Privacy Protection for RFID-based Tracking Systems

Chiu C. Tan
Department of Computer Science
College of William and Mary
cct@cs.wm.edu

Lei Xie
State Key Laboratory of
Novel Software Technology
Nanjing University
xielei@dislab.nju.edu.cn

Qun Li
Department of Computer Science
College of William and Mary
liqun@cs.wm.edu

***Abstract*—RFID technology is increasingly being deployed in ubiquitous computing environments for object tracking and localization. Existing tracking architecture usually assumes the use of a *trusted server* which is invulnerable to compromise by internal and external adversaries. However, maintaining such a trusted server is unlikely in the real world. In this paper, we consider the problem of adding privacy protection to object tracking systems built upon passive RFID tags, without relying on a trusted server assumption. Our protocol continues to protect user privacy in the event of partial compromise of a server.**

I. Introduction

The low cost and rugged design of RFID tags make them suitable to be attached to everyday objects such as key chains and coffee mugs. With every object embedded with its own RFID tag, ubiquitous computing concepts like object tracking and localization or location based services can be realized [1], [2], allowing many new applications to be developed upon this infrastructure. At the same time, widespread RFID deployment creates privacy risks for everyone, not just a few willing early adopters, thus the need for privacy protection becomes critical.

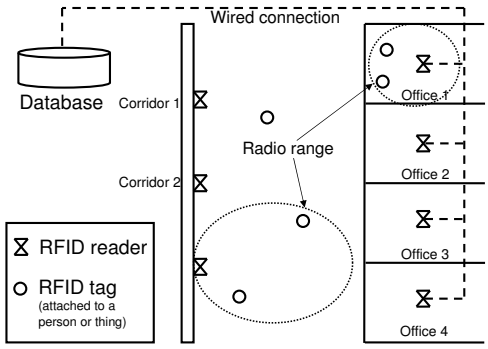

Fig. 1. Typical RFID tracking system

Fig. 1 shows an environment where RFID tags are attached to people and objects. Each RFID tag has an associated unique ID which represents that tag in the system. A network of RFID readers are deployed in the building, with each reader associated with a specific location, such as "Office 1" or "Corridor 2". The readers are connected to a database server via a wired network. A reader will periodically broadcast a query, and all the RFID tags within the vicinity will respond to that reader. The reader will then forward these tag responses to the database. A user wanting to know for instance who was at a meeting in Office 1 at 1:00 pm can query the database to determine the IDs of the tags, and thus the identity of the meeting participants.

The potential for abuse of such a powerful tracking system is apparent. The straightforward approach is to limit the access to the database data using passwords. Each user is associated with a list of RFID tags they are authorized to access, and must supply the correct password when querying the database. For example, a user with ID=101 running a query "`Select` location `From` database `Where` ID=101 and Time=time" must first supply the appropriate permissions for his tag before the location information is released. More advance techniques such as role-based access control [3], [4], and Hippocratic databases [5], can also be used to protect the database.

This type of solutions have a common requirement. **A trusted database is needed to maintain and enforce the privacy policy.** Without a database that is invulnerable to compromise by criminal hackers or disgruntled employees, and always managed by competent system administrators, we cannot protect user privacy. We believe that maintaining such a trusted server is difficult in practice. Even three letter government agencies with strong emphasis on security have been vulnerable to database breaches or mistakes [6], [7], [8].

In this paper, we consider the problem of how to build a tracking system that does not rely on a trusted database to protect user privacy. Our solution divides up the database operations into separate servers such that we can tolerate the compromise of any one of the databases. Our protocols are designed to take into account the computationally weak capabilities of an RFID tag, and allow a user to efficiently query the database for answers.

We make the following contributions in this paper. We propose a technique to divide a database query, originally intended to a single database, into multiple databases such that if only one of the database is compromised, the user's privacy is protected. We also consider how to detect, but not prevent, adversary disruption attacks such as the data deletion and manipulation.

The rest of the paper is as follows. We formulate our problem and adversary model in Section II. Section III introduces

two strawman protocols to motivate our solutions. Sections IV presents our protocol against different types of adversaries. Section V considers additional security issues not directly related to user privacy. Finally we review the related work in Section VI and conclude in Section VII.

II. Related Work

Privacy protection is an important component in ubiquitous computing environments [9], [10]. The technique of using mix zones was proposed by [11], [12] where by users could specify certain areas where nobody could trace their movements. Other researchers [13], [14], [15] proposed techniques to help users define privacy policies. This approach is more flexible than mix zones at the cost of additional complexity in specifying the policies. The idea of allowing users to specify "virtual walls" was suggested by [16] to simplify the creation of a privacy policy. Physical access control [17], [18] addresses the problem of specifying a privacy policy. Users can only obtain the location information of people that are were present together at the same time. The intuition is that users that were at the same location at the same time already know each other's presence, and thus there is no privacy issues when releasing that information later. Our work differs from these proposed techniques in that we do not rely on trusted servers to protect user privacy. Our idea of protecting privacy by separating location, time, and identity is similar to that proposed by [19], but our solutions are designed to work with RFID tags.

