Password Streaming for RFID Privacy

Victor K.Y. Wu and Roy H. Campbell

University of Illinois at Urbana-Champaign, IL, USA
{vwu3,rhc}@illinois.edu

Abstract. We propose a novel privacy-protecting RFID system using *password streaming*. Our scheme assumes a centralized back-end system containing a database of tag passwords. A reader uses this database to broadcast password "guesses" continuously in its domain, without it needing to scan for the presence of tags. A tag that receives a matching password tells the reader it is present, and the password is replaced with a new one that is synchronized both in the tag and the database. Our scheme also assumes multiple readers with their respective domains in a large physical space. As tags and attackers (adversaries intent on compromising privacy) move between domains, the back-end system coordinates the readers' password streams to defend against attacks. Our scheme pushes the computational complexity to the back-end system, very much in the spirit of RFID. It defends against stronger types of attacks than those in [4].

1 Introduction

In an RFID (radio frequency identification) system, RFID tags carrying unique tag IDs (identifiers) are affixed to 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. 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 to protect privacy [1]. Inventorying refers to an attacker acquiring a tag ID, and using it to learn information regarding its associated object. Tracking refers to an attacker physically (or otherwise) following a tag by successively querying it, also to learn information regarding the associated object. Tracking is still possible if a tag responds with a meta identifier instead of its tag ID.

In this paper, we seek a solution to the privacy problem. Since RFID tags (especially passive tags that 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 password-specific 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. 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 mention current solutions in the literature that steer clear of this problem. We then offer a novel solution and discuss its advantages.

The authors in [2] use a password-less system based on *secret sharing* [3]. In a (t, n) sharing scheme, a secret is divided into n shares, which are 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. A tag ID is first divided into its secret shares. The shares are concatenated, and the resulting string replaces the tag ID in 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 number of successive queries due to throttling, an assumption of the weakened attack model of [4]. This scheme is indeed strong. Nonetheless, if we do allow an attacker to query a tag for a long and undisturbed period of time, the attacker does finally acquire the tag ID. The weakened attack model may allow a reader and a tag occasionally to 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 this scheme is not necessarily effective in stronger attack scenarios.

The authors in [4] 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 reader query 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 Sect. 4.1. 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 from being inefficient. 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 has a good approximation (based on information provided by a centralized back-end system) of the tags in its domain, and immediately starts streaming the passwords of those tags, in a broadcast fashion, allowing for the tags to respond in between passwords. A tag that receives a matching password updates its local copy by randomly generating a new password to replace the existing one, and sends back the update to the reader. The reader thus confirms which tag is present in its domain. The back-end system coordinates the password streams of multiple readers in their respective domains. Thus, tag passwords and 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 where the tag has moved. This prevents an attacker to use the stale password to compromise privacy. In particular, the main contributions of this paper include the decision criteria for and choice of the passwords that are streamed.

Note that an immediate advantage of our scheme is that it is a singulation algorithm, although very different from the traditional tree-walking [5] and Aloha-based [6] algorithms. That is, we do not require privacy to be a separate communications layer. Our proposed scheme is efficient and light-weight. In Sects. 3 and 4, we point out the low computational requirements of 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, and therefore push computational requirements to the centralized back-end system. This structure allows our proposed scheme to defend against stronger types of attacks than those in [4].

The rest of the paper is organized as follows. Section 2 sets up the system model. Section 3 gives a generalized description of our proposed scheme. In Sect. 4, we refine our scheme to defend against increasingly stronger attacks. Section 5 describes how the supporting infrastructure of an RFID system can be explored in relation to privacy. Finally, Sect. 6 concludes the paper, and offers future research directions.

2 System Model

We define the components in our privacy system.

- **tag**: A tag refers to a legitimate passive RFID tag affixed 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.
- **back-end system**: The back-end system is the abstract upper layer that sits above the physical and data link layers of the tags and readers. It coordinates how readers 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 or 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 eventually 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.

3 Proposed Scheme

We describe our proposed scheme, in a purposely generalized notion. In Sect. 4, we provide specific details according to increasingly stronger attacker models and requirements of the application.

