A New Object Searching Protocol for Multi-tag RFID

Subhasish Dhal* and Indranil Sengupta

Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, INDIA

Email: Subhasish Dhal*- sdhal@cse.iitkgp.ernet.in; Indranil Sengupta - isg@iitkgp.ac.in;

*Corresponding author

Abstract

Searching an object from a large set is a tedious task. **R**adio **F**requency **ID**entification (RFID) technology helps us to search the desired object efficiently. In this technology, a small chip called RFID tag, that contains the identification information about an object is attached to the same object. In general, a set of objects are attached with RFID tags. To find out a particular object preserving the possible security requirements, the RFID reader requests the tag in desired object to respond with its encrypted identification information. Since there is a response only from the tag in desired object the adversary gets the knowledge of existence of the desired object. Fake response from tag in undesired objects may fool the adversary. However, computation for fake responses is an overhead. In this paper, we propose a search technique which has a negligible amount of computation for fake responses. Multiple tags in the same object increases the detection probability and also the probability of success in search process. Our aim is to search a particular object efficiently preserving the possible security requirements amid various resource limitations in low-cost RFID tag.

keywords: multi-tag , RFID , authentication , resiliency

1 Introduction

Searching an object of interest from a large number of objects is not an easy task. This is because the underlying information about the object may not be distinguishable manually, and hence may not be easily visible. We may keep the underlying information of all objects in a database and provide a unique ID to each object. Thus, we can distinguish an object using its ID. In RFID technology [1], the underlying information about an object is kept in a workstation called backend server, and a unique ID which is used to relate the underlying information about the object is kept in a RFID tag attached to the same object. To search an object, the RFID reader requests the tag attached with the desired object to respond with its ID and thus it finds out the desired object.

There are many security and privacy risks involved in searching process. In some systems, the process of searching an object may reveal valuable information to the adversary. Using this information, the adversary may be able to achieve her objectives. Furthermore, RFID is a pervasive computing technology which is easily susceptible to various kind of attacks. Therefore any communication in this technology needs to be secure. On the other hand, RFID tag being a low cost device suffers from various resource limitations. Therefore, traditional cryptography is not applicable in this technology. The challenge is to devise a search technique which not only satisfies most of the security requirements but is also applicable to resource sensitive environment. Any light-weight authentication scheme [2] [3] [4] [5] [6] can be used to search an object. However, we have to authenticate each tag until we get the desired tag, which requires to authenticate $\frac{n}{2}$ number of tags on the average for each search, where n is the total number of tags in the environment.

The search process needs to be such that the adversary should not even know the existence of the desired object. The fake responses from undesired objects can fool the adversary. However, the search process requires useless computations due to fake responses. We propose a solution which requires a negligible amount of computations for the fake responses.

During search process, if the location within the desired object where the tag is attached with is not within the communication range of reader, the object cannot be detected. However, a few locations of the same object may be within the communication range. Therefore lying within the communication range, the object is undetected. Attachment of multiple number of tags in the same object with proper alignment [7] solves this problem.

Recent works [8] [9] [10] [11] [12] have tried to solve the object searching problem with the assumption that an object is attached with single tag and they keep only one set of security related information for an object. However, if an object is attached with multiple number of tags, then there will be more resources and it is possible to keep more than one set of security related information. This can make the search process more powerful with respect to security and privacy. This is because the adversary now needs to compromise multiple number of tags for compromising an object. Furthermore, if the search process mandates all the tags in an object to respond for its detection, the achievable security and privacy benefit will be maximum. However, this requirement may decrease the detection probability since all the tags need to be within the communication range of reader. A threshold scheme can mitigate the said issue, which uses the fact that an object is detectable only when a minimum number of tags attached to it are within the communication range of the reader. It is quite obvious that if the threshold value is less, the detection rate is more but security is less. On the other hand, if the threshold value is more, the detection rate is less but security is more. Therefore, the threshold value needs to be chosen according to the requirement. We have tried to find out the solutions to the following questions:

a) How a reader can search at least a threshold number of tags from a large number of tags to identify an object?

b) What are the extra benefits corresponding to security and privacy requirements that we can obtain?

c) How can we decrease the computation overhead for the fake responses?

d) What is the performance of the threshold scheme?

In this paper, we have proposed a search technique assuming the existence of multiple number of tags in the same object. Our scheme takes advantage of multiple resources in the same object and hence improves the security and privacy benefits. This is also light-weight and thus practically applicable to object searching problem.

The remainder of the paper is organized as follows. In section 2, we have briefly discussed the related schemes which have been proposed recently. In section 3, we have introduced the communication model and possible threats in it. We have described our proposed object searching scheme in section 4. In section 5, we have analyzed the proposed scheme an compared our scheme with the existing schemes followed by conclusion in section 6.

2 Related works

Object search is a very important problem since the desired object needs to be searched efficiently preserving the possible security and resource requirements. However, the literature has not focused this problem adequately. A few tag searching schemes have been suggested in the literature. We briefly revisit and analyze those schemes in this section.

Tan et al. [8] proposed four tag searching schemes. In their basic scheme (depicted in Figure 1), the reader broadcasts the desired tag information. The tag will check the validity and then respond. Although the request information is encrypted, the adversary may use this information to track the tag. They provide

Figure 1: Proposed protocol by Tan et al. [8] scheme 1

$$
\begin{array}{rcll}
R_i \longrightarrow T* & : & h(f(r_i, t_j) \| n_r) \oplus id_j, n_r, r_i & 1 \\
T* & : & \text{Derive } h(f(r_i, t) \| n_r) \text{ and XOR with} & \\
 & & h(f(r_i, t_j) \| n_r) \oplus id_j & 2 \\
 & : & \text{if } id = id_j \text{ and } n_r = oldn, & \\
 & & \text{Update } oldn = n_r & 3 \\
R_i \longrightarrow T_j & : & h(f(r_i, t_j) \| n_t) \oplus id_j, n_t & 4
\end{array}
$$

a solution (sketched in Figure 2) by using a list of random numbers in the tag. In each session, the reader will use a new random number. Hence, the tag will check the random number sent by reader in its list and if it finds any match, it will ignore the request. Otherwise, the tag will accept the request and add the

Figure 2: Proposed protocol by Tan et al. [8] scheme 2

$$
\begin{array}{rcll}
R_i \rightarrow T* & : & h(f(r_i, t_j) || n_r) \oplus id_j, n_r, r_i & (1) \\
T* & : & \text{Deriving } h(f(r_i, t) || n_r) \text{ and XOR with} & \\
 & & h(f(r_i, t_j) || n_r) \oplus id_j & (2) \\
 & : & \text{If } id = id_j \text{ and } n_r \neq oldn, & \\
 & & \text{update } oldn = n_r & (3) \\
R_i \leftarrow T_j & : & h(f(r_i, t_j) || n_t) \oplus id_j, n_t & (4)
\end{array}
$$

new random number to its list. This solution is not scalable since the tag has to manage the list of random numbers which will increase in each session. The other problem is that the adversary may not have any knowledge about the object, however, she gets the idea about the existence of it. This scheme also suffers from tracking attack [13]. In another improvement (see Figure 3), they allow all tags in the vicinity to reply

Figure 3: Proposed protocol by Tan et al. [8] scheme 3

$$
\begin{array}{rcll}
R_i \rightarrow T* & : & \text{Broadcast } [id_j]_m, r_i, n_r & (1) \\
T* & : & \text{If } id_m = [id_j]_m & (2) \\
R_i \leftarrow T_j & : & h(f(r_i, t_j) || n_r || n_t) \oplus id_j, n_t & (3) \\
R_i & : & \text{Determines } f(r_i, t_j) \text{ from } L, \text{ obtain } id_j & (4)
\end{array}
$$

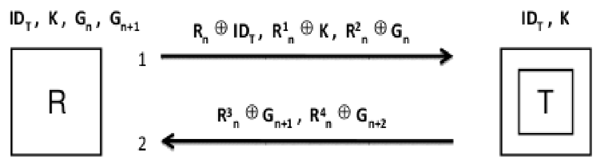
revealing a few bits of tag identifier which has a proper structure. The reader will distinguish the desired tag using the revealed bits. However, a few bits of tag identifier is revealed to the adversary providing some knowledge which allows the adversary to track the tag [13]. Finally, they have another improvement (illustrated in Figure 4) where all tags other than the desired one will reply with certain probability. In this improvement, they have used two hash functions in both reader side and tag side which is less efficient and suffers from ID disclosure attack [13].

