Secure and Serverless RFID Authentication and Search Protocols

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

Abstract

With the increased popularity of RFID applications, different authentication schemes have been proposed to provide security and privacy protection for users. Most recent RFID protocols use a central database to store the RFID tag data. The 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 securely search for RFID tags will be increasingly useful.

1 Introduction

Radio Frequency Identification (RFID) technology is increasingly being deployed in diverse applications ranging from inventory management to anti-counterfeiting protection [25]. Features such as the ability for a reader to read data off an RFID tag located several meters away, make RFID tags an attractive replacement for barcodes which require close proximity to a reader before being read. Nonetheless, RFID tags have yet to supplant the ubiquitous barcode found on almost every grocery product. This slow adoption is partly due to the security and privacy concerns over the pervasive deployment of RFID tags. Such concerns include the illicit tracking of RFID tags which violate the privacy of the holders of the tags. Until these concerns are adequately addressed, large scale adoption of RFID is unlikely to materialize.

Recent work [7, 18, 22, 26] attempts to solve the RFID security and privacy problem by utilizing the “central database model”. There are three players in this model: an RFID reader, an RFID tag, and a secure central database. To obtain data from a tag, the reader first queries the tag and then forwards the tag reply to the central database. The reader obtains no useful information from the tag reply. After the database authenticates the reader and verifies that the tag reply is genuine, the database returns the tag information to the reader. While the central database approach provides security and privacy protections, it is dependent on a reliable connection between an RFID reader and the central database. Consider for example, a truck driver dispatched to an off-site location to collect some merchandise tagged with RFID tags. He has with him a PDA which doubles as an RFID reader. Due to the remote location, the truck driver is unable to connect to the central database to authenticate the goods. As a result, despite having an authorized reader and genuine RFID tags, the driver is unable to obtain the data.

A simple alternative, analogous to using a central database, is to download the information from the database onto the reader. The RFID reader can then continue to access the RFID tags as before. However, having multiple readers increases the likelihood of a stolen or misplaced reader. These compromised readers are more valuable than the readers under the central database model, since they contain information originally found only in the database. This information can include the unique ID and secret password of an RFID tag. An adversary can use this information to create fake RFID tags that are indistinguishable from the real ones. The adversary first obtains a “blank” RFID tag and then proceeds to store data from the compromised reader onto this blank tag. Since this fake tag has the same information as a real RFID tag, a reader is unable to distinguish between the two. In this paper, we suggest protocols that provide similar security and privacy protections as the central database model without requiring a persistent connection to the database. The protocols also prevent an adversary from using a compromised reader to create indistinguishable fake RFID tags.

After providing security and privacy protection to a single reader querying a single tag, a natural extension is to provide the same protection to situations where there is a single reader and multiple tags. One such situation is when a reader needs to search for a particular RFID tag out of a large collection of tags. As the number of RFID tags in circulation increases, the ability to search for RFID tags is invaluable when the reader only requires data from a few tags rather than all the tags in a collection. Authenticating each tag one at a time until the desired tag is found is a time consuming process. Surprisingly, the problem of RFID search has not been widely addressed in the literature, despite the availability of search capabilities in commercial RFID products. In this paper, we examine the challenges of extending security and privacy protection to RFID search, and suggest several solutions.

We make the following three 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 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 the tag. Third, we introduce the problem of searching for RFID tags with security and privacy protection, and suggest several solutions.

The rest of the paper is as follows. The next section reviews related work on RFID security. Section III explains security and privacy in the context of RFID. Section IV and Section V contain the authentication protocols and security analysis respectively. Section VI introduces the secure RFID search problem and presents several possible solutions. Section VII discusses the shortcomings of a serverless approach and how to overcome them. It also includes a discussion on the cost and efficiency of our protocols. Finally, we conclude in Section VIII.

2 Related Work

RFID security and privacy research can be broadly divided into two categories. The first category is protocol based. Its emphasis is on designing better protocols using mostly lightweight primitives [27] known to be implementable on RFID tags. Our paper falls under this category. The second category is hardware based. The emphasis is on improving RFID tag hardware to provide additional security primitives like elliptic curve cryptography. For the remainder of this section, the focus is on prior work done in the first category. A brief discussion of RFID hardware improvements is given at the end. Interested readers can refer to an online resource by Avoine [1] for up-to-date information, and recent survey papers [14, 24] for more details.

