Serverless Search and Authentication Protocols for RFID

Chiu C. Tan, Bo Sheng, and Qun Li
{cct,shengbo,liqun}@cs.wm.edu
Department of Computer Science
College of William and Mary
Williamsburg, VA, 23187

Abstract

With the increasing popularity of RFID applications, different authentication schemes have been proposed to provide security and privacy protection to users. Most recent RFID protocols use a central database to store the RFID tag data. An RFID reader first queries the RFID tag and returns the reply to the database. After authentication, the database returns the tag data to the reader. In this paper, we proposed a more flexible authentication protocol that provides comparable protection without the need for a central database. We also suggest a protocol for secure search for RFID tags. We believe that as RFID applications become widespread, the ability to search for RFID tags will be increasingly useful.

1 Introduction

Radio Frequency Identification (RFID) technology has already been found in a diverse set of applications ranging from inventory management to anti-counterfeiting protection [15]. RFID technology is expected to continue to grow and diffuse into our everyday lives. However, in order to realize the full potential of RFID technology, the security and privacy aspects of a large scale RFID deployment will have to be addressed. This realization has resulted in the growing body of work in RFID security and privacy issues.

Many recent RFID papers [4, 10, 12, 16] have utilized what we term as the "central database" model. There are three players in this model, the RFID reader, the RFID tag, and the central database. The RFID reader queries the RFID tag and then returns the reply to the central database. This reply is usually structured in such a way that (i) only a genuine RFID tag can correctly generate it, (ii) does not leak any information to the reader, and (iii) only the central database is able to interpret the reply. Accomplishing (iii) usually means that the RFID tag returns a different reply each time it is queried. One possible way is to change the tag secret after every successful query [4]. Ensuring that the central database is able to link different replies with the correct tag is an important component in the central database model. After the central database verifies the reader and tag, the central database returns the information of the RFID tag to the reader. All this is possible because the central database has knowledge of all the RFID tag data as well as RFID tag secrets.

A major shortcoming of the central database model is the assumption of a reliable, always available connection between the RFID reader and the central database. This condition will inevitably be violated in a real world setting. Under the central database model, this means that no authentication can be performed without a connection, hence the entire RFID system breaks down. For example, a truck driver may be dispatched to an off-site location to pick up some merchandise tagged with RFID tags. He has with him his PDA which acts as an RFID reader. There is no connection between his off-site location and the central database due to his remote location. Relying on a central database means that the truck driver will be unable to authenticate his goods. Furthermore, having a central database creates a single point of failure, opening up the entire RFID system to denial of service attacks. For RFID technology to flourish, we have to consider the reliability of RFID protocols, in addition to the security and privacy considerations.

A simple alternative, analogous to the central database model, is download the necessary information from the database into the RFID reader. The RFID reader can then continue to access RFID tags as before. However, as mentioned earlier, the central database has to be able to associate the correct RFID tag data with the changing tag replies. For certain protocols, this means that the central database has to be updated after each successful read, so as to keep up with the RFID tag. Under the simple alternative, instead of updating one central database where multiple RFID readers will access, we could potentially have multiple RFID readers to update. This is especially challenging when there are multiple readers capable accessing the group of tags.

Furthermore, having multiple readers increases this risk of an adversary compromising a reader. Since the central database has knowledge of the data as well as secrets of the RFID tags, a compromised reader could yield much damaging information for the adversary. This is not a concern when there is only one backend database. Ideally, we would like to modify the data downloaded from the secure database to the RFID reader such that the damage done by a compromised reader is limited.

Authentication protocols have been widely studied by the research community because RFID privacy and security can be achieved by having good authentication protocols. Authentication ensures that only authorized RFID readers have access to RFID tag data, and that the tag data is indeed accurate. Under the central database model, this is achieved by the central database. The RFID reader only obtains the required information after being authenticated by the central database. Since the database is assumed to be trusted, the reader is confident the information is indeed correct.

Searching for RFID tags is a natural extension of RFID authentication. With authentication, we can authenticate every RFID tag individually until we find the one we are looking for. However, this method is inefficient when there are many RFID tags available. An ideal solution is for an RFID reader to issue a search query, and only RFID tags that meet the query reply.

In this paper, we mainly consider three problems. First, how does a reader know the tag is a valid tag? Second, how does the tag know that the reader is a valid reader? Third, how does a reader search a certain tag?

The first problem is fundamental issue in RFID research, the ability to distinguish a real RFID tag from a fake RFID tag, to differentiate a tag that belongs to one owner from another. The second problem is more recent. This is an important problem because it prevents a malicious or unauthorized reader from accessing the tag information. A malicious reader obtaining this information can violate the privacy of the object the tag is attached to, as well as jeopardizing the security of the RFID system. This is because the adversary can use the tag information to create cloned tags that are identical to the real ones. The third problem is regarding a reader query a group of tags about the existence of a specific tag. This function will become increasingly common as RFID tags become ubiquitous. As we demonstrate later in the paper, implementing a search function for RFID tags may incur additional security and privacy concerns.

1.1 Our Contributions

We have three main contributions in this paper. First, we propose an authentication protocol that provides mutual authentication between the RFID reader and RFID tag *without* the need for a persistent central database. This is a departure from more recent work on RFID security and privacy research.

Second, our schemes consider security for both the RFID reader and the RFID tag. This differs from some of the earlier research which focused on only protecting the reader or only on the tag. This is important since an end user is vulnerable to attacks from both sides. Consider an RFID tag attached to a user's packet of medication. Protecting the tag from unauthorized readers protects the user's privacy. However, there is also a need to protect the reader from reading fraudulent tags. Thus, any security protocol will need to protect both the tag and the reader.