RFID security is an active area of research with many different protocols being proposed [20], [21], [22]. While our paper also proposes a simple security protocol, our focus is less on the security and privacy between RFID reader and tag, but oriented more towards data already collected and archived.

Closely related to our paper is research on searching encrypted data. In this problem, a user encrypts his data and stores it at an untrusted server. The user wants to be able to search of part of his data in an efficient manner. Since the server is untrustworthy, the user cannot send over his secret key. The user also cannot request the server to transmit all the encrypted data back since it is inefficient. An search system using symmetric key to encrypt data was proposed by [23], while [24] suggested a public key based scheme. Practical encrypted database query retrieval systems were proposed by [25], [26]. However, unlike our paper, prior research in this area do not consider the privacy implications of ubiquitous environments such as malicious tracking of users. This was shown in the second strawman approach by using [26] as an example. Furthermore, these prior techniques assume that more advance hardware such as laptops are used, rather than computational weak RFID tags.

III. Problem Formulation

We consider a network of RFID readers, $R_1, \cdots, R_n$, are deployed throughout a facility. Each reader is programmed to periodically broadcast a query to read all the RFID tags within its vicinity. The captured data is then forwarded to a backend database for storage. In the unencrypted scenario, the database stores this information as a $ID : time : location$ tuple, where ID is a number that identifies that RFID tag, time is the time that tag was read, and location is the physical location corresponding to that particular reader. We assume that the database knows the locations of all the RFID readers, and can associate the data from a reader to a location.

TABLE I
Unencrypted table in database

	ID	Time	Location
1.	101	10:00am	Office 1
2.	102	10:00am	Office 3
3.	101	10:15am	Office 2
...	...	...	...

The database can represent the data from all the RFID readers as a table with 3 attributes,ID, Time, and Location. Table I illustrates this database. A user who own tag 101, can query this database later by issuing a query,

`Select * from` DB `where` ID=101 and Time=10:00 am

and determine he was at Office 1 at 10 am.

Such a setup provides no privacy protections, since anyone can query the database to determine anybody's movements. Under the trusted server assumption, we can protect privacy by associating a password with each ID. The user running the above query for instance, will have to supply the correct password associated with ID=101, before the database will release the information. While more complex schemes can be designed to provide better access control, a fundamental requirement is that the database *cannot* be compromised. An adversary with access to Table I learns everything.

In this paper, we consider the problem of how to protect privacy in the event of such a server is compromised by an adversary.

A. Adversary model

We assume that the adversary seeks only to track the movements of a user. The adversary succeeds if he is able to extract the identity of the user from the database, or if he is able to link two entries in the database to the same user.

We assume that the adversary can have free access to the database data such as in Table I, as well as observe the database interactions between a user and the database. The adversary is however, unable to determine the identity of someone querying the database. For instance, the adversary cannot deduce a user identity through the MAC address of the user's device when the user is querying the database. Techniques such as an anonymizing network [27] can be deployed to achieve this. Also, the adversary cannot reprogram the database to execute functions it otherwise will not perform.

In our paper, we assume that the RFID tags are able to perform simple operations such as generating random numbers, perform a hash function, and XOR two bitstrings together. These are common assumptions made in RFID security literature [28], [29], [30]. While our solution in the paper is

presented using hash functions, symmetric key encryption can be substituted instead with minor modifications.

Finally, we assume that having a rational adversary precludes attacks such as deleting or shuffling entries in the database since such actions do not help the adversary identify a user. This is reasonable since such disruption attacks increases the risk of detection. We will discuss more about defending against such attacks in the Section IV.

IV. Strawman solutions

We motivate our solutions by considering the limitations of seemingly possible solutions. For all these strawman solutions, we assume a more powerful RFID tag that can do symmetric key encryption is used. We always assume the user is the owner of RFID tag 101.

A. Strawman 1

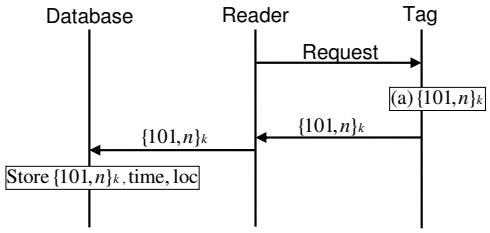

Fig. 2. Strawman protocol 1. The tag ID is 101.