Figure 4: Proposed protocol by Tan et al. [8] scheme 4

$$\begin{array}{rcll}
R_i \rightarrow T* & : & \text{Broadcast } h(f(r_i, t_j)||n_r) \oplus id_j, n_r, r_i & (1) \\
T* & : & \text{Derive } h(f(r_i, t)||n_r) \text{ and XOR with} & \\
 & & h(f(r_i, t_j)||n_r) \oplus id_j & (2) \\
 & : & \text{If } id = id_j : & \\
 & & \quad R_i \leftarrow T_j : h(f(r_i, t_j)||n_t) \oplus id_j, n_t & (3) \\
 & : & \text{Else :} & \\
 & & \quad R_i \leftarrow T_j : (rand, n_t) \text{ with prob. } \lambda & (4)
\end{array}$$

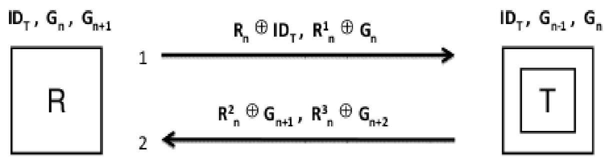
Kulseng et al. [9] proposed three algorithms based on Linear Feedback Shift Register (LFSR) and Physically Unclonable Function (PUF). In first algorithm (depicted in Figure 5), the reader has information $\mathtt{ID}_T$, pairwise secret $\mathtt{K}$ and two greetings g_n, g_{n+1} for tag T. On the other hand, tag T has the same identifier

Figure 5: Proposed protocol by Kulseng et al. [9] scheme 1


ID_T, K, G_n, G_{n+1}
$R_n \oplus ID_T, R^1_n \oplus K, R^2_n \oplus G_n$
ID_T, K
1
R
T
$R^3_n \oplus G_{n+1}, R^4_n \oplus G_{n+2}$
2

$\mathtt{ID}_T$ and pairwise secret key $\mathtt{K}$ in its memory. Reader will broadcast a search query with encrypted tag information. The desired tag will respond with authentication information. In this scheme, the only desired tag will reply and adversary will be able to know about its existence. In the second algorithm (illustrated in Figure 6), they have improved the first by keeping old information which removes the synchronization problem. This solution also suffers from same problem. In third solution, they have removed this problem

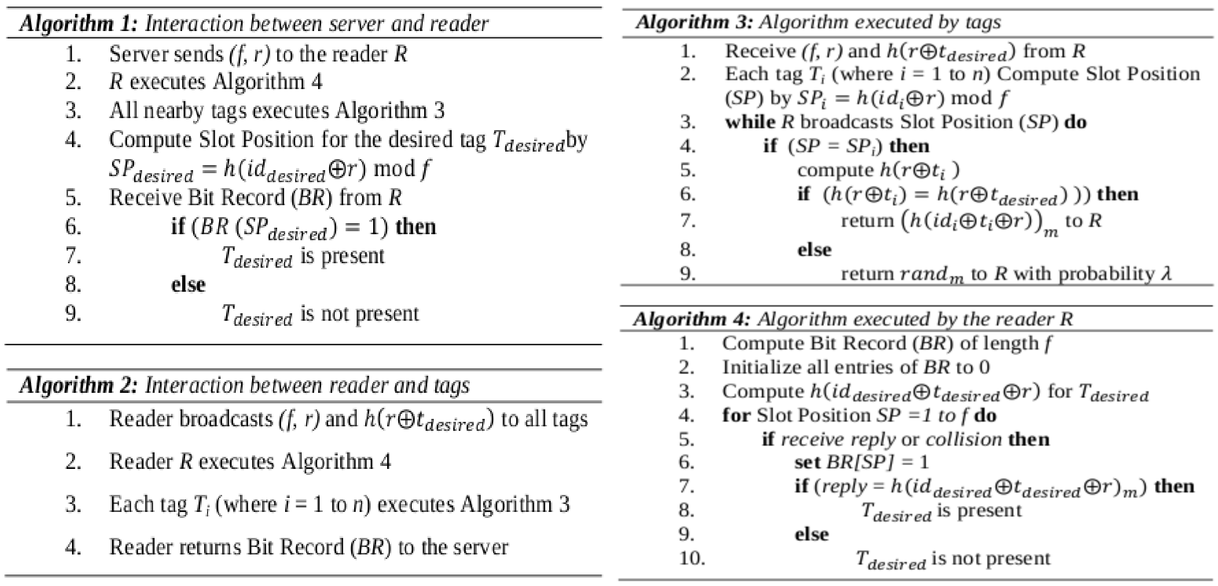
Figure 6: Proposed protocol by Kulseng et al. [9] scheme 2


ID_T, G_n, G_{n+1}
$R_n \oplus ID_T, R^1_n \oplus G_n$
ID_T, G_{n-1}, G_n
1
R
T
$R^2_n \oplus G_{n+1}, R^3_n \oplus G_{n+2}$
2

by allowing all tags in the vicinity to respond where the undesired tags will respond with certain probability using fake information. In this scheme, the reader needs to process all responses one by one in the same way the reader needs to process the valid response.

Hoque et al. [10] proposed the S-search protocol (see Figure 7) for finding a tag. In their scheme, the server will send a frame length and a random number to the reader. The reader will generate encrypted tag information and broadcast it along with slots for the tags. Each tag in the vicinity of reader will compute the slot for itself and then check the tag information. The desired tag will generate authentication information

Figure 7: Protocol proposed by Hoque et al. [10]



```
Algorithm 1: Interaction between server and reader
1. Server sends (f, r) to the reader R
2. R executes Algorithm 4
3. All nearby tags executes Algorithm 3
4. Compute Slot Position for the desired tag $T_{desired}$by
   $SP_{desired} = h(id_{desired} \oplus r)$ mod f
5. Receive Bit Record (BR) from R
6.     if (BR ($SP_{desired}$) = 1) then
7.         $T_{desired}$ is present
8.     else
9.         $T_{desired}$ is not present
```

```
Algorithm 2: Interaction between reader and tags
1. Reader broadcasts (f, r) and $h(r \oplus t_{desired})$ to all tags
2. Reader R executes Algorithm 4
3. Each tag $T_i$ (where i = 1 to n) executes Algorithm 3
4. Reader returns Bit Record (BR) to the server
```

```
Algorithm 3: Algorithm executed by tags
1. Receive (f, r) and $h(r \oplus t_{desired})$ from R
2. Each tag $T_i$ (where i = 1 to n) Compute Slot Position
   (SP) by $SP_i = h(id_i \oplus r)$ mod f
3. while R broadcasts Slot Position (SP) do
4.     if (SP = $SP_i$) then
5.         compute $h(r \oplus t_i)$
6.         if ($h(r \oplus t_i) = h(r \oplus t_{desired})$ )) then
7.             return $\left(h(id_i \oplus t_i \oplus r)\right)_m$ to R
8.         else
9.             return $rand_m$ to R with probability $\lambda$
```

```
Algorithm 4: Algorithm executed by the reader R
1. Compute Bit Record (BR) of length f
2. Initialize all entries of BR to 0
3. Compute $h(id_{desired} \oplus t_{desired} \oplus r)$ for $T_{desired}$
4. for Slot Position SP =1 to f do
5.     if receive reply or collision then
6.         set BR[SP] = 1
7.         if (reply = $h(id_{desired} \oplus t_{desired} \oplus r)_m$) then
8.             $T_{desired}$ is present
9.         else
10.            $T_{desired}$ is not present
```

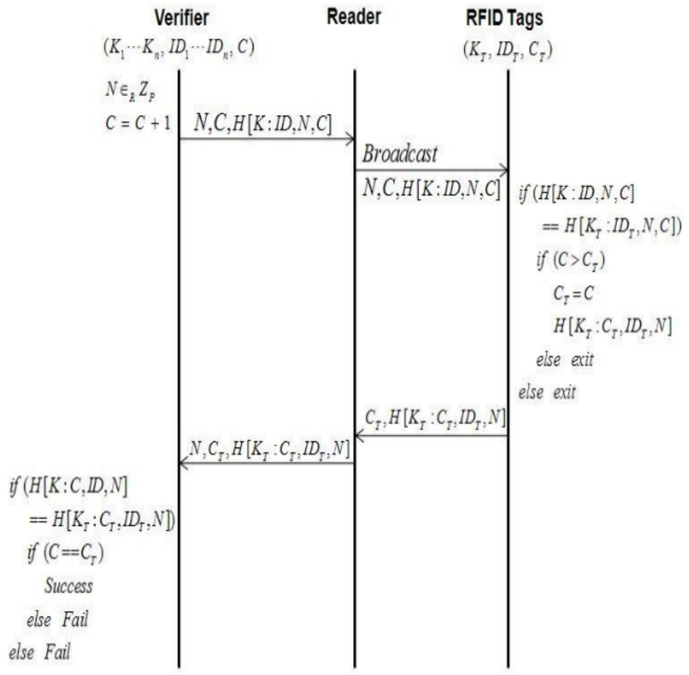
and respond along with the slot to reader. The other tags will respond with a certain probability and reply with a random number along with the slot assigned to them. The reader will generate a bit record (BR) and verify the presence of desired tag and then send this to server. The server will investigate the bit record and confirm about the existence of desired tag. In their scheme, the computation is more due to hash functions in both reader and tag which makes the scheme less efficient.