Third, we introduce the problem of secure search of RFID tags and suggest several secure and effective solutions. We believe that the problem of effective search for RFID will become increasing important as RFID deployment increases. Efficient search of RFID tags is difficult because the very act of replying to a query serves as an identifier for the tag. In this paper, we explore the security and privacy concerns and then present several possible solutions.

The rest of the paper is as follows. The next section covers related work in RFID authentication. Section 3 details the criteria we use to evaluate our solution. Section 4 and section 5 contains the authentication protocol and security analysis respectively. Section 6 details the efficient secure search problem and a few possible solutions. Section 7 discusses some shortcomings of not having a centralized database and how to overcome them. Section 8 concludes.

2 Related Work

Due to space limitations, we cannot do justice to the many RFID authentication protocols available. We refer interested readers to the excellent resource maintained by Avoine [1] and recent survey papers [6, 14] for more information. Instead, we try to show general RFID authentication techniques by examining a few protocols in depth. We have chosen RFID protocols using mainly lightweight primitives as listed in [17].

Early work by Weis et al. [18] suggested the use of a backend database to perform RFID authentication. A reader querying the RFID tag will receive a $metaID$. The reader then obtains the real ID of the RFID tag from the database via the $metaID$. The authors recognized that returning the $metaID$, a constant value, poses a privacy threat because it allows an adversary to track a particular RFID tag based on the $metaID$. A randomized hash lock scheme was proposed, where the tag sends its ID as $(r, ID \oplus f_k(r))$ to a reader. Variable r is a random number generated by the tag, k is the tag's secret key and f_k is a pseudorandom function. The reader sends this package back to a secure server which then searches its database for the ID/key pair to return the

ID back to the reader. Since each query to the tag results in a different reply, tag privacy is protected.

Molnar and Wagner [11] pointed out that this scheme does not protect against a replay attack, since an adversary can eavesdrop to learn $(r, ID \oplus f_k(r))$ and then impersonate the tag. They suggested having both the reader and the tag each contribute a random number, r_1 and r_2 respectively. In their scheme, they assume that the reader and tag both share a common secret s. The tag returns $ID \oplus f_s(0, r_1, r_2)$ to the reader. Since the reader knows the shared secret s, he can obtain ID. This protocol works without a central database. However, the protocol does not consider the case when a reader has been compromised. Since the reader shares a secret with a tag, an adversary compromising the reader can learn the secret keys to every tag the reader has access to. The adversary can then make duplicate tags to fool other readers. Our protocol addresses this particular vulnerability.

Dimitriou [4] is a more recent example of a protocol utilizing a secure database. In this protocol, both reader and tag contribute a random number n_r and n_t. The tag returns to the reader $(h(ID_i), n_t, h_{ID_i}(n_t, n_r))$ which the reader gives to the secure server. After authenticating the reader, the secure server calculates ID_{i+1} and then returns $h_{ID_{i+1}}(n_t, n_r)$ to the reader. The reader sends this back to a tag. The tag will determine ID_{i+1} on its own and verify the $h_{ID_{i+1}}(n_t, n_r)$ sent by the reader. If the value match, then the tag knows that the reader has been authenticated by the server, and changes its secret to ID_{i+1}. Otherwise, the tag keeps ID_i. Similar protocols [10, 12] also adopt the idea of changing the tag secret after every reader query.

One key feature of Dimitriou's protocol is how it attempts to prevent tag and server from being out of sync. An adversary will not be authenticated by the server, thus the server maintains ID_i. The adversary cannot derive $h_{ID_{i+1}}(n_t, n_r)$, so tag maintains ID_i. However, this protocol relies on the reader having a persistent connection to the secure server. As we pointed out in the introduction, this requirement introduces significant drawbacks.

An alternative approach is to have authentication via challenge and response. One version was suggested by Juels and Weis [9]. They observed that since RFID tags have limited resources like humans, ideas from human authentication can be applied to RFID authentication. They introduced HB and HB+ protocols. The HB protocol has a reader issue a new challenge each time it queries a tag. The tag computes the binary inner product and returns the answer to the reader. The reader verifies the answer is correct before accepting the tag as legitimate. Noise can introduced such that the tag will sometimes return a wrong answer. HB+ protocol introduces an additional binding factor from the tag to defend against an active adversary. Later work by [13, 5, 2] improves on this idea.

YA-TRAP [16] introduces a novel idea of using timestamps in RFID authentication. This is a novel approach because RFID tags have no self-contained power source to keep track of time. In YA-TRAP, a reader will send a timestamp of the current time to a tag which then decides whether to return a random reply or an encrypted reply based on the received timestamp and its own internal timestamp. The reader sends this reply back to a backend server to obtain the tag data. [3] suggested an improvement over this protocol.

3 Evaluation Criteria

In this paper, we consider both authentication and secure search for RFID tags. In both cases, our solutions eliminates the need for a central database. The solutions are based on RFID tags being able to perform simple primitives like hash functions, comparison and appending bits together. These are the most common primitives adopted in RFID security research. We evaluate authentication and search protocols based on the following three criteria.

First, our solution has to protect privacy. RFID tags could be attached to private objects like bottles of medication.Protecting privacy means that the RFID tag data should only be obtained by an authenticated RFID reader, and that an unauthorized reader should not be able to identify a particular RFID tag even after repeated queries.

Second, our solution has to provide security. This means that our scheme has to be clone resistant. An adversary cannot easily create large number of fake RFID tags to fool legitimate readers. This also means that our protocols have to be resistant to eavesdropping attacks. An adversary being able to observe *all* communications between the reader and the tag should still be unable to impersonate a real RFID tag or a real RFID reader. Finally, providing security means that the effects of physical attacks on either the RFID reader or RFID tag should be limited.