In this simple protocol, we let the RFID tag return its encrypted ID in the form of $\{ID, n\}_k$ where n is a random number, and k is the tag's secret key. For completeness, we show this interaction in Fig.2. Now, in the database, we have

TABLE II
Database table from Strawman 1

	ID	Time	Location
1.	$\{101, n_1\}_{k_1}$	10:00am	Office 1
2.	$\{102, n_2\}_{k_2}$	10:00am	Office 3
3.	$\{101, n_3\}_{k_1}$	10:15am	Office 2
...	...	...	...

where k_1 and k_2 are the secret keys of tags 101 and 102 respectively. We see that this approach protects privacy since the adversary observing this table cannot learn the actual IDs, 101 and 102, of the tags. There is also no linkability, since $\{101, n_1\}_{k_1}$ and $\{101, n_3\}_{k_1}$ use different random numbers n_1 and n_3 while encrypting the same ID, thus resulting in different ciphertexts.

The problem with this solution arrises when the user wants to query the database. He can no longer simply do a "`Select * from` DB `where` ID=101", since all the ID attributes now have a random number component. The user cannot recall the n_1 and n_3 values used since the limited storage capacity of the RFID tag means these random numbers have to be generated on-the-fly and never archived. The only option is for the user to retrieve the *entire* table, and attempt to decrypt each entry's ID field until he finds the all the entries with IDs equal to his own. This is clearly inefficient given the large size of the table.

B. Strawman 2

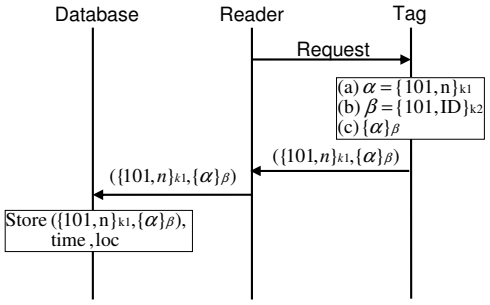

Fig. 3. Strawman protocol 2. The tag ID is 101.

We can modify a technique from [26] to improve the query performance. Now, each RFID tag maintains two keys that it keeps secret, i.e. tag 101 will have $k1^{101}, k2^{102}$. We remove the superscript and use $k1$ and $k2$ to denote keys to simplify the presentation. The protocol is shown in Fig. 3, and after the data is collected, the table resembles Table III.

TABLE III
Database table by Strawman 2

	ID	Time	Location
1.	$\{101, n_1\}_{k1}, \{\alpha_1\}_{\beta_1}$	10:00am	Office 1
2.	$\{102, n_2\}_{k1}, \{\alpha_2\}_{\beta_2}$	10:00am	Office 3
3.	$\{101, n_3\}_{k1}, \{\alpha_3\}_{\beta_1}$	10:15am	Office 2
...	...	...	...

This approach also provides the same protections against an adversary as the earlier strawman protocol, since knowing $\{101, n_1\}_{k1}, \{\alpha_1\}_{\beta_1}$ does not lead to knowing 101. Furthermore, the adversary cannot link $\{101, n_1\}_{k1}, \{\alpha_1\}_{\beta_1}$ and $\{101, n_3\}_{k1}, \{\alpha_3\}_{\beta_1}$ together since a different random n is used.

When a user queries the database, he will issue a "`Select * from` DB `where` ID=$\hat{\beta}$ and Time=10:00am", where

$$\hat{\beta} = \{101, ID\}_{k2}$$

The database will encrypt the first portion of each ID field with $\hat{\beta}$, and check whether it matches the second portion. If it does, the database will return that entry to the user. For example, encrypting $\{101, n_1\}_{k1}$ with $\hat{\beta}$ will match $\{\alpha_1\}_{\beta_1}$, and hence this entry belongs to the user. Since $\{101, n_1\}_{k1}$ is protected via $k1$ which is only know by the user, the database does not have to verify the user. It is clear that strawman 2 is more efficient than strawman 1.

However, once an adversary observes $\hat{\beta}$, the adversary can determine future time and locations that $\hat{\beta}$ have visited. For instance, using $\hat{\beta}$, the adversary know that the same tag visited Office 2 at time 10:15 am. The reason is that while at time 10:15 am we have $\{101, n_3\}_{k1}$ which uses a different n_3, the same β is used, since $\beta = \{101, ATTR\}_{k2}$. The $ATTR$ is a fixed value in the database and cannot be changed by the

user. Thus, strawman 2 only prevents linkability so long as the adversary *never* observes a user querying the database.

This vulnerability does not occur in [26] because in their encrypted database, all fields are encrypted by the user and stored into the table. In other words, "10:15 am" and "Office 2" are all encrypted separately. The "β" used to encrypted "10:00 am" and "10:15 am" will be $\{10{:}00\text{ am}, Time\}_{k2}$ and $\{10{:}15\text{ am}, Time\}_{k2}$, so knowing the β value of 10:00 am does not reveal anything about 10:15 am. We cannot do the same in an RFID based tracking system because the RFID tag **cannot** determine the time and location independently.

