Password Streaming for RFID Privacy

Victor K. Y. Wu
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Email: vwu3@illinois.edu
Phone: 650 521 7227
Address: 507 E. Clark St., Apt. 23, Champaign, IL, 61820
Roy H. Campbell
Department of Computer Science
University of Illinois at Urbana-Champaign
Email: rhc@illinois.edu
Phone: 217 333 0215
Address: Thomas M. Siebel Center for Computer Science, 201 N. Goodwin Ave., Rm. 3122, Urbana, IL, 61801

Abstract

In this paper, we propose a novel privacy-protecting RFID system using *password streaming*. We avoid the chicken-and-egg problem by eliminating the initial flow of a reader "knocking on the door" of a tag. Instead, our proposed scheme assumes a reader knows the presence of multiple tags, in its transmit range, and immediately starts streaming tag passwords, associated with tag IDs, in a broadcast fashion, allowing for tags to respond in between passwords. A tag that receives a matching password, updates it locally, and sends the update back to the reader. The reader thus learns which tag is present in its domain. A centralized back-end system coordinates the password streaming of multiple readers in their respective domains. Thus, tag passwords, and tag locations are closely tracked. As a result, a "fresh" password that no tag accepts in a domain, is quickly rendered "stale", since the password is quickly updated in another domain. This prevents an attacker to use the stale password to compromise privacy.

Our proposed scheme pushes complexity to the readers, and even more so, to the supporting infrastructure, in the very spirit of RFID. The benefits include an efficient, and light-weight privacy system, as well as being able to defend against a stronger attacker model, than that introduced in [5].

Index Terms

RFID, privacy, security, singulation

Password Streaming for RFID Privacy

I. Introduction

RFID (radio frequency identification) technology provides an efficient, and automated means to access information regarding physical objects. RFID tags carrying unique identifiers, which we call tag IDs, are affixed to the objects, usually one per object. An RFID reader queries a tag by transmitting a wireless signal, to which the tag responds with its tag ID. The reader then uses the tag ID as a pointer to access a database to quickly determine potentially a plethora of information regarding the object. The tag ID itself presumably follows some format. Therefore, some immediate information is already garnered without even accessing the database.

Unfortunately, the presence of malicious entities, (which we call attackers) precludes the *privacy* of this object information. In this paper, we focus on defending against *inventorying*, and *tracking* of tags, as a means to protect privacy [1]. Inventorying refers to an attacker acquiring a tag ID, and using it to learn information regarding its associated object, through the aforementioned means. Tracking refers to an attacker physically, (or otherwise) following a tag by successively querying it, in order to indirectly learn information regarding the associated object. Note that even if the tag does not respond with its tag ID, tracking is still possible, if for example, a meta identifier is used instead.

In this paper, we seek a solution to the privacy problem. Since RFID tags, and especially passive tags, which we exclusively consider, are severely constrained in their computational power, our solution must push most of the computational requirements to the readers, and their supporting infrastructure. A seemingly plausible solution is the use of passwords, specific to tag IDs. That is, to access a tag, a reader has to first authenticate itself by providing the tag's password. Presumably, the password, associated with this tag's ID, is initially stored in a database that the reader has access to. But to know which password to use, the reader has to first know the tag's ID, which if it knew, would defeat the goal of the solution. In other words, we have a chicken-and-egg dilemma [2]. We discuss several solutions in the literature that steer clear of this dilemma. We then offer a novel solution that avoids the shortcomings of these previous ones.

In [3], the authors use the physical layer component of *distance*, as a key component of trust between a reader, and a tag, and thus, provide privacy for the tag. That is, an attacker is assumed to be constrained in its distance to a tag, and a closer querying entity is assumed to be likely a legitimate reader. Also, a distant transmitter's signal arrives with a weak signal strength at the tag. The paper develops a distance-based metric (a calculation based on the tag's received signal-to-noise ratio), and uses it to give a tiered authentication model. That is, as the tag becomes more confident its querying entity is legitimate, it reveals more bits. This scheme effectively removes the need for passwords, and instead uses distance. However, the simulations in the paper indicate that the metric used does not provide a very strong correlation with distance, since multi-path propagation, and other effects make an RFID channel rather unpredictable. Only at a scale of larger distances does the correlation hold. Another weakness is that a tag must be able to measure the power of its received signals, and perform some calculations to determine the metrics. This is not immediately feasible for passive tags. Therefore, this scheme can at most can be a secondary privacy-protecting tool.