Early work by Weis et al. [31] used a backend database to perform RFID authentication. A reader querying the RFID tag will receive a $metaID$. The reader forwards this $metaID$ to the backend server which then retrieves the real tag ID for the reader. Every tag has a unique $metaID$ and will always reply with the same $metaID$ value when queried. This creates a privacy problem since an adversary can track the movements of a tag by repeatedly querying and comparing $metaID$ values. The authors proposed the randomized hash lock scheme to solve this problem. Under this scheme, the tag returns $(r, ID \oplus f_k(r))$ when queried by a reader, where 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 forwards this reply to a secure database which then searches its database for the ID/secret key pair that matches the tag reply. Once found, the tag ID is returned to the reader. Since every new reader query results in a different reply, the adversary is unable to track the tag.

Molnar and Wagner [21] pointed out that the randomized hash lock scheme does not defend against an eavesdropper. An adversary can eavesdrop on the communication between reader and tag to learn the tag reply, $(r, ID \oplus f_k(r))$. The adversary then uses this information to impersonate the RFID tag to fool a reader. In their paper, the authors suggest having both the reader and tag each contribute a random number, r_1 and r_2 respectively. Their approach assumes that the reader knows the tag secret k. After the reader and tag exchange random numbers, the tag replies with $ID \oplus f_k(0, r_1, r_2)$. Since the reader knows k, he can derive $f_k(0, r_1, r_2)$ and obtain ID. The protocol works without a central database. However, it does not consider the case of a compromised reader. An adversary with a compromised reader will know the tag secret of every tag the reader has access to. The adversary can then use this information to make duplicate tags to fool other readers. Our protocols address this particular vulnerability.

Dimitriou [7] is a more recent example of a protocol based on a database. In this protocol, both the reader and tag exchange random numbers, n_r and n_t, at the start of the query. The tag then returns $(h(ID_i), n_t, h_{ID_i}(n_t, n_r))$ to the reader, where ID_i is the tag secret. The reader learns nothing from this reply, and forwards it to the database. The database uses $h(ID_i)$ to determine the matching tag secret ID_i. This ID_i is applied to n_t and n_r to verify the tag reply. Once satisfied, the database updates the tag secret from ID_i to ID_{i+1}. The tag information, together with $h_{ID_{i+1}}(n_t, n_r)$, is returned to the reader. The reader completes the protocol by forwarding $h_{ID_{i+1}}(n_t, n_r)$ back to the tag. The tag determines ID_{i+1} independently, and applies it to the two random numbers used earlier. If the result matches $h_{ID_{i+1}}(n_t, n_r)$, the tag knows that the reader has been authenticated by the database. The tag updates its secret to ID_{i+1} and the protocol terminates. Otherwise, the tag retains the old secret ID_i. Similar protocols [18, 22] also use the idea of changing the tag secret after every query. A key feature of this protocol is how desynchronization between tag and server is avoided. A fake RFID tag will not be able to generate a reply to convince the database to update the tag secret ID_i. A rogue reader is unable to derive $h_{ID_{i+1}}(n_t, n_r)$ to convince an RFID tag to change its secret. Work by [20, 19] examines desynchronization attacks in greater detail.

While RFID with database protocols are relatively new, a similar problem is found in 3GPP mobile authentication [32, 12]. In 3GPP authentication, mutual authentication is required between the mobile user and network. Synchronization of sequence numbers used by a mobile user and the home network is also required. These requirements are similar to the mutual authentication between a reader and a tag, and the synchronization of tag secret between the database and the RFID tag.

An alternative method for RFID authentication is based on a "challenge and response" between a reader and a tag. Juels et. al. [16] observed that human authentication protocols can be applied to RFID, since RFID tags, like humans, have weak computational capabilities. They introduced HB protocol, in which a reader issues a new challenge to a tag each time it queries an RFID tag. The tag computes the binary inner product based on the reader's challenge, and returns the answer to the reader. The reader authenticates the tag by verifying the tag response. The HB+ protocol is an improvement over the HB protocol by using an additional binding factor from the tag to defend against an active adversary. Later work by [23, 9, 5] improves on this idea.

YA-TRAP [26] introduces a novel technique using timestamps in RFID authentication. This is a novel approach since 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. Chatmon et. a. [6] suggested an improvement to this protocol.