C. Discussion

From the strawman protocols, we can make the following observations. First, a direct encryption of data by the RFID tag (strawman 1) protects user privacy even if the database is compromised. However, this approach has slow query performance since the database cannot return only the user's data to him. Second, the solution cannot merely focus on building an encrypted database, but must also consider the user query process. The protocol has to ensure that the user's query does not reveal useful information that an adversary can use to violate user's privacy (strawman 2). Finally, unlike more powerful laptop devices [31], the RFID tag cannot maintain an internal clock to determine time by itself, not capture IP address or GPS coordinates to determine its location independently. While [32] gives a solution for the RFID tag to determine time, it is unclear that the same techniques can be applied to location information since the tag movements is unpredictable.

The intuition behind our approach is to utilize separate database servers to perform different roles in the RFID tracking system, as well as store different types of data in each server. There are two roles in the RFID tracking system, where the tag was read (location), and at what time the tag was read (time). Our approach uses two database servers, one to control and store time data and the other to control and store location data. **In this paper, we assume that the adversary can only compromise one of the following, (a) the database storing time information, (b) the database storing location information, (c) a small number of RFID readers.**

We justify this assumption because the two servers can be housed at different physical locations, running different operating systems, and managed by a separate group of system administrators. This raises the bar for an adversary to compromise both servers. Also, given the large number of RFID readers deployed, it is reasonable to assume that the adversary cannot control a majority of them.

The following notations are used to describe our protocols. The server controlling the timestamps is the timestamp server TS, and the server controlling the location is the location server LS. We use terms server and database interchangeably in this paper. While both TS and LS can communicate with the RFID readers, only the LS knows the location of these RFID readers. We denote an RFID reader as R, and an RFID tag as T. Each T maintain its own secret s, ID and a simple incremental counter ct. The values of s, ID, and ct are kept secret and only known to the tag owner. The tag also maintains a simple incremental counter ct. We use subscripts i when denoting more than one reader or tag.

We denote the RFID tag identifier as ω, and ω will be stored under the "ID" attribute in the LS. For instance, when there is no security at all, the ω value is just ID, in strawman protocol 1, the ω value will be $\{ID, n\}_k$, and in strawman protocol 2, $\omega = (\{101, n\}_{k1}, \{\alpha\}_\beta)$. A summary of the notation used in this paper is given in Table IV.

TABLE IV
SUMMARY OF NOTATIONS

R	RFID reader
T	Tag
TS	Timestamp Server
LS	Location Server
s	Tag's secret, known only to tag owner
ID	Tag's ID, known only to tag owner
ct	Tag's counter value, known only to tag owner
ω	Tag identifier
t	timestamp, stored in TS
loc	location, stored in LS
n	Nonce generated by tag
N_{ls}	Nonce generated by LS

V. RFID PROTOCOL

The intuition behind our protocol is for the RFID tag to generate two pieces of data each time it responds to an RFID reader. The reader will store one piece of data associated with the time the tag was read in the TS, and the other piece of data associated with the location of the tag in the LS. An adversary compromising either LS or TS will be unable to violate the privacy of the RFID tag. When a legitimate user wishes to query the databases, he will query both TS and LS separately and combine the result to satisfy his query. The overview of an RFID reader collecting data and user querying the servers is shown in Fig. 4.

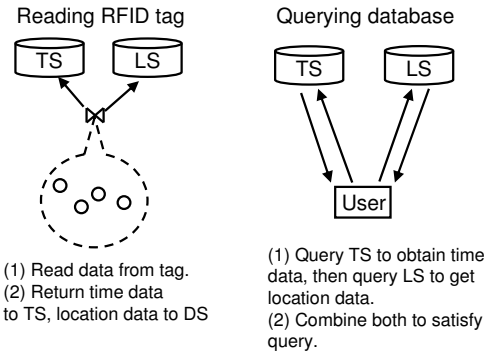

Fig. 4. (L) Obtaining data from tags. (R) Querying databases for data.

A. Collecting data from tag

Fig. 5 shows the protocol for collecting data from an RFID tag. When a reader queries the RFID tag in Step (1), the tag will first generate a random number n by hashing its secret s

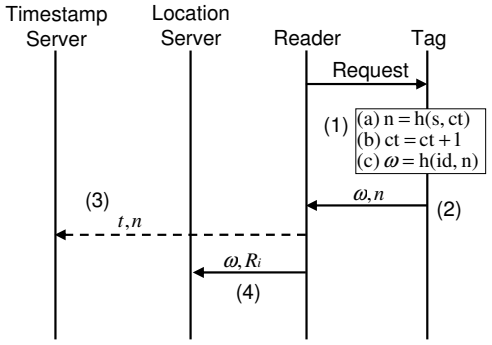

Fig. 5. Reader-tag interaction. The dotted line in step (3) denotes that the reader transmits directly to the timestamp server, bypassing the location server.

with the current counter value, ct. The tag will then increment the ct by one (Step 1b), and create the identifier ω as $h(ID, n)$. Finally, in Step (2), the tag returns ω, n to the reader.