Third, our solutions has to provide reliability. This means that under our protocols, a legitimate RFID reader can continue to authenticate a legitimate RFID tag regardless of what happens to the central server. For RFID authentication, the objectives are for an RFID reader to obtain the data stored in an RFID tag he is authorized to access, and the RFID tag should only release its data to authorized RFID readers. In this paper, we consider this data to be the unique identity of the RFID tag.

4 RFID Authentication

We present two different authentication protocols. The first version performs challenge and response before sending the tag secret to the reader. The second version sends the tag secret in such a way that only an authenticated reader

can decrypt. For RFID authentication, the objectives are for an RFID reader to obtain the data stored in an RFID tag he is authorized to access, and the RFID tag should only release its data to authorized RFID readers. In this paper, we consider this data to be the unique identity of the RFID tag. First let us describe the notation used in the paper.

We consider an RFID reader denoted as R. Each R has a unique identifier r and an access list, L. We will describe the contents of L a little later. R obtains r and L from a certificate authority, CA, after authenticating itself. The CA is a trusted party responsible for deploying all the RFID tags and authorizing any RFID reader. We assume that communications between R and CA are performed using some secure channel.

Each RFID tag, T, contains a unique value id, a unique secret t, knowledge of a function $f(.)$ and a hash function $h(.)$. The id is a unique identifier of T. It is known by the particular T, the CA, and readers authorized by CA to read that particular T. The secret t is only known by the particular RFID tag and the CA. The function $f()$ takes in two arguments and returns the hash of the arguments concatenated together. This is a one-way hash. T will use its secret t as one of the arguments, and the other argument is supplied by the RFID reader. So if the RFID reader sends an argument r, T will have $f(r,t) = h(r||t)$ where $||$ denotes concatenate. The hash function $h(.)$ is also a one-way hash. We assume that the result of $h(.)$ is of length l, and the CA predefines a length $m < l$ that is known by all tags.

Subscripts are used to describe a particular R or T and their respective variables. Thus a particular RFID reader i will be R_i, with a identifier r_i and access list L_i. A tag j is T_j and has a secret t_j. The access list L contains information about the RFID tags which a particular R has access to. L has a list of all $f(r,t)$ that CA has authorized R to access. So reader i, R_i authorized to access tags $T_1 \cdots T_n$ will have L_i where

$$L_i = \begin{cases} f(r_i,t_1) & : \quad id_1 \\ \cdots & : \quad \cdots \\ f(r_i,t_n) & : \quad id_n \end{cases}$$

Note that R_i does not know any of the tags secret t. It only knows the outcome of the function $f(r,t)$.

We assume that the CA cannot be compromised, and that all readers once authenticated by the CA are trusted. They will not reveal their L to anyone else. We denote an adversary as α. The goals and capabilities of α are discussed in the next section of the paper. The authentication protocols are as follows.

4.1 Authentication Protocol 1

$$R_i \rightarrow T_j \quad : \quad request \tag{1}$$

$$R_i \leftarrow T_j \quad : \quad n_j \tag{2}$$

$$R_i \rightarrow T_j \quad : \quad r_i, n_i \tag{3}$$

$$R_i \leftarrow T_j \quad : \quad ([h(f(r_i,t_j)||n_i||n_j)]_b, \; ques_r^1, \cdots, ques_r^k) \tag{4}$$

$$R_i \quad : \quad \text{Checks if } [h(f(r_i,t_j)||n_i||n_j)]_b \text{ is in } L_i \tag{5}$$

$$\text{If exists and } k \leq \tfrac{l-b}{2} \tag{6}$$

$$R_i \rightarrow T_j : (ans_r, ques_t^1, \cdots, ques_t^k) \tag{7}$$

$$\text{Else} \tag{8}$$

$$R_i \rightarrow T_j : (rand, ques_t^1, \cdots, ques_t^k) \tag{9}$$

$$T_j \quad : \quad \text{Checks if } ans_r \text{ is correct} \tag{10}$$

$$\text{If correct and } \forall x, y, ques_r^x \neq ques_t^y \tag{11}$$

$$R_i \leftarrow T_j : (ans_t) \tag{12}$$

$$\text{Else} \tag{13}$$

$$R_i \leftarrow T_j : (rand) \tag{14}$$

where n_i and n_j are random numbers generated by R_i and T_j respectively. $[h(f(r_i,t_j)||n_i||n_j)]_b$ is the first b bits of the hash of the concatenation of $f(r_i,t_j)$ and n_i and n_j. $ques_r^1, \cdots, ques_r^k$ are the k randomly generated positions from the last $l-b$ bits of $h(f(r_i,t_j)||n_i||n_j)$. This is the challenge to the reader. ans_r is the actual bits in positions $ques_r^1, \cdots, ques_r^k$. This is response of the reader. Similarly, $y_1, \cdots, y_k$ is the list of random positions of the last $l-b$ bits, and ans_t is the bits in those positions. This is the challenge to the tag and response of the tag. In both instances, $k \leq \frac{l-m}{2}$. $rand$ is a random bit string of length k.