In [2], the authors also use a password-less system based on *secret sharing*. In a (t, n) sharing scheme, a secret is divided into n shares, and distributed. If at least $t \leq n$ shares are recovered, the secret can be reconstructed. Anything less than t shares reveals nothing about the secret. The authors use the version of the scheme developed by Shamir [4]. Thus, they call their tags, Shamir Tags. A tag ID is first used to generate its secret shares. The shares are concatenated, and the resulting string replaces the tag ID in the the tag. A reader querying the tag receives random bits from the string, with the position of the bit indicated. Eventually, enough shares are collected, and the secret tag ID can be reconstructed. The process is accelerated by using caches in the reader. An attacker is allowed only a limited successive number of queries. The assumption is that the attacker has to eventually leave the tag range after a period of time, for fear of being caught. As a result, the tag can throttle responses by slowing down its transmissions accordingly. This is reflected in the weakened attacker model of [5]. This scheme is indeed strong. However, if we do allow an attacker to query a tag for a long period of time, or if we assume weaker throttling, the attacker does finally acquire the tag ID. The weakened attacker model may allow a reader, and a tag to occasionally talk to each other without attacker interference, to perhaps update the secret shares. But if we adjust the attacker model slightly to allow for another attacker, for example, to listen to that communication as well, the scheme fails. Therefore, the password-less nature of Shamir tags is not necessarily effective in stronger attack scenarios. An update mechanism is also missing.

In [5], the authors do use a password-based system. Rotating *pseudonyms* transmitted by a tag are used to confuse attackers, and are mapped by legitimate readers to the desired unique tag ID. Specifically, a tag responds to a query from a legitimate reader with a pseudonym α_i, and the reader maps the pseudonym to a tag ID. The reader then authenticates itself by sending a password β_i, unique to α_i. Then, the tag authenticates itself to the reader with a password γ_i, unique to α_i. Finally, α_i, β_i, and γ_i are updated using an exclusive or (XOR), one-time pad algorithm, that we borrow, and describe in Section IV-A. The algorithm guarantees that for an attacker to learn anything about α_i, β_i, and γ_i, it must listen to m successive reader-to-tag communications, where m

is a system parameter. This scheme suffers from a number of inefficiencies. First, it requires many communication flows to establish privacy. Secondly, the pseudonyms impose additional memory constraints on the tags.

In this paper, we propose a novel privacy-protecting RFID system using *password streaming*. We avoid the chicken-and-egg problem by eliminating the initial flow of a reader "knocking on the door" of a tag. Instead, our proposed scheme assumes a reader knows the presence of multiple tags, in its transmit range, and immediately starts streaming tag passwords, associated with tag IDs, in a broadcast fashion, allowing for tags to respond in between passwords. A tag that receives a matching password, updates it locally, and sends the update back to the reader. The reader thus learns which tag is present in its domain. A centralized back-end system coordinates the password streaming of multiple readers in their respective domains. Thus, tag passwords, and tag locations are closely tracked. As a result, a "fresh" password that no tag accepts in a domain, is quickly rendered "stale", since the password is quickly updated in another domain. This prevents an attacker to use the stale password to compromise privacy.

Note that an immediate advantage of our scheme is that it is itself a singulation algorithm, although very different from the traditional tree-walking [6], and Aloha-based [7] algorithms. We do not require privacy to be a separate communications layer. Our proposed scheme is efficient, and light-weight. As we describe the details in Sections III, and IV, we point out algorithmic, and memory requirements, and show that they are extremely low for tags, very much in the spirit of RFID. This is because our proposed scheme relies heavily on the supporting infrastructure of the RFID system. RFID technology emphasizes low-cost tags, and that complexity should be pushed to the readers. We believe that this is true also for privacy. We therefore place the computational requirements at the readers, and even more so on the centralized back-end system. One immediate benefit is that we can defend against a stronger attacker model, than that introduced in [5].