When the reader receives this tuple, the reader will append the current timestamp t together with n, and send that to the TS in Step (3). This messages bypasses the LS completely, meaning the LS never learns n. Then, the reader will append the reader's ID, R_i, and ω, and send everything to the LS in step (4). Using R_i, the LS can determine the reader's.

TABLE V
TABLE MAINTAINED BY TS

	Time	Random value
1.	10:00 am	n_i
2.	10:00 am	n_j
3.	10:15 am	n_k
...	...	...

At the end of the protocol, the TS maintains a table shown in Table V. The TS table has 2 attributes, time, and a random value. The random values associated with time 10:00 am for instance, are all the n values transmitted by all the RFID readers. Each n value represent a different RFID tag. Similarly,

TABLE VI
TABLE MAINTAINED BY LS

	ID	Location
1.	ω_1	Office 1
2.	ω_2	Office 3
3.	ω_3	Office 2
...	...	...

the LS maintains a table shown in Table VI. The LS table has 2 attributes, the tag identifier ω and the location that tag was read. Each entry in the LS table represent a different RFID tag response. Note that the LS does not update the table in real time, and thus the ordering of entries do **not** indicate the time an RFID tag was read.

B. Querying the database

Let the user wanting to learn his location at 10:00 am. He will need to query LS using the ω value $h(101, n)$, where n is the value he choose at 10:00 am. Notice that the LS table is similar to strawman protocol 1's table. In both instances, the ID 101 is protected by a random number n. To query LS, the user must first determine the value of n he used at 10:00 am. In our protocol, we let $n = h(s, ct)$ and increment ct immediately afterwards. While the tag cannot recall the random n value used at 10:00 am, the owner of the tag can determine the *current* value of ct in the tag. (We assume that there is a command to recover this.) Knowing the secret s as well as the initial and current value ct, the user can re-generate all the random n values the RFID tag has used so far. For instance, the user can generate a list The user then

TABLE VII
TABLE MAINTAINED BY USER

ct	n	time
...	...	...
ct_{i-1}	$h(s, ct_{i-1})$	?
ct_i	$h(s, ct_i)$	?
ct_{i+1}	$h(s, ct_{i+1})$	?
...	...	...

queries the TS database using

$$\texttt{Select Time from } TS \texttt{ where Random Value} = n_i, \quad (1)$$

where $n_i = h(s, ct_i)$.

After receiving the time corresponding to n_i from Table V, the user determine whether the returned time is larger than or smaller than his target time of 10:00 am. If the returned time is larger, the user will pick another an smaller ct value, compute n, and query the TS again. If the returned time is smaller, the user will pick a target ct value. Using a simple binary search, the user only needs to execute $O(\log n)$ queries to TS.

With the user now knowing his n corresponding to 10:00 am, he can now query the LS server using the following query.

$$\texttt{Select * from } LS \texttt{ where ID=}\omega, \quad (2)$$

where $\omega = h(101, n)$.

C. Security analysis

We consider the security of our system after the adversary compromises the TS, the LS and the reader R_i. Since we allow anyone to query the databases, the adversary controlling the TS for instance, can also query the LS like a regular user.

Compromise TS server: The adversary succeeds in attacking TS if he can determine which ω in LS belongs to a user, or if the adversary can determine that two ω values belong to the same person. The reason for focusing on ωs is because through ω, the adversary determine the whereabouts of a user.

The adversary controlling TS is able to access all the records such as those in Table V, as well as observe multiple queries (Query 1) made by a user and the corresponding response. The adversary can also query the LS using information from his observations.

We begin by examining what the adversary can learn from controlling TS. For a single RFID tag T_i with secret s_i being queried twice, the n results from $h(s, ct)$ and $h(s, (ct+1))$ will

be different since the ct value is automatically incremented each time the tag responds, as shown in Fig. 5 Step 1(b). We see that the adversary simply knowing the entire Table V cannot determine whether two n values belong to the same user or not. The adversary can only determine which n values belong to the same user after observing a user execute Query 1 multiple times. The reason is that the answer to Query 1 is the n value associated with a particular time. Since only a user knows his own s values, successive Query 1 link the n values to the same user.

Let us assume the adversary after some observation can determine that the times t_i and t_j, and the corresponding n_i and n_j belong to the same user. Now the adversary tries to query LS to try and determine where this user has been. The table maintained by LS only contains a set of ωs and their corresponding locations. There is no indication *what* time each ω was obtained. The adversary cannot determine which ω belongs to the user he is tracking because $\omega = h(ID, n)$, and that the adversary cannot link $\omega = h(ID, n_i)$ and $\omega = h(ID, n_j)$ together without knowing the secret ID.