An assumption made by earlier research, as well as this paper, is that RFID tags are capable of executing cryptographic hash functions. However, most current commercial RFID tags do not provide these hash functions, mainly due to the higher production cost [31]. A cryptographic hash function requires additional gates to be implemented in the tag, raising the overall cost per tag. Common hash functions like MD4, SHA-1 and SHA-256 require between 7350 and 10868 additional gates [8]. This suggests that the majority of the proposed protocols are likely to be feasible only on expensive RFID tags attached to more valuable items. Recent work by [4] suggested using physically unclonable functions (PUF) in RFID tags since they only require 545 gates to implement. However, the same paper also noted that PUF-based hash functions are difficult to analyze since they are influenced by physical environment. How to design security protocols using PUF-based hash functions remain an open problem.

An orthogonal approach to RFID security focuses on changing the physical hardware of the RFID tag itself. Efforts by [3, 17, 2] investigated the possibility of building RFID hardware that is capable of performing public key based authentication. Their efforts have centered on using a particular flavor of public key cryptography based on elliptic curve cryptography (ECC). ECC has been suggested as a good replacement for RSA based public key cryptosystems since a 160-bit ECC offers the same level of security as a 1024-bit RSA encryption. While a public key cryptosystem for RFID tags greatly improves RFID privacy and security, it is also more costly to implement than cryptographic hash functions. Furthermore, it is unclear whether tiny sensor motes will be used in lieu of these RFID tags, since current sensor motes are already capable of efficiently performing ECC primitives [30, 29] and protocols [28].

3 RFID Privacy and Security

For RFID tags attached to personal items like a passport, exposing information from these tags to an unauthorized reader violates the privacy of the owner of the item. There are two ways information about a tag can be exposed. The first is when an unauthorized reader queries the tag and gets back the tag data. This can be solved by encrypting the tag reply such that only an authorized reader can decrypt the reply. The second is when an unauthorized reader obtains a constant reply from an RFID tag. The unauthorized reader can use this information to track the movements of the holder of an RFID tag. For instance, consider a tag attached to a passport. An unauthorized reader queries the tag and obtains a constant encrypted reply. Even though the unauthorized reader cannot decrypt the reply, it can compare tag replies at different locations. When the same tag reply is obtained in two separate locations, the unauthorized reader can infer that the holder of the tag has been to these two locations. This is also known as violating the “location privacy” of the tag. Location privacy can be solved by having each tag reply be different and unlinkable to previous tag replies.

RFID tags are also widely used as a means of identification. For example, an RFID tag can be attached to a container of pharmaceuticals so that a reader can query the tag and learn the contents without opening up the container. An adversary manufacturing counterfeit pharmaceuticals will attempt to create a fraudulent RFID tag to place onto his container of counterfeit drugs. An RFID reader that queries and accepts the fraudulent tag as a real RFID tag will then accept the counterfeit drugs as genuine.

A basic component of RFID security is to allow a reader to distinguish a real RFID tag from a fake tag. This is accomplished by having a secret known only to a reader and a genuine tag. The RFID tag can then use this secret to prove itself to a reader. An adversary attempting to create a fraudulent tag indistinguishable from a real tag needs to obtain this secret. The adversary has three methods to try to obtain this secret. The first is by eavesdropping on the communication between a reader and a tag. The second is by repeatedly querying the RFID tag to obtain enough information to derive the secret. Finally, the adversary can physically compromise the RFID tag to obtain the secret. In this paper, we only defend against the first two methods. Tamper proof hardware capable of foiling a physical attack is beyond the scope of this paper.

4 RFID Authentication

Two authentication protocols are presented in this section. The first transfers the data from a tag to a reader via a challenge and response. The second sends the tag data such that only an authenticated reader can understand. The protocols are evaluated in the next section. For the remainder of this paper, we consider the data a tag transfers to a reader to be the ID of the tag.

We consider an RFID reader denoted as R. Each R has a unique identifier r and an access list, L. 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 all the RFID readers. We assume that communications between R and the CA are performed via a secure channel. Subscripts are used to distinguish one reader from another. Thus RFID reader i will be R_i, with a identifier r_i and access list L_i.