Yoon et al. [11] proposed a tag searching scheme (illustrated in Figure 8) where a trusted verifier and tag will maintain a counter. The verifier will send encrypted tag information to reader along with the counter value. The reader will broadcast this information.

Desired tag will check the tag information and then check the counter value. If all are verified, it will update its counter value and generate the authentication information. Then the desired tag will respond with authentication information to verifier through reader. The verifier will validate this information and report the search result. In this scheme, the adversary has the knowledge about the existence of desired tag. Another problem is that they have used two hash operations in tag which will consume more power and make the protocol less efficient.

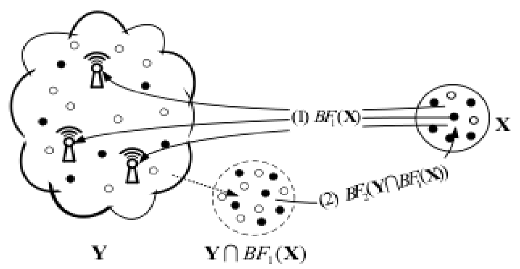
Zheng et al. [12] proposed a two-phase compact approximator based tag searching protocol (see Figure 9) using bloom filter. In their scheme, multiple number of tags can be searched in same query. In the first phase of their scheme, the reader will broadcast the membership information of desired tags using a bloom

Figure 8: Protocol proposed by Yoon et al. [11]



filter. A tag will check the membership and get silent if it is not a member. However, due to false positive property of bloom filter, a few undesired tags will be selected along with desired tags. Thus, a number of

Figure 9: Protocol proposed by Zheng et al. [12]



(1) $BF_1(\mathbf{X})$
$\mathbf{X}$
(2) $BF_2(\mathbf{Y} \cap BF_1(\mathbf{X}))$
$\mathbf{Y}$
$\mathbf{Y} \cap BF_1(\mathbf{X})$

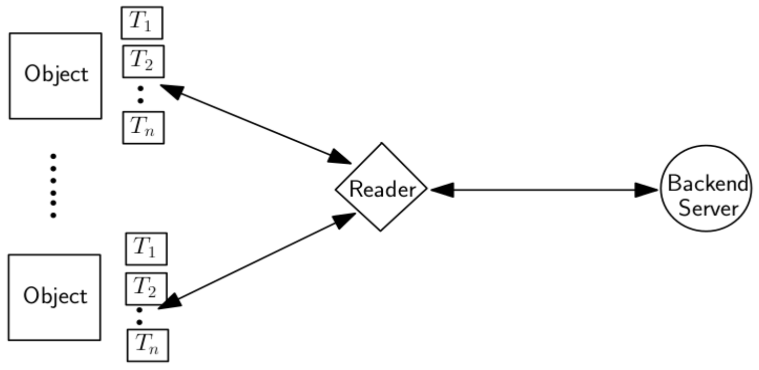
tags including the desired tags will be ready to cooperate further. In second phase, the selected tags in first phase will respond on a query from reader. The reader will forward these responses to server. Then the server will construct a virtual bloom filter and filter the desired tags. In this scheme, there is a probability that a number of undesired tags could be selected even after second filtration due to false positive property of bloom filter.

The schemes in [8] [9] [10] [11] [12] assume that an object is attached with single tag. These are also suffer from one or more flaws. Thus, the compromise of single tag will enable the adversary to search the object attached by that tag. We have proposed an object searching scheme with the assumption that an object is attached with multiple number of tags and to get valid response from the desired object, a minimum number of tags attached to it need to be visible to the reader. In this scheme, the compromise of single tag is not sufficient for the adversary. She needs to compromise a minimum number of tags equal to a threshold value and no less to get information about the object and search further.

3 Communication model and possible threats

In this section, we briefly introduce the communication model and the possible threats in it.

Figure 10: Components of communication model



3.1 Communication model

In our proposed object searching scheme, we have three components which can communicate with each other, namely, *object, reader* and *backend server* as shown in Figure 10.

1. **Object**: We assume that there are multiple number of objects and each object is attached with multiple number of RFID tags. There are four type of tags, namely, *active, semi-active, semi-passive* and *passive*. In this paper, we have assumed that the tags are of type *passive*. However, our scheme is applicable to other type of tags also.

2. **Backend server**: Backend server is a stationary component that checks the authentication of information about an object and updates various security parameters accordingly. It also maintains a database. Various information about each object and hence about each tag are kept in this database.

3. **Reader**: RFID reader is responsible for initiating a search with the help of backend server. It acts in between an object and backend server as shown in Figure 10. We assume that the communication between reader and backend server is secure while the communication between a reader and object is insecure.

3.2 Possible threats

During the search process, a number of complications may arise. An adversary may try to mount various kind of attacks since the communication between reader and object is insecure. We define each attack as follows:

i) **Eavesdropping:** The adversary listens to the communication between RFID reader and tag and be able to get valuable information.

ii) **Physical attack:** The adversary clones a legitimate tag and responds as a valid tag.

iii) **Traceability:** The adversary traces an object by observing the response pattern in each successful session.

iv) **Traceability within two consecutive successful sessions:** The adversary broadcasts the same query in between two consecutively successful sessions. The tag, on the other hand, replies with same information. Thus, the adversary is successful to trace the tag and hence object.

v) **Man in the middle attack:** The adversary disrupts the search process by modifying the information communicated between reader and tag.

vi) **Forward security:** The adversary guesses the variable secret like session key and be able to guess the future secrets utilizing the learnt secrets.

vii) **Backward security:** The adversary guesses the variable secret like session key and be able to guess the past secrets utilizing the learnt secrets.

viii) **Replay attack:** The adversary grasps the information exchanged during a valid search and utilizes the same to impersonate as a legitimate entity.

ix) **Information leakage:** The adversary gets the knowledge about the existence of an object by viewing the number of responses.

x) **Synchronization attack:** The adversary blocks the communication between reader and tag to desynchronize the information stored in backend server and tag.

4 Object searching protocol

We have proposed an object searching scheme preserving the possible security and privacy threats. In this section, we illustrate the problem and then we describe our proposed scheme. We have used the notations in Table 1 for explaining the scheme.

4.1 Problem definition

There is a large set of objects and each object is attached with multiple number of tags. A backend server keeps the information about all the tags. The owner of all the objects tries to find out a specific object from the set by initiating a search query in the server.

The goal of the server is to search a set of tags attached to the desired object efficiently preserving the necessary security and privacy requirements mentioned in section 3.2. All the tags attached to the object may not be within the communication range of RFID reader and hence the server requires to search at least a threshold number of tags attached to the object.

4.2 The protocol

We have proposed an object searching scheme based on the communication model explained in section 3.1. Before the explanation of our proposed scheme, we describe the data structures used in various components.

4.2.1 Database

Each tag contains a secret key, two session keys, and two identifiers. We keep two session key fields among which one is old and another is new. Similarly, there are two identifier fields. In Figure 11, there are five

Figure 11: Tag database

S_i	N_{inew}	N_{iold}	ID_{inew}	ID_{iold}

fields which are the contents in the memory of tag T_i attached in object G. The first field is the secret key, the second and third fields contain the new and old session keys and the fourth and fifth fields contain the new and old identifiers.

We keep information about each object in a database in backend server. There is a table which contains the records for each object. Therefore, for n objects there are n records in the table. Figure 12 shows various fields of a record for an object G. The first field contains the secret keys for the tags attached in object G.

Figure 12: Record in Backend server

$S_1, S_2, ..., S_n$	$N_1, N_2, ..., N_n$	$ID_1, ID_2, ..., ID_n$

The second and third fields contain the session keys and identifiers for each tag in object G.

4.2.2 Initialization

A few parameters are initialized and preloaded before the deployment of objects, reader and backend server. Following is the initialization steps in our proposed object searching scheme. For the sake of clarity, we mention the existence of only one object. However, the scheme is applicable to multiple objects.

a) The $i^{th}(i = 1, 2, ..., n)$ tag of object G is assigned a secret key S_i which is kept in secret key field in the memory of the same tag.

b) The $i^{th}(i = 1, 2, ..., n)$ tag of object G is assigned a session key N_i, which is kept in both N_{inew} and N_{iold} fields in the memory of the same tag.

c) An identifier ID_i is also assigned to $i^{th}(i = 1, 2, ..., n)$ tag of object G and this is kept in both ID_{inew} and ID_{iold} fields in the memory of same tag.

d) In backend server, the database is initialized with records of all objects. The information S_i, N_i, and ID_i for each tag attached in object G are kept under the record of G.