The property of separating time and location information into the TS and LS respectively defends against leaking information in the more extreme instances where there are few users in the entire tracking system. Consider the tracking system of an office building at night, and we have TS table,

	Time	Random value
1.	2:00 am	n_i
2.	2:15 am	n_j

Assuming there is no one else in the building, the adversary can infer that n_i and n_j belong to the same person. Now, the adversary can attempt to query LS to determine where that person has been. The adversary cannot determine any ωs from n_i and n_j, and can only issue a query "`Select * from` LS `where` ID=*" to retrieve *everything* from LS to try and determine where this tag has been. Since LS does not store time, the adversary cannot filter the LS data to narrow down possible locations the tag has been.

Compromise LS server: Next we consider the adversary controlling the LS server. The adversary will now be able to access all the records such as those in Table VI, as well as observe multiple queries (Query 2) made by a user and the corresponding response. The adversary can also query the TS using information from his observations. The goal of the adversary remains the same.

From controlling LS, the adversary knows the time and location associated with each ω. Given that $\omega = h(ID, n)$, RFID tags with different IDs will have different ω values, and the same tag will also have different ωs at different times since the n values will change due to $n = h(s, ct)$, and the tag's ct values increments each time it replies. Unlike controlling the TS, the adversary observing multiple Query 2 cannot assume they all belong to the same user and link the ωs together. This is because a user does not have to issue Query 2 more than once to obtain an answer.

Let us assume that the adversary knows that ω_i is associated with time t_i. The adversary can query TS doing "`Select * from` TS `where` Time=t_i". However, since there are many tags that respond at each time, all the adversary obtains is a set of n values. The adversary cannot determine which n value corresponds to his ω_i.

Compromise RFID reader: An adversary controlling an RFID reader will be able to read ω, n from a tag (Step (2) in Fig. 5), and is assumed to know the location of the reader it has compromised. We allow the adversary to physically observe a user transmitting a particular ω_i, n_i once, in other words, being able to associate a user's identity to a ω_i, n_i tuple. The adversary succeeds if he is able to use this information to determine additional information regarding that user.

One attack has the adversary trying to use ω_i, n_i in querying TS and LS. The adversary learns from querying TS, since he already knows the time and n_i values. Since the tag will use a different n each time, the adversary cannot determine whether other n values belong to the same user. Similarly, since the n value is constantly changing, the adversary observing LS cannot use ω_i to determine which ωs belong to the same user. Thus, no additional information can be obtained from TS or LS from ω_i, n_i.

Another attack has the adversary after manually determining the identity associated with ω_i, n_i, trying to determine if a future ω_j, n_j belongs to the same user. This is useful if the adversary controls the reader deployed outside an sensitive location like a clinic. Since the adversary cannot always be physically present to determine a user's identity, this attack allows the user to determine if the same user has visited that location again in the future. However, since knowing the n_i and n_j cannot be linked together because the ct value is incremented and hashed with a secret s known only the tag, the resulting ω_i and ω_j cannot be linked together.

Finally, we consider the scenario where the adversary controls multiple RFID readers. Controlling multiple readers does not give the adversary any additional advantage, since a tag does not have to authenticate the RFID reader before transmission. The tag will always generate a different ω, n tuple to any reader that queries it.

D. Protocol discussion

Our protocol uses the counter ct that automatically increments each time the tag is queried. This feature allows an adversary, using his own reader, to query the tag simply to increment the ct value. However, this behavior only degrades the user's performance, and does not help the adversary learn anything about the user. A rational adversary will not launch this type of attack.

Our choice of ω in the protocol is $h(ID, n)$. Given an adversary, a possible alternative is to set ω as $h(ID, t)$. This will have the same properties as $h(ID, n)$ since the time value t will only occur once and never repeat. Using $h(ID, t)$ will also give better query performance, since the user can directly determine the appropriate ω and query the LS, instead of doing a binary search on TS to determine n. The reason we do not use $h(ID, t)$ is that different readers may have a slightly

different clock skew. Thus, honest readers may issue the same t value to the RFID tag, resulting in similar ω values at different locations. This allows that tag to be linked to two locations, thus violating privacy. Our choice of $h(ID, n)$ does not have this problem since the ct will automatically increment after each query, resulting in different ω values each time.

VI. Additional Discussion

Here we consider disruption attacks that do not impact user privacy but can disrupt regular user operations.

A. Detect deleted data

The adversary controlling either TS or LS can decide to delete selected entries from the respective tables. The adversary can do this simply to disrupt the database operations, or to conceal other malicious activities. Consider for instance an adversary planning to steal something from an office. The adversary can attack the TS or LS to avoid storing any tag data collected around that time or location to cover his tracks. Possible witnesses that check the system to verify their locations will determine that they were not actually present at that time.

Recall that when a user wishes to query "`Select * from` LS `where` ID=ω", he will first build a Table VII to determine his target time. To determine whether any data has been deleted, the user can expand Table VII to include enumerate all previous ct values until up to the current ct value in his RFID tag. He then queries the TS until to determine the n value corresponding to each ct. If the adversary has compromised TS and deleted his entry at a particular time, the n value corresponding to that time will be missing. In other words, if the user has some counter value ct_i where there is no n value in Table V matches $h(s + ct_i)$, the user will suspect that his data has been deleted.