The rest of the paper is organized as follows. Section II sets up a relevant system model. Section III gives a purposely generalized description of our proposed scheme. In Section IV, we refine our scheme to defend against increasingly stronger attacks. Section V describes how the supporting infrastructure of an RFID system can be further explored, in relation to privacy. Finally, Section VI concludes the paper, and offers future research directions.

II. System Model

We describe the system model. We first define the different components in our privacy system, for conciseness in the rest of the paper. Fig. 1 highlights the interactions of these components.

- **tag**: A tag refers to a legitimate passive RFID tag, attached to a physical object by the user (or application) of the privacy system.
- **reader**: A reader refers to a legitimate RFID reader. All readers are authorized to interrogate any tag. All readers are controlled by the centralized back-end system.

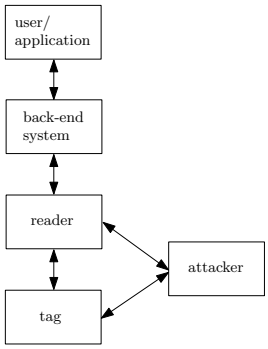

Fig. 1. **RFID Privacy System Hierarchy:** The attacker is a low-level entity. It can only interact with tags, and readers.

- **back-end system**: The back-end system is the abstract upper layer that sits on top of the physical, and data link layers of the tags, and readers. It coordinates how readers are to interact with tags in their respective domains, as well as manage tag IDs, and passwords. It thus provides a privacy-protecting RFID service to the user (or application).
- **attacker**: An attacker is a malicious entity that can interact only with tags, and readers wirelessly, using radio frequency signals. It may be stronger or weaker than a reader, in terms of transmitting, receiving, and processing these signals. It may even physically attack tags, and readers. It does not have access to the back-end system. The back-end system also cannot detect attackers directly.

Our system model assumes a large physical space where readers are initially distributed throughout. The readers are used to singulate tags in their respective wireless ranges (which we call domains). The readers remain stationary, and their domains do not overlap. Each reader has a direct connection to the back-end system (via a dedicated wifi access point, using a separate channel from RFID communications, for example). Tags are also distributed throughout the physical space. They are mobile, and each tag may at most be in one domain at any time instant. Wireless signals from readers, and tags are confined to their current domain, and do not interfere with the wireless signals from other domains. Attackers are mobile, and can simultaneously communicate with readers, and tags in at most one domain. We also assume some minimal level of communications, and physical security. That is, attackers wirelessly communicating with each other, or otherwise transmitting with large powers are evenutally detected. Attackers in the physical space, and even just outside the boundary, are eventually caught if they do not leave after a relatively short period of time. Too many of them, especially if they are in motion, also lead to their quick capture. Fig. 2 summarizes these ideas.

III. Proposed Scheme

We describe our proposed scheme, in a purposely generalized notion. In Section IV, we provide specific details according to increasingly stronger attacker models, and requirements of different applications.

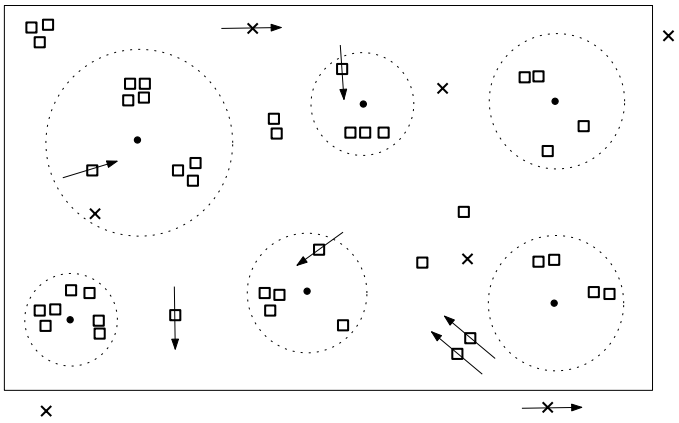

Fig. 2. **System Model:** The large rectangle indicates the boundary of the physical space. The small dots are stationary readers. Each has a domain indicated by a dotted circle. The squares are tags, and the crosses are attackers. Tags, and attackers that are currently moving are indicated by arrows, showing their respective movement directions. Note that this architecture models a space in an inventory area of a storage building.