Each RFID tag, T, contains a unique value id, a unique secret t, knowledge of functions $f(.,.)$ and $h(.)$. The id is an unique identifier for T, and is the tag data requested by a reader. The secret t is the tag secret known only by the tag itself and CA. The func-

Table 1. Notations

CA	Trusted party, responsible for authenticating readers and deploying tags
R_i	RFID reader i
r_i	id for RFID reader R_i
L_i	access list for RFID reader R_i
n	number of entries in L_i
T_i	RFID tag i
id_i	id for RFID tag T_i
t_i	secret for RFID tag T_i
$h(x)$	one-way hash function
$f(x, y)$	Concatenate x and y, then applying $h(.)$, $h(x\|\|y)$
l	number of bits of hash $h(.)$
m	CA defined number of bits, $m < l$

tion $h(.)$ is a one way hash function that outputs a bitstring of length l. A shorter length $m < l$ is predefined by the CA and known to all readers and tags. The function $f(.,.)$ is the hash function $h(.)$ applied to the concatenation of two arguments. For instance, a tag T applying $f(.,.)$ to an argument r sent by R will then have $f(r, t) = h(r||t)$ where $||$ denotes concatenation.

After reader R_i authenticates itself to CA and obtains access to RFID 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. 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 : For every entry in L_i, first determine
$h(f(r_i, t_x)||n_i||n_j)$, $x \in \{1, \cdots, n\}$.
Then check if the first b bits match
$[h(f(r_i, t_j)||n_i||n_j)]_b$ (5)

If match and $k \leq \frac{l-b}{2}$

$$R_i \rightarrow T_j : ans_r, \; ques_t^1, \cdots, ques_t^k \tag{6}$$

Else

$$R_i \rightarrow T_j : rand, \; ques_t^1, \cdots, ques_t^k \tag{7}$$

T_j : Check ans_r against
$ques_r^1, \cdots, ques_r^k$ (8)
If correct and $\forall x, y, ques_r^x \neq ques_t^y$

$$R_i \leftarrow T_j : (ans_t) \tag{9}$$

Else

$$R_i \leftarrow T_j : (rand) \tag{10}$$

where n_i and n_j are random numbers generated by R_i and T_j respectively. The first b bits of the resulting hash of $f(r_i, t_j)$, n_i and n_j concatenated together is depicted as $[h(f(r_i, t_j)||n_i||n_j)]_b$. The challenge generated by L_j to R_i is $ques_r^1, \cdots, ques_r^k$. This challenge consists of k random positions chosen from the last $l - b$ bits of $h(f(r_i, t_j)||n_i||n_j)$. R_i's response to the challenge, ans_r, is the actual bits in positions $ques_r^1, \cdots, ques_r^k$. The challenge from R_i to L_j and L_j's repose is $ques_t^1, \cdots, ques_t^k$ and ans_t respectively. For both reader and tag challenges, $k \leq \frac{l-m}{2}$. A random bit of length k, $rand$, is returned if ans_r or ans_t is incorrect.

The intuition here is to have both 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 a limited probability.

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

At the start of every query, R_i and T_j first exchange random numbers. R_i also sends his identifier r_i to T_j together with his random number. R_i then receives the first b bits of $h(f(r_i, t_j)||n_i||n_j)$, together with L_j's challenge. Using these first b bits, R_i consults his L_i to determine if there are any partial matches. Note that R_i has to first hash each entry in L_i with n_i and n_j before checking. The probability of having another tag with the same b first bits is

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

If there are no matches, R_i knows that T_j is not an RFID tag the CA has authorized him to access. R_i then returns a random k bit reply $rand$ and his challenge $ques_t^1, \cdots, ques_t^k$ to T_j. Otherwise, R_i returns the bit values in positions of $ques_r^1, \cdots, ques_r^k$ as ans_r and his challenge. In both cases, R_i's challenge must be different from T_j's 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 $ques_r^1, \cdots, ques_r^k$. A correct ans_r indicates that R_i is an authorized reader. Only an authorized reader knows $f(r_i, t_j)$, and thus is able to generate the correct $h(f(r_i, t_j)||n_i||n_j)$ to get pick out the positions. Tag T_j then checks if R_i's challenge is different from its own, and returns the correct answer reply, ans_t, to R_i's challenge if ans_r is correct. Otherwise T_j returns a random answer $rand$. R_i uses ans_t to determine whether T_j is a legitimate tag. After executing the protocol once, the probability of R_i identifying T_j is