If the user can obtain all the n values corresponding to his ct values, he then queries the LS for each ω associated with each n. If the adversary has deleted his entry from LS, the user will not receive any location associated with one of his ωs. Note that the adversary can only select entries to delete based on either time, if controlling TS, or location if controlling LS. The adversary cannot single out a particular tag's information to be deleted since he cannot distinguish between two tags.

The adversary can compromised the RFID reader instead of TS or LS so that the reader does not broadcast any requests. When this happens, no tag data exists in either TS or LS, and the RFID tag will not be triggered increment it's ct value. While a user can infer from "gaps" in the time and location information from TS and LS that a particular RFID reader might be compromised, the user cannot be certain since the gaps may also be caused by environmental conditions or faculty readers. Nonetheless, the occurrence of such gaps will trigger an investigation and detect any compromised readers.

B. Detect tampered data

Instead of deleting the data, the adversary can choose to tamper with the time or location information and launch an attack as follows. Consider an RFID tag at 10:00 am was read outside "Office 1". There will be an entry in the TS

	Time	Random value
1.	10:00 am	n

where $n = h(s + ct)$. The corresponding entry in LS will be

	ID	Location
1.	ω	Office 1

Now let the adversary compromises TS, and changes the time variable from 10:00 am to 11:00 am. The user querying TS with "`Select * from` TS `where` Random Value = n", will receive the answer 11:00 am. The same user now querying LS with $\omega = h(ID, n)$ will believe that he was outside Office 1 at 11:00 am instead of 10:00 am. Since no data was deleted, the user using the technique for detecting missing data above will not find any problems. The adversary that compromises LS can execute the same attack by changing "Office 1" to "Office 2".

The reason this attack is successful is because there is nothing linking the value n to 10:00 am, nor the value ω to "Office 1". We can modify our protocols to let the RFID reader transmit both the t and R_{ID} information when querying a tag. The tag will then compute a new variable ϵ where

$$\epsilon = h(ID, n, t, R_{id}).$$

and return this value to be stored in LS. Thus, the table in LS will become

	ID	Location
1.	ω, ϵ	Office 1

After the user queries for his location from LS, he will compute $\hat{\epsilon}$ by hashing his ID with the n value and time values he received from TS, and the location provided in LS. If $\hat{\epsilon}$ matches ϵ, the user will accept the answer.

We can use a separate mechanism to detect whether an adversary has compromised an RFID reader to an incorrect t or R_{ID} values. When LS receives the data from an RFID reader, it can check whether the contained R_{ID} matches the RFID reader IP address that transmitted the data. A warning will be flagged if there is a discrepancy. The same check can be performed by TS to verify if the reader used an incorrect t value.

The adversary controlling TS can use this ϵ to determine the location of user if he is able to associate an n value with ϵ since he will then be able to search LS for a matching ϵ value. The adversary cannot obtain ϵ directly since this value is never forwarded to TS. The adversary cannot deduce this value from n and t because ϵ requires knowing R_{ID} and ID, both which the adversary does not know.

For the adversary controlling LS, the addition of ϵ can be used to track a user if the adversary can observe the LS table and determine two identical ϵ values. This will imply that the same user visited both locations. However, each ϵ contains a time t from the RFID reader which will never repeat itself, and the adversary controlling LS cannot manipulate the reader

to reuse an older t value. Thus, the adversary cannot link two locations to the same user.

Finally, the adversary controlling the RFID reader may attempt to reuse old t values to track a user. The adversary can program the RFID reader to always use the same time t value, The idea here is to try to get an RFID tag to return a response that has been repeated before. This way, the adversary can determine that that same tag has move pass the reader twice. Here, we let ϵ to contain an always changing value n which is dependent on an incrementing counter ct. Therefore, even if the same same t, R_{ID} and ID values are used, the resulting ϵ will not be the same.

VII. Conclusion

As ubiquitous systems move away from research prototypes to real world deployments, privacy systems that do not rely on trusted servers will become increasing important. We believe that our work is an initial step away from the trusted server model towards more robust alternatives. Our future work considers two extensions. The first is to allow users to delegate data access control to other users, and the second is to explore techniques to improve range query performance.

Acknowledgment

We would like to thank the reviewers for their comments. This project was supported in part by US NSF grants CNS-0721443, CNS-0831904, CAREER Award CNS-0747108.

References

[1] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and localization with rfid technology," in *IEEE International Conference on Robotics and Automation*, 2004.

[2] V. Stanford, "Pervasive computing goes the last hundred feet with rfid systems," *IEEE Pervasive Computing*, 2003.

[3] E. Bertino and R. Sandhu, "Database security-concepts, approaches, and challenges," *IEEE Transactions on Dependable and Secure Computing*, 2005.

[4] C. Ramaswamy, R. Sandhu, R. Ramaswamy, and R. S, "Role-based access control features in commercial database management systems," in *In Proceedings of 21st NIST-NCSC National Information Systems Security Conference*, 1998.