The intuition here is to have both the R_i and T_j issue a challenge that only legitimate party is able to answer. Both R_i and T_j pick k positions from the last $l-b$ bits and challenge the other to reply with the correct k bit string. An adversary, impersonating either R_i (T_j), can only produce the correct ans_r (ans_t) with limited probability.

$$\text{Prob(Adversary correctly answers challenge)} = (\frac{1}{2})^k$$

For every new query, R_i and T_j exchange random numbers and R_i sends his identifier r_i to T_j by XORing with n_j. The purpose of this is explained in the next section. R_i then receives the first b bits of $h(f(r_i,t_j)||n_i||n_j)$. Using this first b bits, R_i consults his L_i to determine if there are any partial matches. Note that R_i has to hash the concatenation of each entry in L_i with n_i and n_j. The probability of having another tag with the same b first bits is just

$$\text{Prob(Another tag sharing first } b \text{ bits)} = (\frac{1}{2})^b$$

If there are no matches, R_i knows that T_j is not an RFID tag AC has authorized him to access. R_i generates a random k bit reply ans_r and his challenge $y_1, \cdots, y_k$ to T_j.

Otherwise, R_i returns the bit values in positions $x_1, \ldots, x_k$ as ans_r and his challenge. If R_i has several entries in his L_i with the same b bits, R_i simply performs the challenge and response several times, each time replying with a different entry. If all the entries do not match, then R_i concludes that T_j is not a tag that he is supposed to access.

When T_j receives ans_r, it checks if the bit values match the positions of $h(f(r_i, t_j)||n_i||n_j)$ generated. A correct ans_r indicates that R_i is an authorized reader, since only an authorized reader can obtain the correct $f(r_i, t_j)$ value from AC, thus knowing the correct sequence to $h(f(r_i, t_j)||n_i||n_j)$. T_j replies R_i's challenge in ans_t correctly if ans_r is correct and that there is no $ques_t$ is the same as $ques_r$. Otherwise T_j returns a random answer. R_i uses ans_t to determine whether T_j is a legitimate tag. After one round of the authentication step, the probability of R_i correctly identifying T_j is

$$\text{Prob(Accurate identification)} = 1 - (\frac{1}{2})^{l-b-k}$$

This is due to the k bits challenge $ques_k^1, \cdots, ques_k^k$ posed by T_j. R_i can always query T_j more than once, picking different k bit challenge each time. A correct T_j will always be able to return the right answer.

4.2 Authentication Protocol 2

$$
\begin{aligned}
R_i \rightarrow T_j &: request && (1)\\
R_i \leftarrow T_j &: n_j && (2)\\
R_i \rightarrow T_j &: n_i, r_i && (3)\\
R_i \leftarrow T_j &: h(f(r_i, t_j))_m, h(f(r_i, t_j)||n_i||n_j) \oplus id_j && (4)\\
R_i &: \text{Checks } L_i \text{ for matching } h(f(r_i, t_j))_m && (5)\\
R_i &: \text{Determines } h(f(r_i, t_j)||n_i||n_j) \text{ to obtain } id_j && (6)
\end{aligned}
$$

where n_i, n_j are random numbers generated by R_i and T_j respectively. T_j sends its id_j as $h(f(r_i, t_j)||n_i||n_j) \oplus id_j$. This is used to protect id_j. The tag also sends $h(f(r_i, t_j))_m$ to help R_i to reduce the time take to search through L_i. An unauthenticated reader cannot obtain id_j since he does not know $f(r_i, t_j)$, and hence the $h(.)$ value. This is a form of tag authenticating reader, since the value of the tag is incomprehensible to an unauthorized reader.

The reader checks his L_i for matching entries that have the same first m bits as $h(f(r_i, t_j))_m$. R_i can precompute the $h(f(r_i, t_*))_m$ for every entry in L_i, and then organizes the result into corresponding groups. If there are no entries in L_i that match the first m bits, then either the RFID tag is a fake, since it is not able to generated a correct $f(r_i, t_j)$, or that it is a tag that R_i is not authorized to access, thus not appearing in L_i. If there is a match, the reader then uses the random numbers n_i and n_j to obtain $h(f(r_i, t_j)||n_i||n_j)$ and the resulting id_j. If the id_j received from the tag does not match entry in L_i then again it either means that the RFID tag is a fake or R_i is not authorized to access the tag. Note here that a different random n_j and n_i are used in each transaction, which means that the shared secret between R_i and T_j used to protect id_j, $h(f(r_i, t_j)||n_i||n_j)$, changes each time. Also, since our hash $h(\cdot)$ is a one way hash function, even knowing the entire $h(f(r_i, t_j))$ does not reveal $f(r_i, t_j)$. We discuss the tracking implications of this in the next section.

To determine the value of m, we first define a *collision space* as $CS = 2^{l-m}$. This is the number of RFID tags whose hashed value share the same first m bits. We define β as the probability that, given a tag, the probability that when a reader reads in another tag having the same first m bits, the two tags are the same. There more privacy we wish, the smaller we set β. Thus, we have

$$\frac{\binom{N}{1}}{N^2} = \frac{1}{N} = 2^{m-l} \leq \beta \Rightarrow m \leq l + \log \beta.$$

The search time for R_i becomes $O(\frac{L}{2^m})$ since R_i can organize L_i into respective groups, when T_j returns the first m bits of $h(f(r_i, t_j))_m$, In this way, R_i does not need to search the entire L_i, but only the smaller group of size $\frac{L}{2^m}$.

5 Security Analysis

We now consider how our protocol performs against different RFID attacks. As mentioned earlier, we denote the adversary as α, a legitimate reader as R_i and a legitimate tag as T_j. We denote a fake tag j impersonating the real tag j as $\hat{T_j}$. For each attack, we first describe the nature of the attack, the common assumptions made, and the capabilities of α. We then demonstrate how our protocol defends against that attack.