Prob(Accurate identification)

$$= 1 - \left((\tfrac{1}{2})^b \left[(\tfrac{1}{2})^{2k} + (1 - (\tfrac{1}{2})^k)(\tfrac{1}{2})^k \right] \right)$$
$$= 1 - (\tfrac{1}{2})^{b+k}$$

4.2 Authentication Protocol 2

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

where n_i and n_j are random numbers generated by R_i and T_j respectively. T_j sends its id_j as $h(f(r_i, t_j)||n_i||n_j) \oplus id_j$. The tag also sends $h(f(r_i, t_j))_m$ to help R_i reduce the time taken to search through L_i. An unauthenticated reader cannot obtain id_j since he does not know $f(r_i, t_j)$, and hence cannot compute the $h(f(r_i, t_j)||n_i||n_j)$ necessary to obtain id_j. 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 organize 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 any entry in L_i then R_i ignores the tag. Note that a different random numbers 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 hash $h(\cdot)$ is a one way hash function, even knowing the entire $h(f(r_i, t_j))_m$ does not reveal $f(r_i, t_j)$.

To determine the value of m, we first define a *collision space* as $CS = 2^{l-m}$. This is the expected 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. The more privacy we wish, the smaller we set β. Thus, we have

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

The search time for R_i becomes $O(\frac{L}{2^m})$ since R_i can organize L_i into respective groups after T_j returns the first m bits of $h(f(r_i, t_j))_m$. Thus, 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

In this section, we analyze our protocols against different types of attacks. For each attack, we first give a brief description of the attack, and the common assumptions about the adversary. This is followed by an explanation of how the protocols defend against the attack. We denote the adversary as α, and a legitimate reader and tag as R_i and T_j respectively. A fake tag j impersonating the real tag j is depicted as $\hat{T_j}$.

Basic Privacy: The basic privacy attack occurs when α wishes to learn of the content of T_j. Consider for example, the tag T_j attached to a valuable container in a warehouse. Under this attack, we generally assume that α has a list of targeted RFID tag . α then queries every tag in the warehouse to decide the most valuable one to steal. 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: Under this attack, 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 passport. By repeatedly querying with a value that yields a consistent reply, α will be able to track the movements of T_j over time. This consistent reply becomes a signature of T_j.

Under our scheme, α can reuse the same n_α and r_α for every query, but cannot predict the random n_j generated each time by T_j. In protocol 1, we return the first b bits of $h(f(r_i, t_j)||n_i||n_j)$, while in protocol 2 we return the entire $h(f(r_i, t_j)||n_i||n_j)$ XORed with id_j. Since n_j is a random number chosen by the tag for each query, α learns nothing from repeated queries.

Note that in protocol 2, we also return $h(f(r_i, t_j))_m$ in step (4) which could be used to track T_j. This is an optimization step done to improve the search time for R_i. Step (4) can be modified to return just $h(f(r_i, t_j)||n_i||n_j) \oplus id_j$ to make tracking impossible. However, by keeping m small, the risk of tracking is minimal since there could be multiple RFID tags with the same first m bits.

Cloning: We consider the "skimming" attack described by Juels [13]. Under this attack, α will usually first query T_j and obtain 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 attempts 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 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.

Eavesdropping: Here α is able to observe *all* interactions between R_i and T_j. In other words, under protocol 1, α learns r_i, n_i, n_j, as well as the challenge and response between the R_i and T_j. Under protocol 2, α learns r_i, n_i, n_j, $h(f(r_i, t_j)||n_i||n_j) \oplus id_j$ and $h(f(r_i, t_j))_m$. α's goal is to use the data to launch any of the three attacks mentioned above. [1]

For both protocols, every transaction between R_i and T_j begin by both parties generating a different n_i and n_j. An α eavesdropping on the communication observes a different query and a different response each time, even if R_i is querying the same tag T_j. Thus, our protocols prevent α from using eavesdropping to launch a basic privacy attack or tracking attack.

An α can try to clone a tag by creating a fake tag with the eavesdropped information. However, α cannot control the random number n_r chosen by the R_i for each new query. Under both authentication protocols, each new query generates a new hashed result $h(f(r_i, t_j)||n_i||n_j)$. Since α does not know $f(r_i, t_j)$, α cannot derive the correct hash result, even if it knew what the random numbers were.