4.2.3 Steps involved in object searching:

The searching process in our scheme has a sequence of steps. We explain the required steps to search an object according to our scheme as follows:

Step 1: Communication from reader to backend server

- Reader requests backend server to send information for object G to be searched.

Step 2: Operation in backend server

- Generates a set A of n random numbers $r_{21}, r_{22}, ..., r_{2n}$.
- Assigns r_{2i} as index of tag $T_i (i = 1, 2, ..., n)$.
- for each tag $T_i (i = 1, 2, ..., n)$, generates a new session key N_{inew} and an identifier ID_{inew}.
- Generates update information $M_{1i} \leftarrow (S_i - N_i) \oplus (ID_{inew} - S_i)$ and $M_{2i} \leftarrow (S_i - ID_i) \oplus (N_{inew} - S_i)$.

Step 3: Communication from backend server to reader

- Backend server sends ID_i, N_i, M_{1i}, and M_{2i} for each tag $T_i (i = 1, 2, ..., n)$ along with A.

Step 4: Operation in reader

For each tag T_i $(i = 1, 2, ..., n)$ in object G

- Generates a random number r_{1i}.
- Calculates $K_i \leftarrow (N_i - r_{1i}) \oplus (ID_i - r_{1i})$, $V_i \leftarrow (r_{2i} - N_i) \oplus ID_i$.

Step 5: Communication from reader to object

- Broadcasts **request** message with $\eta_1\eta_2...\eta_n$. Here $\eta_i = r_{1i}, K_i, V_i, M_{1i}, M_{2i}, (i = 1, 2, ..., n)$.

Step 6: Operation in reachable tag T_i

- Retrieves $\eta_i (= r_{1i}, K_i, V_i, M_{1i}, M_{2i})$[1] intended to it from $\eta_1\eta_2...\eta_n$.
- Calculates $K'_i \leftarrow [(N_{inew} - r_{1i}) \oplus (ID_{inew} - r_{1i})]$
- **if** $K'_i = K_i$ **then**
  - Calculates $(r_{2i} - N_{inew}) \leftarrow V_i \oplus ID_{inew}$ and then $r_{2i} \leftarrow (r_{2i} - N_{inew}) + N_{inew}$
  - Calculates $(ID'_i - S_i) \leftarrow M_{1i} \oplus (S_i - N_{inew})$ and then $ID'_i \leftarrow (ID'_i - S_i) + S_i$
  - Calculates $(N'_i - S_i) \leftarrow M_{2i} \oplus (S_i - ID_{inew})$ and then $N'_i \leftarrow (N'_i - S_i) + S_i$
  - Calculates $P_i \leftarrow r_{2i} \| [(ID_{inew} - r_{2i}) \oplus (N_{inew} + r_{1i})]$.
  - Updates $N_{iold} \leftarrow N_{inew}, ID_{iold} \leftarrow ID_{inew}$ and then $N_{inew} \leftarrow N'_i, ID_{inew} \leftarrow ID'_i$.
- **else**
  - Calculates $K'_i \leftarrow (N_{iold} - r_{1i}) \oplus (ID_{iold} - r_{1i})$

[1]The i^{th} tag in an object receives only the i^{th} chunk of broadcast message and test for validity

```
    – if $K_i' = K_i$ then
        ∗ Calculates $(r_{2i} - N_{iold}) \leftarrow V_i \oplus ID_{iold}$ and then $r_{2i} \leftarrow (r_{2i} - N_{iold}) + N_{iold}$
        ∗ Calculates $(ID_i' - S_i) \leftarrow M_{1i} \oplus (S_i - N_{iold})$ and then $ID_i' \leftarrow (ID_i' - S_i) + S_i$
        ∗ Calculates $(N_i' - S_i) \leftarrow M_{2i} \oplus (S_i - ID_{iold})$ and then $N_i' \leftarrow (N_i' - S_i) + S_i$
        ∗ if $N_{inew} \neq N_i'$ and $ID_{inew} \neq ID_i'$ then
            * Calculates $P_i \leftarrow r_{2i} \| [(ID_{iold} - r_{2i}) \oplus (N_{iold} + r_{1i})]$.
            * Updates $N_{inew} \leftarrow N_i^{'}, ID_{inew} \leftarrow ID_i^{'}$.
        ∗ else
            * Generates $r_{2i}$, $P_i \leftarrow r_{2i} \|$fake data.
        ∗ end if
    – else
        * Generates $r_{2i}$, $P_i \leftarrow r_{2i} \|$fake data.
    – end if
• end if
```

Step 7: Communication from tag T_i to reader

```
• if $T_i$ is valid then
    – Sends $P_i$ to reader.
• else
    – Sends $P_i$ to reader with probability $\lambda$.
• end if
```

Step 8: Operation in reader

```
• Extracts $r_{2i}$ from $P_i$ for the responded tag $T_i$
• if $r_{2i} \notin A$ then
    – Ignores the message.
• end if
```

Step 9: Communication from reader to backend server

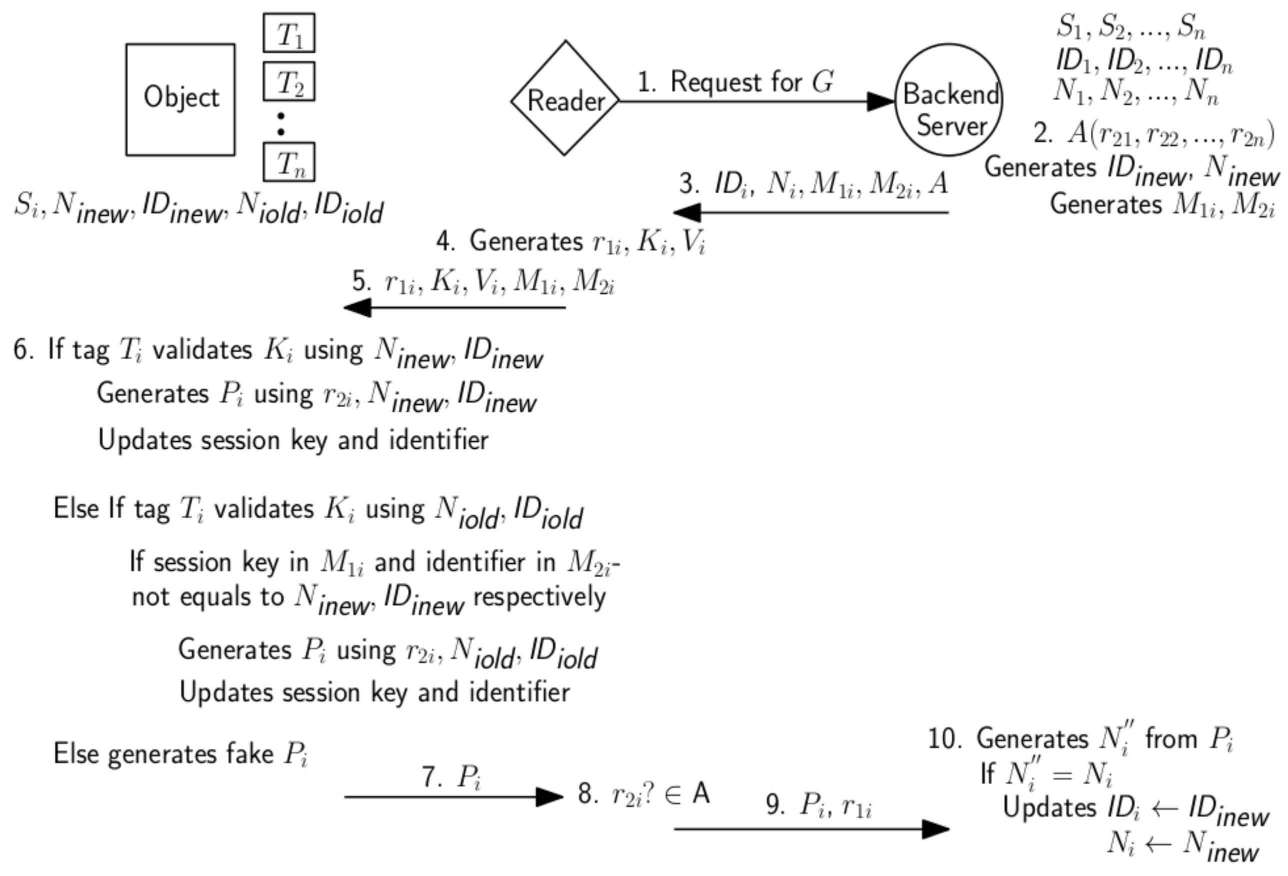
- **if** reader finds at least threshold number (l) of valid responses **then**
    - For a valid tag T_i, it forwards P_i, r_{1i} to backend server.
- **else**
    - Stops and reports object not found.
- **end if**