[5] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Hippocratic databases," in *VLDB*, 2002.

[6] "Commerce Department: we lose laptops," 2006, http://arstechnica.com/security/news/2006/09/7809.ars.

[7] "FBI lost 160 laptops in last 44 months," 2007, http://arstechnica.com/old/content/2007/02/8821.ars.

[8] "A Chronology of Data Breaches," 2009, http://www.privacyrights.org/ar/ChronDataBreaches.htm.

[9] D. Anthony, T. Henderson, and D. Kotz, "Privacy in location-aware computing environments," *IEEE Pervasive Computing*, 2007.

[10] S. Lederer, J. I. Hong, X. Jiang, A. K. Dey, J. A. Landay, and J. Mankoff, "Towards everyday privacy for ubiquitous computing," Computer Science Division, University of California, Berkeley, Tech. Rep. UCB-CSD-03-1283, 2003. [Online]. Available: http://www.cs.berkeley.edu/projects/io/publications/privacy-techreport03a.pdf

[11] A. Beresford and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervasive Computing*, 2003.

[12] A. R. Beresford and F. Stajano, "Mix zones: User privacy in location-aware services," in *Pervasive Computing and Communications Workshops (PERCOMW)*, 2004.

[13] U. Hengartner and P. Steenkiste, "Access control to people location information," *ACM Trans. Inf. Syst. Secur.*, 2005.

[14] J. I. Hong and J. A. Landay, "An architecture for privacy-sensitive ubiquitous computing," in *International conference on Mobile systems, applications, and services (MobiSys)*, 2004.

[15] G. Myles, A. Friday, and N. Davies, "Preserving privacy in environments with location-based applications," *IEEE Pervasive Computing*, 2003.

[16] A. Kapadia, T. Henderson, J. J. Fielding, and D. Kotz, "Virtual walls: Protecting digital privacy in pervasive environments," in *Proceedings of the Fifth International Conference on Pervasive Computing (Pervasive)*, 2007.

[17] T. Kriplean, E. Welbourne, N. Khoussainova, V. Rastogi, M. Balazinska, G. Borriello, T. Kohno, and D. Suciu, "Physical access control for captured rfid data," *IEEE Pervasive Computing*, 2007.

[18] V. Rastogi, E. Welbourne, N. Khoussainova, T. Kriplean, M. Balazinska, G. Borriello, T. Kohno, and D. Suciu, "Expressing privacy policies using authorization views," in *Workshop on Ubicomp Privacy, (Ubicomp)*, 2007.

[19] T. Rodden, A. Friday, H. Muller, and A. Dix, "A lightweight approach to managing privacy in location-based services, equator-02-058," University of Nottingham and Lancaster University and University of Bristol, Tech. Rep. CSTR-07-006, 2002.

[20] S. Weis, S. Sarma, R. Rivest, and D. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," in *International Conference on Security in Pervasive Computing*, 2003.

[21] D. Molnar and D. Wagner, "Privacy and Security in Library RFID: Issues, Practices, and Architectures," in *Conference on Computer and Communications Security*, 2004.

[22] K. Ouafi and R. C.-W. Phan, "Privacy of Recent RFID Authentication Protocols," in *4th International Conference on Information Security Practice and Experience – ISPEC 2008*, 2008.

[23] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *IEEE Symposium on Security and Privacy*, 2000.

[24] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *EUROCRYPT*, 2004.

[25] S. Wang, X. Ding, R. H. Deng, and F. Bao, "Private information retrieval using trusted hardware," in *European Symposium On Research In Computer Security (ESORICS)*, 2006.

[26] Z. Yang, S. Zhong, and R. N. Wright, "Privacy-preserving queries on encrypted data," in *European Symposium On Research In Computer Security (ESORICS)*, 2006.

[27] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *USENIX Security Symposium*, 2004.

[28] G. Avoine and P. Oechslin, "A Scalable and Provably Secure Hash Based RFID Protocol," in *International Workshop on Pervasive Computing and Communication Security (PerSec)*, 2005.

[29] C. Castelluccia and G. Avoine, "Noisy Tags: A Pretty Good Key Exchange Protocol for RFID Tags," in *International Conference on Smart Card Research and Advanced Applications (CARDIS)*, 2006.

[30] R. D. Pietro and R. Molva, "Information Confinement, Privacy, and Security in RFID Systems," in *European Symposium On Research In Computer Security (ESORICS)*, 2007.

[31] T. Ristenpart, G. Maganis, A. Krishnamurthy, and T. Kohno, "Privacy-preserving location tracking of lost or stolen devices: cryptographic techniques and replacing trusted third parties with DHTs," in *Usenix Security Symposium*, 2008.

[32] G. Tsudik, "Ya-trap: Yet another trivial rfid authentication protocol," in *PERCOMW*, 2006.