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, and α will learn everything about R_i and T_j. Hardware-based defenses

[1] This version of eavesdropping is stronger since it assumes that α can eavesdrop on both reader-to-tag and tag-to-reader communications. A weaker version of eavesdropping considered by some researchers assume that α can only eavesdrop on the reader-to-tag communication.

against physical attacks are beyond the scope of 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 $T_1, \cdots, T_n$. The goal is 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 $\hat{T_j}$ that can fool another authenticated RFID reader R_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 cannot derive t_j from $f(r_i, t_j)$.

Next, 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 be 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): The adversary α here does not try to obtain information from the tag, but rather tries to ensure that a legitimate R_i cannot access the data stored in T_j. To launch a DoS attack, α sends a large number of requests to the backend server to overwhelm the server. This results in a legitimate R_i being unable to access the database to obtain information about the tag. Under our solutions, a reader only needs to contact the server once to obtain an access list L_i. The reader is then able to interact with RFID tags without further interaction with the server. A DoS attack under our schemes will not affect readers that have already been authenticated. Only readers yet to obtain an access list are affected. Thus, our serverless protocol mitigates the damage of a DoS attack.

6 RFID Search

Complex RFID operations which require data from a large collection of RFID tags usually assume that the data have already been collected and stored into a database [10, 11]. Any RFID authentication protocol which provides security and privacy protection can be used. However, as the number of RFID tags increases, the cost of collecting data can be very high. More efficient methods for performing different RFID operations are needed. In this paper, we consider one such operation: searching for an RFID tag from a large collection of tags. Search is a basic and invaluable tool for sifting through large amounts of data. Consider for example, a large pharmacy stocked with RFID embedded medication. A pharmacist wanting to find a particular drug can broadcast his query and receive an answer. Due to the limited broadcast range of RFID readers, the pharmacist can even determine the approximate locality of the medication by directing the RFID reader at different locations, i.e. different shelves.

Ideally, we want a reader to be able to query for a specific tag and have only that tag to reply. To illustrate, we have R_i wanting to find the tag T_j.

$$R_i \rightarrow T* \quad : \quad id_j \tag{1}$$

$$T* \quad : \quad \text{If } id = id_j \tag{2}$$

$$R_i \leftarrow T_j \quad : \quad \text{Reply} \tag{3}$$

where $T*$ refers to an arbitrary tag in the collection. However, this simple protocol does not provide any privacy or security protections. An adversary, for example, can query for valuable tags to steal. To provide security and privacy, an RFID tag should authenticate the reader before replying. Also, the RFID reader should ensure that only genuine RFID tags receive his query. This prevents an adversary from learning the content of the query. The adversary knowing the query and observing a reply, can conclude that a particular tag is in the collection, since only a tag matching the query will reply. We can thus characterize the 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.**

A solution is for the reader to issue a search request 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 content of the query, observing the existence of an answer is not useful. For the remainder of this section, “query” and ”search request” are used interchangeably. The secure search protocol is as follows.

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

The search request for id_j is sent as $h(f(r_i,t_j)||n_r) \oplus id_j$. A tag needs to have the tag secret t_j to successfully execute step (2) and obtain id_j. Since α does not know t_j, he is unable to determine what the reader is searching for. Each reader’s query is different due to the random n_r generated for each new search request. Thus, even if the reader repeatedly searches for the same tag, α will obtain a different search request each time. A reader receiving a tag reply $h(f(r_i,t_j)||n_t) \oplus id_j, n_t)$ needs $f(r_i,t_j)$ to obtain id_j, and $f(r_i,t_j)$ is known only to the authorized reader. Thus, α cannot create a fake tag $\hat{T_j}$ to fool the reader.

6.1 Security Analysis

The security analysis in section V also applies to the search protocol with one exception, the search protocol presented above is not resistant to tracking.

Consider the following attack where α eavesdrops on a transaction between a reader and a group of tags. Adversary α is unable to decrypt the query or the reply, but can detect the presence of a query and reply. α then broadcasts the same query repeatedly. Since the query is legitimate, the tag with the corresponding value will reply. Even though the reply is different every time due to the random n_t generated by the tag, there can only be one reply since each tag has its own unique secret t. α can extend the attack by isolating each tag in the group and repeating the query, waiting for a reply. α then combines this with physical observation to determine the identity of a tag.