Step 10: Operation in backend server

- For tag T_i, separates $(ID_i'' - r_{2i}) \oplus (N_i'' - r_{1i})$ and r_{2i} from P_i.
- Locates P_i and r_{1i} to appropriate record in database indexed by G and r_{2i}.
- Calculates $(N_i'' - r_{1i}) \leftarrow [(ID_i'' - r_{2i}) \oplus (N_i'' - r_{1i})] \oplus (ID_i - r_{2i})$
- Calculates $N_i'' \leftarrow (N_i'' - r_{1i}) + r_{1i}$
- **if** N_i''=N_i **then**
    - The entry for T_i in record for G is valid
    - Updates $ID_i \leftarrow ID_{inew}, N_i \leftarrow N_{inew}$
- **endif**
- **if** at least threshold number (l) of valid entries found for G **then**
    - Reports search is successful
- **else**
    - Reports search is unsuccessful and stops.
- **end if**

Brief description: Figure 13 illustrates the proposed scheme for searching an object G attached with n number of tags. We briefly describe how search process takes place in our scheme. In initialization, each object is assigned with different set having n number of tags where each tag T_i is loaded with a pairwise

Figure 13: Tag searching scheme for multi tag RFID


Object
T_1
T_2
T_n
$S_i, N_{inew}, ID_{inew}, N_{iold}, ID_{iold}$
Reader
1. Request for G
Backend Server
$S_1, S_2, ..., S_n$
$ID_1, ID_2, ..., ID_n$
$N_1, N_2, ..., N_n$
2. $A(r_{21}, r_{22}, ..., r_{2n})$
Generates ID_{inew}, N_{inew}
Generates M_{1i}, M_{2i}
3. ID_i, N_i, M_{1i}, M_{2i}, A
4. Generates r_{1i}, K_i, V_i
5. $r_{1i}, K_i, V_i, M_{1i}, M_{2i}$
6. If tag T_i validates K_i using N_{inew}, ID_{inew}
Generates P_i using $r_{2i}, N_{inew}, ID_{inew}$
Updates session key and identifier
Else If tag T_i validates K_i using N_{iold}, ID_{iold}
If session key in M_{1i} and identifier in M_{2i}- not equals to N_{inew}, ID_{inew} respectively
Generates P_i using $r_{2i}, N_{iold}, ID_{iold}$
Updates session key and identifier
Else generates fake P_i
7. P_i
8. $r_{2i}? \in$ A
9. P_i, r_{1i}
10. Generates $N_i^{''}$ from P_i
If $N_i^{''} = N_i$
Updates $ID_i \leftarrow ID_{inew}$
$N_i \leftarrow N_{inew}$

secret key S_i, a tag id ID_i in both the fields ID_{inew} and ID_{iold}, and a session key N_i in both the fields N_{inew} and N_{iold}. The database in backend server is loaded with records for each object. The record for an object G contains the information about the tags assigned to it. The information are pairwise secret keys $S_1, S_2, ..., S_n$, tag ids ID_1, ID_2, ..., ID_n, and session keys $N_1, N_2, ..., Nn$.

During search process, the RFID reader requests the backend server to send information about the object G to be searched in step 1. The backend server generates a set A having n number of random numbers $(r_{21}, r_{22}, ..., r_{2n})$ in step 2 and then assigns the random number $r_{1i}, (i = 1, 2, ..., n)$ as index for the tag T_i in G. Therefore, a tag T_i in G has two index $\{G, r_{2i}\}$. It then generates new tag id ID_{inew} and new session key N_{inew} for T_i. Using these newly generated information and pairwise secret key, it generates the encrypted update information M_{1i} and M_{2i}. The backend server sends the ID, session key, and update information for each tag in G along with A to reader in step 3.

For each tag T_i in G, the reader generates a random number r_{1i} and then encrypts the ID and session key to generate K_i and V_i in step 4 using r_{1i} and r_{2i}. The reader then broadcasts a search request $\eta_1\eta_2..., \eta_n$, where $\eta_i = r_{1i}, K_i, V_i, M_{1i}, M_{2i}$ $(i = 1, 2, ..., n)$ in step 5.

Reachable tag T_i retrieves η_i from the broadcast information and checks the validity in step 6. To check the validity, firstly it uses the information in the fields ID_{inew} and N_{inew} in its memory and if validity confirms it extracts r_{2i} from V_i and generates the authentication information P_i in step 6 using the information in fields ID_{inew},N_{inew}, and r_{2i}. It copies the information in ID_{inew} and N_{inew} to ID_{iold} and N_{iold} respectively and updates ID_{inew} and N_{inew} using the ID and session key extracted from M_{1i} and M_{2i}. If validity does not confirm using the information in fields ID_{inew} and N_{inew}, it checks the validity using the information in fields ID_{iold} and N_{iold}. If validity confirms, it extracts the update information and checks whether the update information has already stored in its memory or not. If there is no match then it extracts r_{2i} from V_i and generates the authentication information using the information in fields ID_{iold}, N_{iold}, and r_{2i} and updates the information in fields ID_{inew} and N_{inew} using the ID and session key extracted from M_{1i} and M_{2i}. The tag T_i which had verified the request successfully is obviously attached with the desired object G. It then sends P_i in step 7 to reader. The tag T_i on which the verification of request had been failed is obviously attached with an undesired object. Therefore it generates a random r_{2i} and then sends a fake P_i having newly generated r_{2i} with a probability λ in step 7.

The reader filters the responses in step 8. It checks whether the r_{2i} in received response belongs to set A or not. The responses having $r_{2i} \in A$ are partially valid. The reader forwards the partially valid responses in step 9 to backend server if it finds at least threshold number (l) of partially valid responses.

In step 10, the backend server maps the responses using index $\{G, r_{2i}\}$ and checks the authentication of each response. It updates the session key and ID for a valid response using the ID and session key generated earlier in step 2. The backend server then reports that the search is successful if it finds threshold number (l) of valid responses. Otherwise it reports that the desired object is not found.

5 Analysis of the scheme

We have analyzed our scheme to see its applicability. To do this, we select the following parameters: authentication, security, computation, communication and memory.

5.1 Authentication

Since RFID is a pervasive computing technology, authentication is an important requirement during search of an object. In our scheme, we have assumed that the communication between reader and backend server is secure while the communication between reader and tag is insecure and hence there is a need to check authentication of those entities which are the source of any information communicates through the insecure medium. In step 6 of our proposed scheme, the tag T_i checks the authentication of r_{1i}, K_i, V_i, M_{1i}, and M_{2i} and hence checks the authentication of reader and backend server. In step 8, reader partially checks the authentication of P_i. In step 10, the backend server fully checks the authentication of P_i and hence the tag T_i is authenticated. Thus, the authentication of all the entities are checked in our proposed scheme.

5.2 Security analysis

The adversary may attempt to mount various kind of attacks in the communication between reader and object since it is assumed to be insecure. In this section, we have analyzed how our scheme prevents all the possible attacks (defined in section 3.2) during the search of an object.

a) *Eavesdropping:* During search process, the adversary listens r_{1i}, $(N_i - r_{1i}) \oplus (ID_i - r_{1i})$, $(r_{2i} - N_i) \oplus ID_i$, $(S_i - N_i) \oplus (ID_{inew} - S_i)$, $(S_i - ID_i) \oplus (N_{inew} - S_i)$, and P_i and may try to learn the secrets N_i, ID_i etc. However these secrets are bound with each other using XOR operation and hence the adversary is unable to learn any secret. If the adversary try to learn the secrets by compromising tags attached to an object, she needs to compromise at least a threshold number (l) of tags and no less. This increases the difficulty for the adversary to mount this kind of attack.

b) *Physical attack:* To mount physical attack, the adversary has to clone at least threshold number (l) of tags. Therefore, we do not claim that our scheme is fully secure from this kind of attack. However, it increases the difficulty for the adversary. Hence, our scheme partially fulfills this requirement.

c) *Traceability:* The security related information N_i, ID_i etc. are generated randomly and updated in each successful session in our scheme. Therefore, there is no relation between the P_i in one session and the P_i in another session and hence the adversary is unable to obtain any pattern from the responses. Thus, the objects are not traceable from the information communicated through insecure medium.