Our proposed scheme is most easily described in the context of a reader querying multiple tags in its domain. In other words, our scheme is a singulation algorithm. As mentioned in Section I, our scheme eliminates the chicken-and-egg problem. The reader streams passwords (chosen according to some algorithm) continuously, in a broadcast fashion, to the tags in its domain. The passwords of all tags in the physical space are initially stored in the back-end system, and are associated with specific tag IDs. Time is allowed between each streamed password for tags to respond. When tags receive passwords that do not match their own respective passwords, they respond with a common, and pre-determined dummy signal. When a tag does receive a matching password, it immediately updates it, and sends the result back to the reader. It then goes into sleep mode (that is, does not respond to any reader queries) until the reader has finished the current singulation session. Passwords are assumed to be unique, and remain so even after updates. When a streamed password does not match, the reader receives a summation of identical dummy signals, and can thus conclude the tag associated with that password is not present in its domain. When a streamed password does match, the reader can subtract out the dummy signals to extract the updated password, and pass on the update to the back-end system. The associated tag is thus singulated. A singulation session is complete when all tags have gone to sleep, and the reader no longer receives any responses.

We present pseudo-code of our proposed scheme. p_t is the password stored in a tag. t_{id} is the tag ID of the tag associated with the chosen password that the reader streams. IDs contains the tag IDs of the tags singulated so far in the current singulation session.

Tag Protocol

1) Listen to password p_r, streamed by reader.
2) **If** $p_t = p_r$
   - **Then** $p_t :=$ new_password(). Send p_t to reader. Go to sleep.
   - **Else** send dummy signal d to reader.
3) **If** not asleep
   - **Then** Go to 1).

Reader Protocol

1) $IDs := \langle\rangle$.
2) $[t_{id}, p_r]$:= choose_password(back-end system). Broadcast p_r.
3) Listen to response x, from the tags.
4) **If** $x =$ null
   - **Then DONE**.
   - **Else** subtract out d from x, and store the result in p_u.
5) **If** $p_u \neq 0$
   - **Then** update_password(back-end system, t_{id}, p_u). $IDs := \langle IDs, t_{id}\rangle$.
6) Go to 2).

IV. Attacks and Defenses

We vary our attacker model by increasingly strengthening the attackers. In each case, we make specifications, and/or modifications to our generalized scheme in Section III to defend against the progressively stronger attack scenarios. Note that the ultimate goals of the attackers are to inventory, and track tags. Therefore, our defenses are focused on protecting against these two aspects of privacy. We also consider how privacy can be traded off for efficiency in some of these cases, as determined by the application.

A. Eavesdropping

Eavesdropping occurs when an attacker listens to the communications between readers, and the respective tags in their domains. Assuming an attacker knows the dummy signal, it can collect passwords, and even differentiate which ones are stale (that is, have been updated), and which ones are still fresh. Note however, that tag IDs are never exposed in any RFID channel, since they always remain in the back-end system, and in readers. Thus, attackers can never succeed in inventorying via eavesdropping.

1) One-way Eavesdropping: Since tags are passive, readers broadcast at higher powers than tags backscatter. As a result, reader-to-tag channels are much more vulnerable to eavesdropping than tag-to-reader channels. Suppose only reader-to-tag eavesdropping is possible. Then, an attacker acquires only very limited information to facilitate tracking. For example, if the attacker detects that the same password has been streamed by multiple readers, in their respective domains, over a relatively long period of time, and assuming passwords are not often reused, the attacker knows that the associated tag is likely not currently located inside any domain. This requires an attacker to eavesdrop for a relatively long period of time, move around to different domains, and/or enlist the help of other attackers, and communicate with them. However, Section II assumes our system model has minimal communications, and physical security that makes this unlikely to succeed.

2) Two-way Eavesdropping: If an attacker can eavesdrop on both reader-to-tag, and tag-to-reader channels, it can easily track tags by following their respective password updates. With the help of other attackers, a tag can be followed as it moves from one domain to another. Even if an attacker misses a password update (as will likely occur due to our system model assumptions, which are essentially equivalent to the assumptions of the weakened attacker model in [5]), and thus loses track of a tag, the attacker can quickly resume following

