocol. Here tags do not authenticate the readers before replying. So they cannot know whether they are replying to an adversary or to a valid reader. Tags should only reply to authorize the reader. But here tags reply whenever they see a query. Sometimes even an adversary may query a group of tags to find out if a particular valuable tag is present. So the tag needs to authenticate the reader before replying. It means that when R_i broadcasts the search query, every tag, not only the tag that satisfies this query, needs to authenticate R_i before replying.

Another issue is, as seeds are not updated in both parties after each search, tags will reply to the same

reader with the same answers in subsequent queries. If an adversary queries with a previously listened r_i, tags will reply with the exact same values as before. Although the adversary will not be able to find out which tag the reader was searching for, it will become sure that the same search is taking place. Querying several times with different r_i, adversary can get a pattern for queries and replies.

The problem of replying with the fixed answer for the same reader can be solved if we update the seed in both parties after each search, which is specified in search protocol 2.

Search Protocol 2

$R_i \rightarrow T*$: $Broadcast\ r_i$
R_i : $Compute\ n_{desired}^k = \mathcal{P}(seed_{desired}^k)$
$T*$: $n_i^k = \mathcal{P}(seed_i^k)$
$seed_i = seed_i^{k+1}$
$R_i \leftarrow T*$: n_i^k
R_i : $seed_j = seed_j^{k+1}\ for\ each\ tag$
$T_j\ replying\ with\ n_i^k\ , where$
$1 \leq j \leq n$
$If\ (n_i^k = n_{desired}^k)\ then$
$T_{desired}\ found$
$Else$
$T_{desired}\ not\ found$

In this protocol, after replying to the search query each tag will update its seed. A reader will update the seeds of only those tags, which have replied. But here the problem is that reader has to update $O(n)$ seeds in worst case scenario. Therefore, the reader is burdened with more computations.

Another problem of this protocol is synchronization. By querying tags, an adversary can desynchronize the tags and reader very easily. As a result after de-synchronization, in spite of the presence of the desired tag, a legitimate reader cannot access it.

Therefore, we can set up our goals for searching as follows. Tags should only respond to authenticated readers. The reader should only query authenticated tags. And both parties should update their seeds after authentication. All these properties are incorporated in our final search protocol which is search protocol 3. Her, the reader issues a query in a way that only an authenticated tag can understand and the tag replies in such a manner that only an authenticated reader can understand.

Search Protocol 3

R_i : $Compute\ n_{desired}^k = \mathcal{P}(seed_{desired}^k)$
$R_i \rightarrow T*$: $Broadcast\ n_{desired}^k$
$T*$: $n_i^k = \mathcal{P}(seed_i^k)$
$If\ (n_i^k = n_{desired}^k)\ then$
$n_i^{k+1} = \mathcal{P}(seed_i^{k+1})$
$seed_i = seed_i^{k+2}$
$R_i \leftarrow T_j : n_i^{k+1}$
$Else$
$R_i \leftarrow T_j$: $rand\ with\ probablity\ \lambda$
R_i : $n_{desired}^{k+1} = \mathcal{P}(seed_{desired}^{k+1})$
$If\ (n_i^{k+1} = n_{desired}^{k+1})\ then$
$seed_{desired} = seed_{desired}^{k+2}$
$T_{desired}\ found$
$Else$
$T_{desired}\ not\ found$

In this protocol, R_i computes $n_{desired}^k$ and broadcasts it to find out $T_{desired}$. All tags receiving $n_{desired}^k$ will compare this with their own individual n_i^k. If a match occurs, the tag will know that it is an authorized reader. A match can occur only in $T_{desired}$ because only a legitimate reader can know its seed. Therefore only a valid reader can generate valid $n_{desired}^k$. Hence after authenticating the reader in this way, $T_{desired}$ will reply with next number (n_i^{k+1}) for this reader and update its seed. And for those tags in which a match doesn't occur, they will reply with a random number with probability λ. Reader now computes $n_{desired}^{k+1}$ and compares it with n_i^{k+1}. If a match occurs, then reader can be sure that it is a valid tag as only a legitimate tag can generate this. Therefore, the reader now updates its seed for $T_{desired}$. This protocol is resistant against almost all the attacks. Security analysis for this protocol is discussed in the next subsection.

In search protocol 3 we let some other tags also reply in addition to the desired tag to put the actual reply in disguise. Each tag receiving a search query that does not match with the request will have some probability λ of replying. So by observing tag replies, an adversary cannot reveal a particular tag that the reader is searching for.

4.3. Security analysis of search protocols

***Tracking*:** Our final protocol is resistant against tracking. Tracking attack in searching is slightly different from the one found in security literature. Here adversary cannot pick a particular tag to track. Rather, he can only track a tag that has been searched for by a legitimate reader. Consider the following attack. ϱ eavesdrops on the transaction between a reader R_i and tags. So he knows the queries and replies. He will not be able to reverse compute the replies or learn the query but he can certainly be sure that a searching has taken place. However he cannot be sure, which tag $T_{desired}$ reader was searching for, as besides the desired tag other tags also replied with probability λ.