d) *Traceability within two successive and successful sessions:* Between two successful sessions, the adversary may try to send same search query r_{1i}, $K_i(= (N_i - r_{1i}) \oplus (ID_i - r_{1i}))$, $V_i(= (r_{2i} - N_i) \oplus ID_i)$, $M_{1i}(= (S_i - N_i) \oplus (ID_{inew} - S_i)$, $M_{2i}(= (S_i - ID_i) \oplus (N_{inew} - S_i))$ multiple times and the tags are expected to respond with same P_i and thus the adversary may trace an object during this period. According to our scheme, if the tag gets same search query the verification of the query will be successful using N_{iold} and ID_{iold} stored in the tag. However, the retrieved session key N_i' and identifier ID_i' in M_{1i} and M_{2i} will be equal to the N_{inew} and ID_{inew} respectively stored in the tag since the same session key and identifier were retrieved in previous session and stored as new session key and identifier. Hence the tag sends a fake P_i which has no relation with the P_i sent as a response to the previous query. Thus the adversary is unable to get any pattern from the responses during the period between two successful sessions.

e) *Man in the middle attack*: Blocking the original $r_{1i}, K_i, V_i, M_{1i}, M_{2i}$ and P_i, the adversary may send fake $r_{1i}, K_i, V_i, M_{1i}, M_{2i}$ and P_i through insecure medium. However, the fake information will be discarded since the adversary does not have any secrets N_i, ID_i, and S_i and hence they cannot generate and send any information that can be validated.

f) *Forward security:* In our scheme, the session key N_i and identifier ID_i are changing in each successful session, i.e. these secrets are variable in nature. The adversary is unable to calculate N_i or identifier ID_i or both for the next sessions using the learnt variable secrets in current session due to the presence of fresh r_{1i} and r_{2i} and the secret key S_i in $K_i, V_i, M_{1i}, M_{2i}, P_i$. Therefore, our scheme provides the forward security benefit.

g) *Backward security*: Use of fresh r_{1i}, r_{2i}, and S_i prevents the adversary to calculate the variable secrets used in past sessions. Thus, our scheme satisfies the backward security requirement.

h) *Replay attack*: The adversary may store r_{1i}, $K_i(= (N_i - r_{1i}) \oplus (ID_i - r_{1i}))$, $V_i(= (r_{2i} - N_i) \oplus ID_i)$, $M_{1i}(= (S_i - N_i) \oplus (ID_{inew} - S_i)$, and $M_{2i}(= (S_i - ID_i) \oplus (N_{inew} - S_i))$ and send later to tag. If the original session has completed successfully, the replayed information will be verified using N_{iold} and ID_{iold}. In the case of incomplete original session, the replayed information will be verified using N_{inew} and ID_{inew}. In both cases, the tag in the desired object will respond with P_i after successful verification which includes r_{1i} of original session. On a new request from reader, the adversary may try to send this P_i. However, this P_i will not be verified since the reader have used the new nonce r_{1i} to send the new request. Also the adversary cannot inject new nonce since she does not have secret keys.

i) *Information leakage:* In traditional search process, only the tags of desired object would reply in response to a search by reader. According to this process, the adversary may not be able to obtain any secure information, however, she can be able to obtain the information about its presence. In some situations, this information may be valuable to the adversary. In our scheme, the undesired tags respond[2] with fake

[2]An undesired tag responds with probability λ

information ($P_i = r_{2i} \| \text{fake data}$). Since the adversary does not have any secret key she cannot distinguish between the fake response and legitimate response. Therefore the adversary cannot conclude about the existence of the desired object.

j) *Synchronization attack:* If adversary blocks P_i, the tag contains new session key and identifier which were updated after successful authentication of $r_{1i}, K_i, V_i, M_{1i}, M_{2i}$. However the backend server cannot update the session key and identifier for the same tag in its database and will contain the old copies. Thus it seems that there can be a synchronization problem. To avoid this problem, we keep old session key and identifier in tag along with the updated session key and identifier. Thus the server can send a request using old session key and identifier and the tag can successfully verify it using old session key and identifier stored in its memory and update accordingly. Hence it will respond and the backend server will update accordingly. Therefore, there is no synchronization problem in our scheme.

5.3 Comparison

We have chosen four parameters, namely, security, computation, communication and memory requirements for comparing our scheme with the existing schemes we have described in section 2.

5.3.1 Security requirements

We have already discussed the assurance of security in our scheme in section 5.2. In this section we provide a comparative study.

In Table 2, we have specified the security requirements with the symbols a, b, c etc. The meaning of symbols are written under the table. Each row indicates the scheme under discussion. Each entry in the table indicates the prevention status of security requirement in corresponding column for the scheme in corresponding row. There are three type of status and they are satisfy, not satisfy, and partially satisfy. The symbols are Y, N, and P respectively.

Our scheme satisfies all the security requirements except the physical attack. However, it increase the difficulty for the adversary to mount this kind of attack. The other schemes [8] [9] [10] [11] [12] in Table 2 do not satisfy all the security requirements and hence in security aspect, our scheme is stronger than the existing schemes.

Resiliency: We define a parameter, namely, resiliency that indicates the strength of a scheme in terms of attack prevention against compromise of various security related information. We assume that the adversary tries to mount various kind of attacks after compromising the security related information such as session key, ID, etc. Compromise of a few information may not help her to mount all possible attacks. However, she can manage to mount some attacks. Therefore, we incrementally compromise the best combination of security related information and analyze how many attacks are still preventable for a particular search scheme. Here best combination implies that the combination of a certain number of information which helps the adversary to mount maximum number of attacks. The other combination of same number of information may not help the adversary to mount as many attacks. We define resiliency as the number of attacks which are preventable on compromise of a certain number of security related information.

Table 3, illustrates the resiliency of the existing schemes [8] [9] [10] [11] [12] and our scheme. A row indicates the the scheme under consideration and a column indicates the number of messages have been compromised. Therefore an entry in the table indicates the number of attacks prevented by the scheme in corresponding row on compromise of the number of messages in corresponding column. We have analyzed the related schemes [8] [9] [10] [11] [12] and our scheme by compromising the best combination of security related information incrementally and deduced the resiliency. We have assumed that the threshold value for our scheme is 3. According to this analysis, Table 3 shows that our scheme have maximum resiliency without compromise of any information. The resiliency level continues high until the compromise of threshold number of information. This is because the compromise of single information for a single tag allows to mount all the attacks for that tag. Thus, to mount attacks for an object the adversary requires to compromise at least one information from each of the threshold number of tags. If we increase the threshold value then the resiliency of our schemes also increases accordingly. However, for the schemes [9] [10] [11] [12], the resiliency level falls down on compromise of less than 3 information. For schemes [9] [12], the resiliency level falls down quickly even before the compromise of 2 information. The resiliency of our scheme is directly proportional to the threshold value and hence can be tuned according to the requirement. The other schemes do not have this facility.

5.3.2 Computational overhead

Various protocols performs various operations. We have selected the operations we have used in our scheme and the other additional operations used in other schemes. Table 4 shows the quantitative analysis of various operations performed in [8] [9] [10] [11] [12] and in our proposed scheme. The operations we have considered in our analysis are XOR, hash, random number generation, attachment/detachment, modulus, addition/subtraction, and others indicated by a, b, c, d, e, f, and g respectively. The components involved in RFID communication are RFID tags, RFID reader, and backend server. We have further classified the tags in our analysis as desired and undesired. This is because the second category of tags are silent or do some other operations which are not common with the operations performed by first category of tags. Table 4 has four columns, namely, desired tag, undesired tag, reader, and backend server. Each column is further divided into seven sub columns. A sub column represents how many time the component specified in corresponding column performs a particular operation specified in it. Each row in Table 4 represents the scheme under consideration in our analysis. Therefore, an entry in the table indicates how many time a component specified in the corresponding column in the scheme specified in corresponding row performs an operation specified in corresponding sub column. We explore a comparative study by analyzing the entries in Table 4.

The desired tag in our scheme does not require to execute any computationally expensive operations such as random number generation and hash operation, however it requires to compute maximum number of light-weight operations i.e, XOR and addition/subtraction. The other schemes, on the other hand, requires one or more hash computations and many random number generations for desired tags which are very time consuming operations. Therefore, the desired tags in our scheme are efficient in compared to other schemes [8] [9] [10] [11] [12] since our scheme requires to compute no heavy weight operations.

The undesired tag in our scheme needs to generate only one random number and less number of XOR and addition/subtract operations. However, it does not requires to compute any hash operations. Therefore the undesired tags in our scheme are also efficient compared to other schemes [8] [9] [10] [11] [12].

The reader in our scheme does not compute any hash operation. It requires to compute more light operations i.e. XOR, addition/subtraction etc. compared to the schemes [8] [9] [10] [11] [12]. This is due to the fact that more than one tags are attached in an object and hence reader needs to find out at least threshold number of desired responses.

The backend server in our scheme also does not require to compute any hash operation. It requires to compute more number of operations compared to other scheme [8] [9] [10] [11] [12] since it has to generate and process search information for more than one tag for a desired object. We assume that the server is a high speed stationary computer which does not suffer from computation constraint.