Privacy: The privacy attacks occur when α wishes to learn of the contents of T_j. This is important when, for example, T_j is attached to a valuable cargo in a warehouse. α can query every tag in the warehouse to decide the most valuable one to steal. α is generally assumed to have a list of targeted RFID tag data. He then queries the tags to see which tag matches his list. In our protocol, each time any reader queries T_j, T_j generates a new response $h(f(r, t)||n_r||n_t)$ for authentication. Thus α cannot identify which RFID tag is on his list. This protects the privacy of the tag.

Tracking: This attack is another form of the privacy attack. Here, α tries to track T_j over time. α succeeds if he is able to distinguish T_j from other RFID tags over time. For example, T_j could be attached to a jacket. Since α being

able to identify T_j over time, α is able to track T_j. This attack is usually performed by α repeatedly querying T_j with a value that will yield consistent reply. This consistent reply becomes a signature of T_j.

In our scheme, α can reuse the same n_α and r_α, but cannot predict the random n_j generated each time by T_j. If we change the authentication protocol step (3) to return just $h(f(r_i, t_j)||n_i||n_j) \oplus id_j$, then tracking is impossible. This is because the returned $h(f(r_\alpha, t_j)||n_\alpha||n_t) \oplus id_j$ changes each time a reader, valid or not, queries it. The penalty of this approach is that performance of R_i will be poor if L_i is very large since R_i will have to check every entry. If T_j returns the first m bits of $h(f(r_i, t_j))_m$, the search time for R_i better since R_i can organize L_i into respective groups. The tradeoff is that now α can *potentially* track T_j by repeatedly querying for the same $h(f(r_i, t_j))_m$ reply. Notice that if m is small, then the tracking is not definite since there could be multiple RFID tags that share the first m bits. This is controlled via the system parameter β.

Cloning: This is a form of providing RFID security. We consider the "skimming" attack described by Juels [7]. In this attack, α will usually first query T_j and obtains a response. α then places the response on a fake RFID tag, $\hat{T_j}$. By creating fake RFID tags that contain the responses of real RFID tags, α hopes to pass off his counterfeits as legitimate. α succeeds if R_i believes that $\hat{T_j}$ is T_j.

Under our protocol, T_j will return a different hash based on the random n_i and r_i provided by R_i. Since α does not know R_i's identifier r_i, and cannot predict the random n_i generated each time by R_i, the hash value that α obtains from T_j will not be the same as the value R_i obtains when he queries T_j. Thus α cannot create a $\hat{T_j}$ that can fool R_i. We consider the case when α can listen in on the transaction between R_i and T_j later.

Eavesdropping: Here α is able to observe *all* interactions between R_i and T_j. In other words, α can learn of r_i, n_i, n_j. α will know $h(f(r_i, t_j)||n_i||n_j) \oplus id_j$ and $h(f(r_i, t_j))_m$. α's goal is to use the data to impersonate a fake reader $\hat{R_i}$ or a fake tag $\hat{T_j}$. [1]

α will learn of R_i's identifier r_i in step (3). However, the hash function $h(f(r_i, t_j)||n_i||n_j)$ requires random numbers generated by two parties, the reader and the tag. α impersonating a R_i or T_j cannot control the random number generated by the other party. Thus even knowing R_i's r_i is useless. α needs to know the value $f(r_i, t_j)$ to impersonate either R_i or T_j. Since $f(r_i, t_j)$ is never passed directly, α cannot determine $f(r_i, t_j)$, unless α can determine the tag secret t_j.

Since every transaction between R_i and T_j involves both parties generating a new n_i and n_j, α cannot launch a replay attack using the previous values.

Physical attack: We consider two different flavors of physical attack. The first is when α compromises the reader R_i. The second is when α compromises the tag T_j. In both cases, we assume that once α has physically compromised R_i and T_j, α will learn everything about R_i and T_j. We do not consider hardware-based defenses against physical attacks in this paper.

First, we consider α compromising R_i. α will know the contents of L_i, as well as r_i. α will therefore be able to impersonate R_i and obtain data from tags $T_1, \cdots, T_n$. We want to prevent α from using the knowledge to create counterfeit tags. Let T_j be in L_i, and α wishes to create a counterfeit tag be $\hat{T_j}$ that can fool another authenticated RFID reader T_x.

α knows $f(r_i, t_j)$ and id_j from L_i. To create $\hat{T_j}$ to fool T_x, α has to be able to derive $f(r_x, t_j)$. This is because each $f(.)$ value in the access list is different for every RFID reader. R_i will have $f(r_i, t_j)$, and R_x will have $f(r_x, t_j)$. Thus α cannot substitute his $f(r_i, t_j)$ and id_j into $\hat{T_j}$. Since $f(.)$ is irreversible, α is unable to derive t_j from $f(r_i, t_j)$.

Second, we consider α compromising tag T_j. α will now be able to create a fake $\hat{t_j}$ that can fool the honest R_i. We want to prevent α from creating another tag that can fool α. We let this other tag be T_x, and assume that T_x is inside L_i.

Since α has compromised T_j, we assume that α knows any information that R_i passes to T_j. To create T_x to fool R_i, α has to able to generate the correct $f(r_i, t_x)$. However, each RFID tag has a unique secret t. Thus α knowing t_j cannot derive t_x. Therefore, α cannot create a fake T_x to fool R_i.

Denial of service (DoS): α here does not try to obtain information, but rather tries to ensure that a legitimate R_i cannot access data stored in T_j. This is a potential problem in RFID authentication protocols that use a trusted server.

A typical way of using a trusted server is for T_j to return some value to R_i which R_i has to redirect to a central database. For protocols that do not require RFID tag and central database to remain in sync, conventional DoS attacks to overwhelm the central database can be launched, resulting the R_i being unable to authenticate T_j.