We stress that the tracking attack presented here is 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 reasons increase the difficulty of launching a tracking attack via the RFID search protocol.

This attack 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 produces a unique reply, the reply becomes an identifier for a particular tag. Encryption does not solve the problem, since encryption only prevents an adversary from learning the content of a message, but not that a message has been sent.

6.2 Search Protocol Improvements

Here we suggest several improvements to the search protocol to minimize the impact of tracking. One solution is to force the reader to use a different random number n_r for each new query. This can be accomplished by having the RFID tag store a list of random numbers used in earlier queries. When a query arrives with an n_r that appears in this list, the tag will refuse to reply. This way, an adversary will not be able to replay an eavesdropped query. An incrementing counter cannot be used by the tag to store the random numbers since a legitimate reader will generate a new random number each time. Below, we present the protocol where a tag can only remember the last used random number.

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

where $oldn$ is the previous random number used. Now, α 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 next time R_i does a search query to obtain a different random number, n'_r. α can now 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. Provided 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 one more successful query than the number of tags to be always successfully track one tag. By 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 $n \cdot m + 1$ queries. However, this method is ineffective 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 id_m to denote the first m bits of a generic tag's id. The protocol is as follows.

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

Under 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 anti-collision 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 to generate the correct hash value. Furthermore, each party contributes a random number n_r and n_t that make up the final hash value needed to successfully obtain the id_j. This prevents an adversary from launching a replay attack from either the query or reply.

This solution does not work well when the id for each tag is 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 can decipher the request.

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

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

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 also avoids leaking any 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 to reply to create noise. We can estimate λ by solving $1 - (1 - \lambda)^S \geq \gamma$. The additional work done by reader to filter out the noise is $O(\lambda \cdot S)$. However, this solution only performs well when we have a reliable S, for example, a group tags are placed in a shipping container.

7 Additional Discussion

Despite the shortcomings of the central database model, it does have two advantages over a serverless solution. 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. To revocate a reader, the central database simply ignores 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 RFID tags are passed from one owner to another. 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 [13, 15]. 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 can provide 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 product 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 $f(r, t^2)$ as well. We can simply extend the number of secrets per tag to as fine a level of access control as desired.

Finally, we discuss cost and efficiency. For our authentication protocols, the first protocol requires the tag to perform two hash functions, $f(.,.)$ and $h(.)$. The second protocol requires three hash functions, $f(.,.)$ once and $h(.)$ twice. For the search protocols, the second search improvement requires the tag to execute two hash functions, and the remaining search improvements require three hash functions. The cost for our protocols is higher than alternative protocols [31, 21, 26] which require the tag to perform only one hash function. The additional hash functions allows our protocols to be serverless and yet avoid exposing the tag secret to the reader. Considering communication cost, the first authentication protocol requires transferring $2 \cdot |n| + |r_i| + b + 2 \cdot |k + ans|$ bits. Assuming that both reader and tag ids have the same length, authentication protocol 2 requires $2 \cdot |n| + 2 \cdot |id_j| + m$ bits. The communication cost for search protocols is higher since the reader's query contains of the tag id he is looking for. Again assuming both tag and reader ids have the same length. search improvement 1 transfers $3 \cdot |id| + 2 \cdot |n|$ bits. Improvements 2 and 3 transfers $2 \cdot |id| + m + 2 \cdot |n|$ bits and $3 \cdot |id| + 2 \cdot |n|$ bits respectively.

In terms of efficiency, the reader R_i has to perform $|L_i|$ hashes and search $|L_i|$ entries for each new query under authentication protocol 1, where $|L_i|$ is the size of the access list. For authentication protocol 2, the reader needs to perform $|L_i|$ hashes once to derive $h(f(r_i, t_*))$. For each new query, the reader only performs the hash for replies that match the first m bits of $h(f(r_i, t_*))$, resulting on average hashing and searching $\frac{|L_i|}{2^m}$ entries. The reader's performance for search protocols is very efficient since the reader only needs to check the access list for the entry it is looking for.