it in the next singulation session of the reader corresponding to that tag's domain. To defend against this, we modify our proposed scheme in Section III, by using the exclusive or (XOR), one-time pad algorithm in [5]. That is, a tag stores the vector $\Delta = \{\delta_1, \delta_2, \ldots, \delta_m\}$, where m is a parameter that determines the eavesdropping resistance. The password of the tag is δ_1. The back-end system also has a copy of Δ. When the reader streams the correct password, namely δ_1, the tag generates another vector $\tilde{\Delta} = \{\tilde{\delta_1}, \tilde{\delta_2}, \ldots, \tilde{\delta_m}\}$, and updates Δ locally as

$$\Delta := \{\delta_2 \oplus \tilde{\delta_1}, \delta_3 \oplus \tilde{\delta_2}, \ldots, \delta_m \oplus \tilde{\delta_{m-1}}, 0 \oplus \tilde{\delta_m}\},$$

where $\oplus$ is the exclusive or (XOR) operator. The tag then responds to the reader with $\tilde{\Delta}$, so that the back-end system can update its copy of Δ. Note that in this algorithm, the updated password is not sent back to the reader. Instead, only the bit-wise difference is sent. Since memory is built into this scheme, and the one-time pad is information theoretically secure, the attacker has no information about the current password, δ_1, unless it had eavesdropped on all of the last m Δ updates of this tag, which is unlikely to have occurred as long as m is large enough. Singulation efficiency obviously degrades as m is increased, and is traded off for eavesdropping resistance, and thus, tracking resistance.

B. Tag Spoofing

Tag spoofing occurs when an attacker masquerades as a tag, to a reader. For example, an attacker listens to a streamed password, and responds with a password update accordingly. If the streamed password matches with a tag's password, the combined response of the tag with the attacker quickly allows the reader to know the presence of the attacker. If the password does not match any tag's password, then the attacker gives the only non-dummy signal response, and the reader is fooled. Note that as a result, there is a tag somewhere that no longer has a password synchronized with the back-end system. This creates an anomaly that can be used to notify the application to take the requisite action. That is, suppose the orphaned tag is in some domain, and the reader there begins a singulation session. The reader is not able to finish the session because the orphaned tag constantly responds with the dummy signal, since its password is not synchronized. A more active defense against tag spoofing is to require tags to authenticate themselves to readers when they respond with updates. One method is to use separate dedicated passwords. Another way is to use password histories, stored in the tags. That is, tags can authenticate themselves by responding with old passwords. The reader can also issue challenges for specific passwords in the tag's history. The idea of recycling passwords originally intended to authenticate readers, to now authenticate tags, deserves further analysis.

Note that tag spoofing does not directly threaten privacy, since attackers are not inventorying, or tracking tags. This however, does not prevent an attacker to launch a tag spoofing attack, as part of a combination attack, to compromise privacy.

C. Reader Spoofing

Reader spoofing occurs when an attacker masquerades as a reader, to a tag. This is fundamentally a reader authentication issue. That is, given our defenses to eavesdropping as described in Section IV-A, attackers are unlikely to acquire passwords. As such, tags only respond with dummy signals to attacker queries, and therefore, no information is leaked. However, suppose that the minimal level of communications, and physical security mentioned in Section II is weakened (or equivalently, that the attackers are strengthened) to the level that eavesdropping allows attackers to acquire passwords. It is within this context that we provide defenses to reader spoofing to protect against tracking. Note that inventorying is not possible, for the same reason explained in Section IV-A.

1) Replaying Single Passwords: Once a reader acquires the password (through whatever means) of a tag, the tag is in great danger. That is, the attacker can immediately transmit the password to the tag, and the tag replies with a password update to the attacker. Note that in this case, we are assuming the attacker also knows where the tag is. Since the back-end system no longer has password synchronization with this tag, the tag is essentially lost. The attacker gains total control of the tag, and can thus track it. This applies even if we use the exclusive or (XOR), one-time pad algorithm. To defend against such an attack, we can use a throttling concept, similar to the ideas presented in [5], [2]. Recall, a tag does not respond to any queries when it is in sleep mode. When a tag goes into sleep mode after it has been singulated, a clock begins in the tag, and is also synchronized with a clock in the back-end system via the reader. The tag wakes up after the current singulation session is complete, or a throttling time period (possibly randomly, and dynamically generated) elapses, whichever occurs later. Usually, the throttling time period is longer than the singulation session time. Assuming that the attacker knows the throttling time period, its best strategy is to query the tag with its password immediately after it wakes up. As a defense, the reader can also schedule its password streams to also transmit that password at that same time. This does not prevent the attacker from losing password synchronization with the tag. But it does prevent de-synchronization of the reader with the tag. We note that this defense is rather weak, and even difficult to implement, since it is not immediately clear how to schedule password streams to query tags at the correct times. However, the attack provided here is rather strong, since acquiring both the password, and location of tag is extremely difficult, as explained in Section IV-A. In the following, we provide a more realistic reader spoofing attack.