For protocols that require some synchronization between central database and tag, a common defense against DoS attacks is to require T_j to change its value only after receiving some confirmation generated by the database [10, 4]. Thus R_i has to perform an addition interaction with T_j after receiving the data. Since R_i is an authorized reader, the central database expects R_i to do the "right thing" by transmitting the confirmation to T_j. The value stored inside the central database changes after R_i queries it. α can desynchronize the process by repeatedly querying T_j, preventing R_i from passing the confirmation to T_j. This way, if an-

[1] Note that this version of eavesdropping attack is stronger since it assumes that α can eavesdrop on both the forward as well as backward channel. A weaker version of eavesdropping assumes α cannot listen in on the tag-to-reader channel.

other reader R_x were to query T_j before R_i manages to update T_j, R_x will obtain the out-of-data reply from T_j.

Our protocol removes the need for a central database. Once a reader is able to authenticate himself, there is no need for further interaction in order to authenticate a tag.

6 RFID Search

As mentioned earlier, searching RFID tags is performed when there are a group of tags and a reader wishes to determine if a particular tag is in this group. A straightforward solution is to perform our authentication protocol above for every tag. Existing probabilistic RFID anti-collision algorithms can be used to distinguish one tag from another. However, this approach is inefficient when there are many tags. Instead, an ideal solution is for the the reader to query for a specific tag, and only that particular tag will reply. For RFID search, the goal is for an authorized RFID reader to search for a particular RFID tag (which he is authorized to access) among a group of RFID tags. Only the tag which matches the search can reply. At the same time, the RFID tag can only release its data to an authorized RFID reader.

An intuitive way of doing so is as follows. We have R_i wishing to find the tag T_j. R_i broadcasts his request for id_j. We let $T*$ refers to an arbitrary tag. $ownid$ refers to the id for each individual tag.

$$\begin{aligned} R_i \rightarrow T* &: \text{Broadcast } id_j && (1)\\ T* &: \text{If } ownid = id_j && (2)\\ R_i \leftarrow T_j &: \quad \text{reply} && (3) \end{aligned}$$

The simple protocol does not provide any security. An adversary can simply query a group of tags to find out if a particular valuable tag is present. One solution to this problem is for the tag to authenticate the reader before replying. This means that when R_i broadcasts his query, every tag, not only those that satisfy the query, needs to authenticate R_i before replying. If only those tags matching R_i's request initiate the authentication, the adversary will know that *that* tag satisfies the request. However, having every tag to authenticate the reader is similar to having the reader authenticate every tag individually.

Another issue is that the reader wants his request to be received by authorized tags only. The reason is that if an adversary can learn of the query, he could simply observe the channel to determine if there are any replies. Even if the reply is encrypted, the adversary will know that the tag does exists.

Therefore, we can characterize our problem as follows. Tags should only respond to authenticated readers. Readers should only query authenticated tags. This creates a chicken-and-egg problem since **readers want to query authenticated tags, but tags will only respond to authenticated readers.**

Our solution to this problem is for the reader to issue a query such that only an authenticated tag can understand, and for the tag to reply in such a manner that only an authenticated reader can understand. An adversary can still observe all the transactions, in that he can observe there has been a query and an answer. However, since the adversary does not know the contents of the query, observing the existence of an answer is not useful. Our secure search protocol is as follows. We use the same notation as before. The new notation $ownt$ refers to the tag secret t for a tag.

$$\begin{aligned} R_i \rightarrow T* &: \text{Broadcast } h(f(r_i,t_j)||n_r) \oplus id_j, \\ &\quad n_r, r_i && (1)\\ T* &: \text{Find } id_j \text{ by deriving} \\ &\quad h(f(r_i, ownt)||n_r) && (2)\\ &: \text{If } ownid = id_j && (3)\\ R_i \leftarrow T_j &: \quad h(f(r_i,t_j)||n_t||n_r) \oplus id_j, n_t && (4) \end{aligned}$$

Notice here that the broadcast request, $h(f(r_i,t_j)||n_r) \oplus id_j$ can only be read by an authenticated tag, since to learn id_j, the tag will need to be able to execute step (2), which implies knowledge of tag secret t_j. An α will not be able to obtain id_j since he does not know t_j. The tag reply, $h(f(r_i,t_j)||n_t) \oplus id_j, n_t$, can only be read by an authenticated reader since it requires knowledge of $f(r_i,t_j)$ which is known only by the authenticated reader.

6.1 Security Analysis

The security analysis of the previous protocol applies to the search protocol with one exception. The search protocol above is not resistant to tracking.

Consider the following attack. α eavesdrops on the transaction between a reader and a group of tags. α is unable to decrypt the query or the reply, but can detect the presence of a query and reply. α than uses the same query individually on each and every tag in the group to determine *what* that query is for. Since the query is legitimate, the tag with the corresponding value will reply. Even though the reply is different each time, due to the random n_t generated by the tag, it remains true that only one tag will reply, since each individual tag has a unique secret t, thus a unique $f(r,t)$.

Now, α can replay the query again and again to identify a particular tag. Note here that the tracking attack here is slightly different from tracking attacks commonly found in RFID security literature. The adversary cannot pick a particular tag to track. Rather, he can only track a tag which has been searched for by a legitimate reader. Furthermore, the

adversary has to iteratively query every tag in a group individually before determining what tag he is tracking. These difficulties increases the difficulty of launching a tracking attack via searching protocol.