From this comparative discussion, we conclude that in comparison to other schemes, our scheme is efficient in terms of generating a response by a tag. The reader and backend server are less efficient than other schemes due to the fact that an object is attached with multiple number of tags and hence to search an object, the reader and backend server need to compute various operations for every tag attached to that object.

5.3.3 Computational overhead due to responses from undesired tags

An adversary may try to know the existence information of the desired object. The adversary can be able to get this information if only the tags in desired object respond. Therefore, although the response from desired object is secure it leaks some information to the adversary. To prevent this kind of attack, the undesired tags respond with fake information. Therefore the adversary cannot distinguish between a fake and legitimate response. However, the legitimate reader and/or backend server has/have to distinguish the legitimate response from all the responses. To do this, they have to process all the responses which introduce an useless computation overhead.

Probability of useless computation in our scheme: In our scheme, the reader partially checks r_{2i} in each response for its validity. The tags in desired object extracts a valid r_{2i} and sends it to reader. Therefore this response is validated and forwarded to backend server. The reader validates simply by comparing the received r_{2i} with the random numbers in A. Therefore, there is no such computational overhead in it. However, the tag which had sent fake response has used a random r_{2i} which may be a member of A. Therefore any response with such kind of r_{2i} would be forwarded to backend server and backend server has to process those fake responses. This introduces an useless computational overhead in backend server. We have computed the probability of such kind of useless computation in our scheme as follows. Let us define various symbols and then deduce the probability.

$P(X)$ is the probability of event X, E_1 be an event that an invalid tag generates a valid r_{2i}, E_2 be an event that an invalid tag responds, E_3 be an event that an invalid tag responds with valid r_{2i}, E_4 be an event that a valid tag responds with valid r_{2i}, E_5 be an event of useless computation in backend server, n be the number of valid r_{2i}, b be the number of bits constitutes a r_{2i}, M_{iv} be the number of invalid tags and M_v be the number of valid tags.

$$P(E_1) = \frac{n}{2^b} \tag{1}$$

Let

$$P(E_2) = \lambda \tag{2}$$

Therefore,

$$P(E_3) = \frac{\lambda n}{2^b} \tag{3}$$

Since the number of invalid tags is M_{iv}, the number of fake responses with valid r_{2i} is

$$\mathit{fake} = M_{iv} P(E_3) = \frac{\lambda n M_{iv}}{2^b} \tag{4}$$

Since a valid tag extracts a valid r_{2i}, it responds with valid r_{2i} and hence $P(E_4) = 1$ and the number of responses from valid tags with valid r_{2i}

$$\mathit{valid} = M_v P(E_4) = M_v \tag{5}$$

Since the reader passes only the responses with valid r_{2i} the total number of responses from valid and invalid tags that passes through the reader

$$\mathit{total} = \mathit{valid} + \mathit{fake} = M_v + \frac{\lambda n M_{iv}}{2^b} \tag{6}$$

From equations 4 and 6, we can say that

$$P(E_5) = \frac{\frac{\lambda n M_{iv}}{2^b}}{M_v + \frac{\lambda n M_{iv}}{2^b}} = \frac{\lambda n M_{iv}}{M_v 2^b + \lambda n M_{iv}} \tag{7}$$

Probability of useless computation in other schemes [9] [10] [12]: To prevent information leakage attack, the other schemes allows the tags in undesired objects to respond with fake information while the tag in desired object responds with valid information. however, the reader or backend server has to process all the responses which introduces an useless computational overhead. We have computed the probability of such kind of useless computation in other schemes as follows. We define the symbols we have used to deduce the probability.

$P(X)$ is the probability of event X, E_6 be an event that an invalid tag responds with fake information, E_5 be an event of useless computation, M_{iv} be the number of invalid tags.
Let

$$P(E_6) = \lambda \tag{8}$$

Since the number of invalid tags is M_{iv}, the number of fake responses is

$$\mathit{fake} = M_{iv} P(E_6) = \lambda M_{iv} \tag{9}$$

Since there is only one valid tag and that tag responds with valid information. Therefore, the number of valid response

$$\mathit{valid} = 1 \tag{10}$$

Therefore, the total number of response

$$\mathit{total} = \mathit{valid} + \mathit{fake} = 1 + \lambda M_{iv} \tag{11}$$

From equations 9 and 11, we can say that

$$P(E_5) = \frac{\lambda M_{iv}}{1 + \lambda M_{iv}} \tag{12}$$

Comparison: Table 5 illustrates the probability of useless computation for $n = 8, M_v = 8, \lambda = \frac{1}{2}$ and various number of invalid tags(M_{iv}). From the table, we see that the probability that the backend server has to process useless computation in our scheme is only 0.0007 for 100 invalid tags. However, the probability of useless computation in other schemes which allows the undesired tags to respond with fake information is 0.9803 for the same number of invalid tags. Therefore our scheme prevents information leackage attack with negligible amount of useless computation in backend server.

5.4 Message communication

Backend server, reader and tags communicate messages with each other. We have made an analysis in terms of number of messages communicated. Table 6 shows the comparison of existing schemes and our scheme in terms of communication overhead. In Table 6, the columns indicate the components involved in communication and each row explores the schemes involved in our comparison. Therefore, each entry in the table indicates the number of messages communicated by the components specified in the corresponding column in the scheme specified in corresponding row. The communication overhead for desired or undesired tags in our scheme is almost equals in comparison to the schemes [8] [9] [11]. However, it is less than the schemes [10] [12]. The communication overhead in reader and backend server is more than the other schemes since an object in our scheme is attached with multiple number of tags.

5.5 Memory requirement

The RFID tags has limited memory and hence we have analyzed the memory requirement of various schemes and Table 7 illustrates the comparative study. Besides listing the memory requirement in RFID tag, we have also listed the memory requirements in RFID reader and backend server which do not have such limitation. Each row in Table 7 represents a particular scheme and each column represents a particular components. Therefore, an entry in the table indicates the number of messages need to be kept in a component mentioned in corresponding column for the scheme mentioned in corresponding row. Requirement of memory in tag for our scheme is more since we keep old information to prevent synchronization attack. The memory requirement in backend server is more due to the fact that an object is attached with multiple number of tags and hence we have to keep information about all the tags attached to an object. Since the backend server does not suffer from memory resource constraint our scheme is practically applicable. There is no memory requirement in reader for our scheme whereas almost all the other schemes have this requirement.

6 Conclusion

We have proposed a light-weight object searching scheme where an object is attached with multiple number of tags. We take the advantage of multiple resources in same object. This has increased the security since the adversary faces difficulty to compromise an object. This is because the adversary needs to compromise threshold number of tags instead of single tag to compromise an object. Also she finds it difficult to clone the threshold number of tags in certain cases. Our scheme is efficient in terms of computation in RFID tag since we have used `XOR` and plus/minus operation for encryption and decryption process and only one random number generation process in RFID tag. The resources in tag support to do these operations. There is a negligible amount of useless computation in our scheme to achieve the requirement of prevention against information leakage problem. A threshold number of tags in an object need to be visible in our scheme limits the detection probability in multi-tag arrangement, however, increases the security. Therefore, an appropriate threshold value can balance between the security and detection probability and the selection of this value can be done depending on the application environment.

References

1. Ngaia EWT, Karen KLM, Frederick JR, Candace YY: **RFID research: An academic literature review (1995–2005) and future research directions**. *International Journal of Production Economics* 2008, **112**(2):510–520.
2. He L, Gan Y, Cai Z, Li N: **An Improved Lightweight RFID Protocol Using Substring**. In *5th International Conference on Wireless Communications, Networking and Mobile Computing* 2009:3717–3720.
3. Boyeon S, Chris JM: **RFID Authentication Protocol for Low-cost Tags**. In *WiSec'08, Alexandria, Virginia, USA* 2008:140–147.
4. Hyung-Joo K, Moon-Seog J: **Light-Weight Mutual Authentication RFID Protocol for Multi-Tags conforming to EPC Class-1 Generation-2 Standards**. In *5th International Conference on Computer Sciences and Convergence Information Technology* 2010:34–39.
5. Jing HK, Widad I, Mohammed IY, Sulaiman MK, Mohammed GR: **Security Problems in an RFID System**. *International Journal of Wireless Personal Communications* 2011, **59**:17–26.
6. Subhasish D, Indranil S: **A New Authentication Protocol for Multi-tag RFID Applicable to Passive Tag**. In *2nd International Conference on Communication, Computing & Security* 2012:880–888.
7. Leonid B, Gabriel R: **Multi-Tag RFID Systems**. *International Journal of Internet Protocol Technology (IJIPT), Special issue on "RFID Technologies, Applications, and Trends"* 2007, **2**(3/4):218–231.
8. Tan CC, Bo S, Qun L: **Secure and Serverless RFID Authentication and Search Protocols**. *International Journal of IEEE Transactions on Wireless Communications* 2008, **7**(3):1400–1407.
9. Kulseng L, Zhen Y, Yawen W, Yong G: **Lightweight Secure Search Protocols for Low-cost RFID Systems**. In *29th IEEE International Conference on Distributed Computing Systems* 2009:40–48.
10. Hoque ME, Farzana R, Sheikh IA: **S-Search: Finding RFID Tags using Scalable and Secure Search Protocol**. In *2010 ACM Symposium on Applied Computing* 2010:439–443.
11. Yoon HS, Heung YY: **An Anonymous Search Protocol for RFID Systems**. *International Journal of Convergence Information Technology* 2011, **6**(8):44–50.
12. Zheng Y, Mo L: **Fast Tag Searching Protocol for Large-Scale RFID Systems**. In *19th IEEE International Conference on Network Protocols* 2011:363–372.
13. Safkhani M, Pedro P, Nasour B, Majid N, Julio CH: **On the Security of Tan et al. Serverless RFID Authentication and Search Protocols**. In *8th International Workshop, RFIDSec 2012* 2012:1–19.