2) Replaying Password Streams: Another means of spoofing readers is replaying password streams, again assuming that an attacker can eavesdrop. Note that in this case, we assume the attacker does not know where the tags associated with the passwords are located. There are several approaches the attacker can take. First, it can replay the reader-to-tag password stream in the same domain. If the replay takes place shortly after the original stream, not much information can be garnered by the attacker. Passwords that had not matched in the original

stream still do not match (assuming no new tags join the domain). Passwords that had matched are no longer be valid. Therefore, this attack is rather weak. An alternative is that the attacker can record the stream, and replay it in another domain (by sending the stream to another attacker, or by moving to the other domain). The idea is that non-matching passwords in the original stream might belong to tags in other domains. However, this attack is again rather weak. For example, the readers can dynamically keep track of where tags are, with the help of the back-end system. So, as long as tags are not highly mobile (which is a good assumption given our example of an inventory area), readers only potentially broadcast a non-matching password when a tag has left its domain, (and has not yet been singulated in another domain), or a new tag has entered its domain. As well, by the time the attacker, (or its affiliate) replays a potentially matching password in another domain, it is likely that the tag that just arrived already has had its password updated, thus rendering the attacker's password stale. A more aggressive defense is for a reader to purposely inject garbage intermittently in its password streams. This obviously slows down singulation times, but the added privacy is that an attacker replaying such a stream is slowed down, causing its passwords to become stale before they can be used, reminiscent of throttling. Note that garbage injection does not help if the stream is replayed (by the attacker's affiliate) in real-time. A more offensive defense strategy is for a reader to inject stale passwords, instead of, or in addition to garbage. Tags are alloted additional memory to hold their password histories. A tag that receives queries containing stale passwords, records this information, and relays it to its reader. This information is collected, processed in the back-end database, and the results are reported to the application, which can take further action to capture the attackers. Of course, a counter-attack is for attackers to record their eavesdropping histories. Note however, that despite our strengthened attackers in this section, their combined abilities are still assumed to be weaker than the centralized, processing power of the back-end system. Thus, the counter-attack is not be very effective, for example, if the attackers' combined history memory is smaller than that of the back-end system.

Second, the attacker can replay the tag-to-reader stream. At first glance, this might seem quite troubling since the passwords are fresh. However, note that the passwords from this stream is useless in other domains. If the stream is replayed in the same domain, then we are back to the situation in Section IV-C1.

Finally, replaying a combination of both the reader-to-tag, and tag-to-reader streams, and even the streams of multiple domains requires detailed analysis which is beyond the scope of this paper, (but is very much the subject of future research).

D. Physically Attacking Tags

We discuss the possibility of attackers physically capturing tags (temporarily, or permanently), and thus having unfettered access to its memory. This certainly is well-beyond the capabilities of a realistic attacker. But we offer a discussion to illustrate the inherent privacy aspects of our proposed scheme.

Tag IDs are naturally stored in tags. However, they are actually never transmitted in any channel in our proposed scheme. It is sufficient for the back-end system to maintain a mapping between tag IDs, and their passwords. Therefore, tag IDs can actually be eliminated from the memory of tags, and thus, such attacks are unsuccessful for inventorying. If the attacker returns the tag to the physical space, after reading its memory, and updating its password, the tag is now totally controlled by the attacker, and thus subject to tracking, similar to the situation in Section IV-C1. There is little that can be done to avert such an attack, except for physical security features that automatically destroy the tag upon physical tampering, for example. Such features are likely not cost-effective. In Section V, we mention ideas the back-end system could use to evaluate the likelihood of a tag having been compromised by attackers.

Note that we do not consider physical tag spoofing attacks, where a malicious tag, (masquerading as legitimate) is placed inside our physical space. This is a natural attack that follows after capturing a tag.