This underscores a fundamental difficulty in developing a secure search protocol for RFID tags. **The very act of replying of a query can be used to identify a tag.** So long as a search query expects a unique reply, the reply becomes an identifier for a particular tag. Encryption alone does not solve the problem, since encryption only prevents an adversary from learning the contents of a message, but not that a message has been sent.

6.2 Search Protocol Improvements

Here we suggest several methods to the search protocol to minimize the impact of a tracking attacking due to the secure search protocol. We stress again that tracking a tag using from the search protocol considerably more difficult than the tracking attack commonly found in RFID security literature.

One solution is to have each tag store the last random number used by a legitimate reader that it answers. If the same random number is used again, the tag will not reply. The improved protocol is as follows.

$$
\begin{aligned}
R_i \rightarrow T* &: \text{Broadcast } h(f(r_i,t_j)||n_r) \oplus id_j, \\
&\quad n_r, r_i && (1)\\
T* &: \text{Find } id_j \text{ by deriving} \\
&\quad h(f(r_i, ownt)||n_r) && (2)\\
&: \text{If } ownid = id_j \text{ and } n_r \neq oldn && (3)\\
R_i \leftarrow T_j &: \quad h(f(r_i,t_j)||n_t) \oplus id_j, n_t && (4)
\end{aligned}
$$

Where $oldn$ is the previous random number used. Now, an adversary cannot replay $h(f(r_i,t_j)||n_r) \oplus id_j, n_r, r_i$ to get a reply, since n_r was just used. The adversary does not know $f(r_i,t_j)$, thus cannot generate his own legitimate query that will be answered by the tag. The adversary can observe the second time R_i does a search query to obtain a different random number, n_r'. Now, he can try to use the previous search query. However, since adversary cannot determine the contents of the query, he cannot know if R_i was querying for the same tag or not. Assuming that the adversary cannot determine *what* R_i is looking for, he cannot track any tag based on two reader queries. In general, an adversary will need at least 1 more successful query than the number of tags to be always successfully track 1 tag. Using the pigeonhole principal, with n tags each capable of storing the last m random numbers of successful reader query, an adversary can only guarantee to be able to track 1 tag after $n * m + 1$ queries.

However, the above method does not work as effectively against an opportunistic adversary who simply replays the overheard queries over and over again to find at least 1 tag to track.

Another solution is to adopt a challenge and response method. The idea is to avoid the condition where replying to a query can be used to identify a tag. We use $[id_j]_m$ to denote the first m bits of id_j and $ownid_m$ to denote the first m bits of $ownid$. The protocol is as follows.

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

In this protocol, any tag that matches the first m bits of id_j will reply to the query. Depending on the length of m, there could be multiple tags that share the same first m bits. R_i can use existing collusion avoidance techniques to obtain id_j. Since multiple tags may share the same m bits, α cannot infer any unique information from the reply. A tag's response is protected by the XORing their value with $h(f(r_i,t_j)||n_r||n_t)$. Only an authenticated reader will know $f(r_i,t_j)$, and be able generate the correct hash value. Furthermore, each party contributes a random number n_r and n_t that makes up the final hash value needed to successfully obtain the id_j. This prevents an adversary from launching a replay attack from using either the query or reply.

This solution does not work well when the id's in each tag are structured. For example, the first several bits of an id could signify general product code, the next several bits the tag origin and so on. In this scenario, the adversary can obtain some information simply by observing $[id_j]_m$. Note that $[id_j]_m$ cannot be XORed with some $f(r_i,t_j)$ since then only T_j decipher the request.

The last solution is to use noise to mask the reply. Each tag that receives a search query that does not match the request will have some probability of replying. Thus,

$$
\begin{aligned}
R_i \rightarrow T* &: \text{Broadcast } h(f(r_i,t_j)||n_r) \oplus id_j, \\
&\quad n_r, r_i && (1)\\
T* &: \text{Find } id_j \text{ by deriving} \\
&\quad h(f(r_i, ownt)||n_r) && (2)\\
&: \text{If } ownid = id_j && (3)\\
R_i \leftarrow T_j &: \quad h(f(r_i,t_j)||n_t) \oplus id_j, n_t && (4)\\
&: \text{Else} && (5)\\
R_i \leftarrow T_j &: \quad \text{Return } (rand, n_t) \\
&\quad \text{with probability } \lambda && (6)
\end{aligned}
$$

where λ is the predefined probability that a tag that does not match id_j will reply. Here, an adversary cannot depend

on replaying a previous query to track a tag since any tag could reply. This method avoids leaking information to an adversary. To estimate λ, we first let S be the number of RFID tags that can hear a single broadcast query. We want to have a probability of γ that at least one tag that is not the answer replies to generate noise. Thus we estimate λ by solving $1 - (1 - \lambda)^S \geq \gamma$. The additional work done by reader to filter out the noise is $O(\lambda \cdot S)$.

7 Additional Discussion

Despite the shortcomings of the central database model, it does have two advantages. The first is the ease of performing revocation, and the second is fine grain access control.

The central database model provides an implicit revocation capability. Since the RFID reader has to contact the central database each time to obtain the tag data, the central database can perform revocation simply by ignoring the reader.

Under our scheme, simple revocation can be accomplished by replacing the existing RFID tag with a new tag containing a new secret t when necessary. This solution is practical when the objects attached with RFID tags change owners frequently. Different owners will want to attach their own RFID tags to their objects to better interface with their existing RFID management applications. An alternative revocation scheme is to retain the RFID tags, but allow the RFID tag's secret t to be changed by trusted parties. A special secret pin can be built into each RFID, and knowledge of the pin will allow the reader to change the tag secret. This pin can be transmitted directly to trusted agents of the CA, or encoded via a different channel like a 2-D barcode next to the RFID tag [7, 8]. In this way, the CA can enforce a time period in which authorized readers can access the tag data.