Now ϱ can replay previously listened $n_{desired}^{k}$ to track $T_{desired}$. But after the previous successful searching between R_i and $T_{desired}$, both parties have changed their seeds. So $n_{desired}^{k}$, send by the adversary, will not match with the one computed by $T_{desired}$. As a result $T_{desired}$ will reply will a random number. At the same time other tags will also reply a random number. If ϱ continues to query with different $n_{desired}^{k}$, all tags including the desired tag will reply randomly. Therefore ϱ will not be able to track a tag.

***Cloning*:** Consider the following cloning attack. R_i queries to search a tag $T_{desired}$. If $T_{desired}$ is present it will reply. At the same time other tags will also reply. Suppose, ϱ finds out the tag the reader was searching for. Now if he is able to clone $T_{desired}$, then he can fool R_i by not replying or even giving a false reply. As a result, R_i will assume that the desired tag $T_{desired}$ does not exist in this group. In our protocol, this attack is impossible. Because ϱ is unable to find out, which tag the reader was searching for.

***Eavesdropping*:** Here ϱ observes all the queries between a reader and tags. And his goal is to use the data to impersonate a fake reader R_i or a fake tag T_j. Our protocol is powerful against this attack. In our protocol ϱ will not be able to find out the expected reply of the reader as more than one tag will reply. He can only observe $n_{desired}^{k}$ send by the reader. With his little knowledge he cannot impersonate R_i or T_j, because after the last successful searching between R_i and $T_{desired}$, both of them have updated their seeds. So both of them are now expecting new values which are not known by ϱ. Therefore by eavesdropping ϱ cannot launch a replay attack by using previous values.

5. Illustrative examples

In this section we have drawn a couple of application scenarios that can be directly benefited from our approach presented in this paper.

User Interactions in a smart space: A smart space typically contains multiple smart objects offering several invisible services. Users' personal devices are usually used to interact with the smart space. Discovering invisible services securely and authenticating the users are interesting research problems in the smart space domain. Our approach offers promising solutions to both of these problems. Iconic images embedded with RFID tags can advertise invisible services and user terminals can be equipped with an RFID reader. A user can search for a specific service (tags in this case) or can initiate a service by touching the tag. Considering the pre-negotiation between the reader and the tags, secure discovery and authentication mechanism can be easily achieved applying our protocol.

Container search within seaports: There are hundreds and thousands of containers within a seaport. Containers are parked and stacked by hundreds of employees and countless drivers who deliver containers from remote locations. Moreover, containers are also unloaded from ships in order to deliver them to different customers and locations. Whether a particular container has already been unloaded from the ship or not, whether a specific container has arrived at the seaport for shipment or not, are some of the major tasks performed within seaports. But it is quite impossible to search for a particular container manually. That is why seaports in different countries have long been searching for technologies that can identify specific containers and that can confirm the existence of containers within seaports. One solution to the aforementioned problem can be to use RFID tags for container identification. Now through the use of our serverless search protocols, it will be quite easy to search for a particular container by searching the tag. If a container's tag id (in fact $seed$) is known, then we can invoke a search operation with this id within the seaport. If the container is present within the seaport then according to our protocol, definitely that particular tag will reply. Thus we can be sure about the container's existence.

6. Conclusions

RFID systems have been developing continuously in selected areas for decades now. It is still a potential technology which can be applied in practically all areas of daily life. Theoretically the application areas of RFID systems are unlimited. In spite of this, secure RFID searching has not gathered much attention till now. But we firmly believe that it will become very important when RFID will be deployed at a larger scale. In this paper we introduce various problems incurred while performing secure RFID tag search. Moreover, we analyzed different attack models of which tag searching is severely vulnerable. And finally we proposed secure serverless RFID tag searching protocols that can safeguard against those major attacks without server's intervention. We also discussed a couple of applications of our proposed serverless protocol in a real life scenario. We are currently working on realizing these scenarios through actual implementations. The application of our protocol is not limited to these examples only, but it can also be applied to some other real life circumstances.

7. References

[1] G. Avoine, and P. Oechslin, “A Scalable and Provably Secure Hash Based RFID Protocol”, In *International Workshop on Pervasive Computing and Communication Security (PerSec ’05)*, IEEE, IEEE Computer Society Press, Kauai Island, Hawaii, USA, March 2005, pp. 110–114.

[2] M.Conti, R. D. Pietro, L. V. Mancini, and A. Spognardi, “RIPP-FS: an RFID Identification, Privacy Preserving Protocol with Forward Secrecy”, In *International Workshop on Pervasive Computing and Communication Security (PerSec ‘07)*, IEEE, IEEE Computer Society Press, New York, USA, March 2007, pp. 229-234.

[3] A. Juels, “RFID Security and Privacy: A Research Survey”, RSA *Laboratories*, September 2005.

[4] M.Ohkubo,K. Suzuki, and S. Kinoshita, “Cryptographic Approach to “Privacy-Friendly” Tags”, In *RFID Privacy Workshop*, MIT, MA, USA, November 2003.

[5] C. C.Tan, B. Sheng, and Q. Li, “Severless Search and Authentication Protocols for RFID”, In *Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom ‘07)*, New York, USA, March 2007.

[6] G. Tsudik, “YA-TRAP: Yet another Trivial RFID Authentication Protocol”, In *International Conference on Pervasive Computing and Communications (PerCom ’06)*, IEEE, IEEE Computer Society Press, Pisa, Italy, March 2006.