E. Physically Attacking Readers

Physically capturing readers is even more difficult than capturing tags. But we again offer a discussion in the same spirit as Section IV-D. Similar to tags, tag IDs do not at any time need to be stored in readers for our proposed scheme to function. The back-end system can make all decisions, and command readers to act accordingly. However, if a reader, or the back-end system are constrained in their communications with each other, it may be more efficient to have some tag ID memory in readers. Therefore, there is a privacy tradeoff. In the unlikely event that a reader is captured by an attacker, it could gain tag ID information, and thus, perform inventorying. As for using this attack to facilitate tracking, the ideas are similar to those in Section IV-D.

We do not consider physical reader spoofing attacks.

V. RFID Supporting Infrastructure

Using our proposed scheme as a foundation, a much richer privacy system can be explored. One aspect is offering the application a dynamic risk assessment of the system, which it can use to make real-time decisions. For example, the back-end system can keep histories of the time lengths between password updates, as well as the locations of tags. These are just two sets of temporal, and spatial data. Other data can also be easily collected, and processed to evaluate the risks of individual parts of the system, as well as the entire system as a whole. For example, a risk value can be dynamically assigned to tags, to indicate their likelihoods of having already been compromised by attackers. Readers then can interact with the tags, based on these tiered risk levels, depending on the requirements of the application.

Another aspect is optimizing the performance of our system. That is, given certain constraints, such as certain attack models, what is the maximum level of privacy our system can provide, and what are the algorithms to achieve this? The optimization can be global, with the back-end system operating

as the centralized controller. For example, what are the optimal ways to stream passwords? It seems sensible to first stream passwords of tags that were present in the previous singulation session, and then afterwards, try passwords of tags that have left nearby domains. However, it remains to be shown that this is indeed optimal. The optimization can also be local, which models a situation where readers have constrained access to the back-end system, and thus, do not have global knowledge of the system.

These ideas are largely related to the implementation of the back-end system, otherwise known as the supporting infrastructure of an RFID system. The research in the relatively young area of RFID has been primarily focused on the interactions between readers, and tags. The next frontier is understanding the supporting infrastructure, and in particular, how privacy can be implemented at this level.

VI. Conclusion

In this paper, we propose a novel privacy-protecting RFID system using *password streaming*. Our proposed scheme assumes a reader knows the presence of multiple tags, in its transmit range, and immediately starts streaming tag passwords, associated with tag IDs, in a broadcast fashion, allowing for tags to respond in between passwords. A centralized back-end system coordinates the password streaming of multiple readers in their respective domains. Our proposed scheme benefits from being efficient, and light-weight, which is achieved by pushing complexity to the back-end system. Yet, it still can defend against strong attack scenarios.

Future work includes exploring our proposed in scheme in detail. This includes investigating the various components of our system, and how they can be used to cooperatively defend against different types of attacks. As well, the role of RFID supporting infrastructure in protecting privacy is not immediately clear. We only touch on a small number of issues in Section V. These, and others require further research as well.

References

[1] A. Juels, "RFID security and privacy: A research survey," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 2, pp. 381–394, Feb. 2006.

[2] M. Langheinrich and R. Marti, "Practical minimalist cryptography for RFID privacy," *IEEE Systems Journal*, vol. 1, no. 2, pp. 115–128, Dec. 2007.

[3] K. P. Fishkin, S. Roy, and B. Jiang, "Some methods for privacy in RFID communication," in *European Workshop on Security in Ad-Hoc and Sensor Networks*, Heidelberg, Germany, Aug. 2004, pp. 42–53.

[4] A. Shamir, "How to share a secret," *Comm. of the ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[5] A. Juels, "Minimalist cryptography for low-cost RFID tags," in *Security in Communication Networks*, Amalfi, Italy, Sep. 2004, pp. 149–164.

[6] C. Law, K. Lee, and K.-Y. Siu, "Efficient memoryless protocol for tag identification," in *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*, Boston, MA, Aug. 2000, pp. 75–84.

[7] H. Vogt, "Multiple object identification with passive RFID tags," in *IEEE International Conference on Systems, Man and Cybernetics*, Hammamet, Tunisia, Oct. 2002.