The other implicit advantage of the central database model is fine grain access control. When the central database returns the tag data to the reader, it can choose to only return part of the information depending on the permissions of the reader.

We accommodate fine grain access control in our scheme by replacing the single secret t in each RFID tag with multiple secrets depending on the granularity. For example, an RFID tag whose data consists of a general produce code and unique identifier will have two secrets t^1, t^2. A reader with access to the general product code will only receive $f(r, t^1)$ in his L while another reader with access to the unique identifier will receive $f(r, f^2)$ as well. We can simply extend the number of secrets per tag to as fine a level of access control as desired.

8 Conclusions

In this paper, we present an authentication protocol and a search protocol for RFID tags. Our authentication protocol provides both tag-to-reader and reader-to-tag authentication that is resistant against common RFID attacks. A major difference from the previous scheme is that our scheme provides similar level of protection without the need of a persistent central database. In this paper, we also introduce a new problem of performing secure search for RFID tags. We detail the difficulties in secure search, and provide several secure search protocols. Finally, we also address the implicit advantages having a secure central database and suggested solutions for overcoming them.

Acknowledgment

The authors would like to thank all the reviewers for their helpful comments. This project was supported by US National Science Foundation award CCF-0514985.

References

[1] G. Avoine. http://lasecwww.epfl.ch/∼gavoine/rfid/.

[2] J. Bringer, H. Chabanne, and D. Emmanuelle. HB^{++}: a lightweight authentication protocol secure against some attacks. In *IEEE International Conference on Pervasive Services, Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing – SecPerU 2006*, Lyon, France, June 2006. IEEE, IEEE Computer Society Press.

[3] C. Chatmon, T. van Le, and M. Burmester. Secure anonymous RFID authentication protocols. Technical Report TR-060112, Florida State University, Department of Computer Science, Tallahassee, Florida, USA, 2006.

[4] T. Dimitriou. A lightweight RFID protocol to protect against traceability and cloning attacks. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm*, Athens, Greece, September 2005. IEEE.

[5] H. Gilbert, M. Robshaw, and H. Sibert. An active attack against HB$^+$ – a provably secure lightweight authentication protocol. Manuscript, July 2005.

[6] A. Juels. RFID security and privacy: A research survey. Manuscript, September 2005.

[7] A. Juels. Strengthening epc tags against cloning. In *WiSe '05: Proceedings of the 4th ACM workshop on Wireless security*, pages 67–76, New York, NY, USA, 2005. ACM Press.

[8] A. Juels and R. Pappu. Squealing euros: Privacy protection in RFID-enabled banknotes. In R. N. Wright, editor, *Financial Cryptography – FC'03*, volume 2742 of *Lecture Notes in Computer Science*, pages 103–121, Le Gosier, Guadeloupe, French West Indies, January 2003. IFCA, Springer-Verlag.

[9] A. Juels and S. Weis. Authenticating pervasive devices with human protocols. In V. Shoup, editor, *Advances in Cryptology – CRYPTO'05*, volume 3126 of *Lecture Notes in Computer Science*, pages 293–308, Santa Barbara, California, USA, August 2005. IACR, Springer-Verlag.

[10] S.-M. Lee, Y. J. Hwang, D. H. Lee, and J. I. L. Lim. Efficient authentication for low-cost RFID systems. In O. Gervasi, M. Gavrilova, V. Kumar, A. Laganaà, H. P. Lee, Y. Mun, D. Taniar, and C. J. K. Tan, editors, *International Conference on Computational Science and its Applications - ICCSA 2005, Proceedings, Part I*, volume 3480 of *Lecture Notes in Computer Science*, pages 619–627, Singapore, May 2005. Springer-Verlag.

[11] D. Molnar and D. Wagner. Privacy and security in library RFID: Issues, practices, and architectures. In B. Pfitzmann and P. Liu, editors, *Conference on Computer and Communications Security – ACM CCS*, pages 210–219, Washington, DC, USA, October 2004. ACM, ACM Press.

[12] M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *RFID Privacy Workshop*, MIT, MA, USA, November 2003.

[13] S. Piramuthu. HB and related lightweight authentication protocols for secure RFID tag/reader authentication. In *Collaborative Electronic Commerce Technology and Research – CollECTeR 2006*, Basel, Switzerland, June 2006.

[14] M. Rieback, B. Crispo, and A. Tanenbaum. The evolution of RFID security. *IEEE Pervasive Computing*, 5(1):62–69, January–March 2006.

[15] C. C. Tan and Q. Li. A robust and secure rfid-based pedigree system (short paper). In *International Conference on Information and Communication Security(ICICS)*, 2006.

[16] G. Tsudik. YA-TRAP: Yet another trivial RFID authentication protocol. In *International Conference on Pervasive Computing and Communications – PerCom 2006*, Pisa, Italy, March 2006. IEEE, IEEE Computer Society Press.

[17] I. Vajda and L. Buttyán. Lightweight authentication protocols for low-cost RFID tags. In *Second Workshop on Security in Ubiquitous Computing – Ubicomp 2003*, Seattle, WA, USA, October 2003.

[18] S. Weis, S. Sarma, R. Rivest, and D. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In D. Hutter, G. Müller, W. Stephan, and M. Ullmann, editors, *International Conference on Security in Pervasive Computing – SPC 2003*, volume 2802 of *Lecture Notes in Computer Science*, pages 454–469, Boppard, Germany, March 2003. Springer-Verlag.