8 Conclusion

In this paper, we present authentication and search protocols for RFID tags. Our authentication protocols provide both tag-to-reader and reader-to-tag authentication and are resistant against common RFID attacks. A major departure from the previous research is that our schemes do not require a persistent connection to a central database. We also introduce a new problem of performing secure search for RFID tags. We examine the difficulties in designing a secure search protocol, and provide several solutions. Finally, we also consider the implicit advantages of having a central database and suggest solutions for overcoming them.

Acknowledgment

This project was supported by US National Science Foundation award CCF-0514985.

References

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

[2] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede. An elliptic curve processor suitable for RFID-tags. Cryptology ePrint Archive, Report 2006/227.

[3] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede. Public key cryptography for RFID-tags. RFIDSec 06.

[4] L. Bolotnyy and G. Robins. Physically unclonable function -based security and privacy in rfid systems. In *International Conference on Pervasive Computing and Communications (PerCom)*, 2007.

[5] J. Bringer, H. Chabanne, and D. Emmanuelle. HB^{++}: a lightweight authentication protocol secure against some attacks. In *SecPerU 2006*.

[6] C. Chatmon, T. van Le, and M. Burmester. Secure anonymous RFID authentication protocols. Technical report, Florida State University, Department of Computer Science, 2006.

[7] T. Dimitriou. A lightweight RFID protocol to protect against traceability and cloning attacks. In *SecureComm*, 2005.

[8] M. Feldhofer and C. Rechberger. A case against currently used hash functions in rfid protocols. In *OTM Workshops (1)*, 2006.

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

[10] H. Gonzalez, J. Han, and X. Li. Mining compressed commodity workflows from massive rfid data sets. In *CIKM 2006*.

[11] H. Gonzalez, J. Han, X. Li, and D. Klabjan. Warehousing and analyzing massive rfid data sets. In *ICDE 2006*.

[12] A. Herzberg, H. Krawczyk, and G. Tsudik. On travelling incognito. In *IEEE Workshop on Mobile Computing Systems and Applications*, 1994.

[13] A. Juels. Strengthening EPC tags against cloning. In *WiSe '05*.

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

[15] A. Juels and R. Pappu. Squealing euros: Privacy protection in RFID-enabled banknotes. In *Financial Cryptography – FC'03*.

[16] A. Juels and S. Weis. Authenticating pervasive devices with human protocols. In *Advances in Cryptology – CRYPTO'05*, 2005.

[17] S. Kumar and C. Paar. Are standards compliant elliptic curve cryptosystems feasible on RFID? RFIDSec 06.

[18] S.-M. Lee, Y. J. Hwang, D. H. Lee, and J. I. L. Lim. Efficient authentication for low-cost RFID systems. In *ICCSA 2005*, 2005.

[19] T. Li and R. H. Deng. Vulnerability analysis of EMAP - an efficient RFID mutual authentication protocol. In *Second International Conference on Availability, Reliability and Security – AReS 2007*.

[20] T. Li and G. Wang. Security analysis of two ultra-lightweight RFID authentication protocols. In *IFIP SEC 2007*.

[21] D. Molnar and D. Wagner. Privacy and security in library RFID: Issues, practices, and architectures. In *CCS*, 2004.

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

[23] S. Piramuthu. HB and related lightweight authentication protocols for secure RFID tag/reader authentication. In *CollECTeR 2006*.

[24] M. Rieback, B. Crispo, and A. Tanenbaum. The evolution of RFID security. *IEEE Pervasive Computing*, 2006.

[25] C. C. Tan and Q. Li. A robust and secure RFID-based pedigree system (short paper). In *ICICS*, 2006.

[26] G. Tsudik. YA-TRAP: Yet another trivial RFID authentication protocol. In *PerCom 2006*.

[27] I. Vajda and L. Buttyán. Lightweight authentication protocols for low-cost RFID tags. In *Ubicomp 2003*.

[28] H. Wang and Q. Li. Distributed user access control in sensor networks. In *DCOSS*, 2006.

[29] H. Wang and Q. Li. Efficient implementation of public key cryptosystems on mote sensors (short paper). In *ICICS*, 2006.

[30] H. Wang, B. Sheng, and Q. Li. Elliptic curve cryptography based access control in sensor networks. *International Journal of Sensor Networks*, 2006.

[31] 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 – SPC 2003*, 2003.

[32] M. Zhang and Y. Fang. Security analysis and enhancements of 3GPP authentication and key agreement protocol. *IEEE Transactions on Wireless Communications*, 2005.