Tables

Table 1: Notations

Symbol	Meaning
G	An object
T_i	i^{th} Tag in G
n	Number of tags attached in G
ID_i	ID of i^{th} tag in G
N_i	Session key of i^{th} tag in G
S_i	Pairwise secret between backend server and i^{th} tag in G
A	A set of random numbers
r_{1i}, r_{2i}	Random numbers for i^{th} tag in G
ID_{inew}	New ID of i^{th} tag in G
N_{inew}	New session key of i^{th} tag in G
$\oplus$	XOR operation
$\Vert$	Concatenation operation
l	Threshold value

Table 2: Assurance of Security

	a	b	c	d	e	f	g	h	i	j
Hoque et al.	Y	N	Y	N	Y	N	N	Y	Y	Y
Kulseng et al. Protocol 1	Y	Y	Y	N	Y	N	N	Y	N	N
Kulseng et al. Protocol 2	Y	Y	Y	N	Y	N	N	Y	N	Y
Kulseng et al. Protocol 3	Y	Y	Y	N	Y	N	N	Y	Y	Y
Yoon et al.	Y	N	Y	N	Y	P	P	Y	N	Y
Tan et al. Protocol 1	N	N	N	N	N	Y	Y	N	N	Y
Tan et al. Protocol 2	N	N	N	N	N	Y	Y	N	N	Y
Tan et al. Protocol 3	Y	N	N	N	N	Y	Y	Y	P	Y
Tan et al. Protocol 4	N	N	N	N	N	Y	Y	N	N	Y
Zheng et al.	Y	N	Y	Y	N	Y	Y	Y	Y	Y
Our scheme	Y	P	Y	Y	Y	Y	Y	Y	Y	Y

a: Eavesdropping, *b*: Physical attack, *c*: Traceability, *d*: Traceability between successful-
e: Man in the middle attack, *f*: Forward security, *g*: Backward security, *h*: Replay attack,
i: Information leakage, *j*: Synchronization attack, *Y*: Satisfy, *N*: Not satisfy, *P*: Partially satisfy

Table 3: Resiliency Statistics

Number of compromised messages	0	1	2	3
Hoque et al.	8	8	3	3
Kulseng et al. scheme 1	7	0	0	0
Kulseng et al. scheme II	8	0	0	0
Kulseng et al. scheme III	9	1	1	1
Yoon et al.	7	7	1	1
Tan et al. scheme I	3	3	3	3
Tan et al. scheme II	3	3	3	3
Tan et al. scheme III	5	3	3	3
Tan et al. scheme IV	3	3	3	3
Zheng et al.	5	1	1	1
Our scheme	10	10	10	1

Table 4: Number of operations performed in various scheme

	Desired Tag							Undesired tag							Reader							Backend Server						
	a	b	c	d	e	f	g	a	b	c	d	e	f	g	a	b	c	d	e	f	g	a	b	c	d	e	f	g
Hoque et al.	3	3	1	0	1	0	0	2	2	0	0	1	0	0	2	2	0	0	0	0	BR	1	1	1	0	1	0	0
Kulseng et al. Protocol 1	5	6	0	0	0	0	0	2	1	0	0	0	0	0	5	4	1	0	0	0	0	0	0	0	0	0	0	0
Kulseng et al. Protocol 2	4	5	0	0	0	0	0	2	1	0	0	0	0	0	4	3	1	0	0	0	0	0	0	0	0	0	0	0
Kulseng et al. Protocol 3	4	5	0	0	0	0	0	2	1	2	0	0	0	0	$4f_1$	$3f_1$	1	0	0	0	0	0	0	0	0	0	0	0
Yoon et al.	0	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	INC
Tan et al. Protocol 1	2	3	1	3	0	0	0	1	2	0	2	0	0	0	2	2	1	3	0	0	0	0	0	0	0	0	0	0
Tan et al. Protocol 2	2	3	1	3	0	0	0	1	2	0	2	0	0	0	2	2	1	3	0	0	0	0	0	0	0	0	0	0
Tan et al. Protocol 3	1	1	1	2	0	0	0	1	1	1	2	0	0	0	f_1	f_1	1	$2f_1$	0	0	0	0	0	0	0	0	0	0
Tan et al. Protocol 4	2	3	3	1	0	0	0	1	2	2	2	0	0	0	$1+f_1$	$1+f_1$	1	$1+2f_1$	0	0	0	0	0	0	0	0	0	0
Zheng et al.	0	k_1+k_2	0	0	0	0	0	k_1	0	0	0	0	0	0	0	0	0	0	0	0	BF_2	0	k_1+k_2	2	0	0	0	0
Our scheme	6	0	0	1	0	11	0	2	0	1	1	0	4	0	$2n$	0	n	f_1	0	$3n$	0	$2n+\delta+\alpha$	0	n	$\delta+\alpha$	0	$4n+\delta+\alpha$	0

a: XOR, *b*: Hash, *c*: Random number generation, *d*: Attachment/Detachment, *e*: Modulas, *f*: Addition/Subtraction, *g*: Other, f_1: Number of tags replied, INC: Increment, k_1, k_2: Number of hash functions used in bloom filter, α: Number of false positive response, δ: Number of valid response to backend server, *n*: Number of tags attached to an object

Table 5: Probability of useless computation in backend server

Number of invalid tags	10	20	30	40	50	60	70	80	90	100
$P(E_5)$ for our scheme	0.000076	0.000153	0.000229	0.000305	0.000381	0.000458	0.000534	0.000610	0.000686	0.000762
$P(E_5)$ for other schemes	0.833333	0.909091	0.937500	0.952381	0.961538	0.967742	0.972222	0.975610	0.978261	0.980392

Table 6: Communication overhead of various scheme

	Desired Tag	Undesired tag	Reader	Backend Server
Hoque et al.	$f+5$	$f+3$	$f+k+6$	3
Kulseng et al. Protocol 1	5	3	5	0
Kulseng et al. Protocol 2	4	2	4	0
Kulseng et al. Protocol 3	4	2	$2(f_1+1)$	0
Yoon et al.	6	3	12	6
Tan et al. Protocol 1	5	3	5	0
Tan et al. Protocol 2	5	3	5	0
Tan et al. Protocol 3	5	3	$3+2f_1$	0
Tan et al. Protocol 4	5	3	$3+2f_1$	0
Zheng et al.	k_2+8	4	$15+k_2+f_1$	8
Our scheme	6	6	$9n+2(\delta+\alpha)+f_1+2$	$4n+2(\delta+\alpha+1)$

f: Length of Bit Record(BR) in Hoque et al., k: Number of single or collied reply, f_1: Number of tags replied, k_2: Number of slots in Zheng et al., δ: Number of valid responses to backend server, α: Number of false positive responses, n: Number of tags attached to an object

Table 7: Memory requirement

	Tag	Reader	Backend Server
Hoque et al.	2	2β	2β
Kulseng et al. Protocol 1	2	4β	0
Kulseng et al. Protocol 2	3	3β	0
Kulseng et al. Protocol 3	3	3β	0
Yoon et al.	3	0	$2\beta+1$
Tan et al. Protocol 1	2	$2\beta+1$	$3\beta+1$
Tan et al. Protocol 2	$2+l$	$2\beta+1$	$3\beta+1$
Tan et al. Protocol 3	2	$2\beta+1$	$3\beta+1$
Tan et al. Protocol 4	2	$2\beta+1$	$3\beta+1$
Zheng et al.	1	0	β
Our scheme	5	0	$3\beta n$

l: Number of completed sessions, β: Number of objects n: Number of tags in each object