Every tag in the physical space has a unique and dynamic password stored in its memory. As well, the passwords are mirrored in a database in the back-end system, each associated with a tag's ID. That is, tag passwords are synchronized between the back-end system and tags. Our proposed scheme is actually a singulation algorithm for the readers described in Sect. 2. In each domain, the reader streams tag passwords (chosen from the back-end system) continuously, in a broadcast fashion. Time is allowed between each streamed password for tags to respond. When a tag receives a password that does not match its own password, it responds with a common (across all tags) and pre-determined dummy signal. When a tag does receive a matching password, it immediately updates its local copy by randomly generating a new password to replace the existing one, and sends back the update 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. Therefore, when a streamed password does not match, the reader receives a summation of dummy signals due to multiple responses from multiple tags. The reader receiver is designed to handle multi-path propagation at the physical and data link layers. This allows the reader to interpret the identical dummy signals as multi-path copies of a single dummy signal, and it can thus conclude that the tag associated with the password it had streamed is not present. When a streamed password does match, the reader interprets the responses as the summation of two signals. One is a dummy signal experiencing multi-path propagation while the other is a password update signal. Therefore, the reader can subtract out the dummy signal from the summation to extract the updated password, and pass on the update to the back-end system to maintain synchronization. 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.

The main contributions of this paper include the decision criteria for and choice of the passwords that are streamed. Since the password space is large, the solution is not immediate. For example, consider the reader first choosing passwords of tags that have been singulated in the previous session. Assuming that tags are not highly mobile, most of the tags in the domain are singulated. Next, the reader can try passwords of tags that have left nearby domains, hoping that they have moved to this domain. This is only one example of what the back-end system can do. In other words, the back-end system collects information at the reader-tag level, fuses it to make system-level decisions, and implements the decisions by telling the readers which passwords to stream. In Sect. 4, we systematically consider increasingly strong attack scenarios, and how passwords can be chosen accordingly to defend against these attacks.

4 Attacks and Defenses

We vary our attack model by increasingly strengthening the attackers. In each case, we make specifications and/or modifications to our generalized scheme in Sect. 3 to defend against the progressively stronger attacks. Our defenses are focused on protecting against inventorying and tracking. We also consider how privacy can be traded off for efficiency in some of these cases, as determined by the application.

4.1 Eavesdropping

Eavesdropping occurs when an attacker listens to 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.

One-way Eavesdropping. Since tags are passive, readers broadcast at greater 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 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. Nonetheless, Sect. 2 assumes our system model has minimal communications and physical security that makes this unlikely to succeed.

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 attack model in [4]), 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 Sect. 3, by using the exclusive or (XOR) one-time pad algorithm in [4]. 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 $\tilde{\Delta}$, and the back-end system updates 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.

4.2 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 responses of the tag and attacker quickly allow 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. 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 associated 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 for 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. This idea of recycling passwords deserves further analysis.

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

4.3 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 Sect. 4.1, attackers are unlikely to acquire passwords. As such, tags only respond with dummy signals to attacker queries, and therefore, no information is leaked. Nonetheless, suppose that the minimal level of communications and physical security mentioned in Sect. 2 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 Sect. 4.1.

Replaying Single Passwords. A tag is in great danger once an attacker acquires its password (through whatever means). That is, the attacker can immediately send the password to the tag, and the tag responds 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 [2], [4]. 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, a simple strategy is for it to query the tag with its password immediately after it wakes up. As a defense, the back-end system can also schedule the reader to transmit that password at that same time. This does not prevent attacker-tag synchronization. But it does prevent reader-tag de-syncronization. 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. Nonetheless, the attack provided here is rather strong, since acquiring both the password and location of a tag is extremely difficult, as explained in Sect. 4.1. In the following, we provide a more realistic reader spoofing attack.

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 a 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 in that short time). Passwords that had matched are no longer valid. Therefore, this attack is rather weak. An alternative is that the attacker replay the stream in another domain (by sending the stream to another attacker, or by the attacker moving to the other domain). The idea is that non-matching passwords in the original stream might belong to tags in other domains. This attack, however, is again rather weak. For example, the readers can dynamically keep track of where tags are with the help of the back-end system. As long as tags are not highly mobile, a reader only potentially streams a non-matching password when a tag has left its domain (and has not yet been singulated in another domain), or when 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 lengthens 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 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. But the passwords from this stream are useless in other domains. If the stream is replayed in the same domain, then we are back to the situation of replaying single passwords.

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).

4.4 Physically Attacking Tags or Readers

We discuss the possibility of attackers physically capturing tags and readers (temporarily or permanently), and thus having unfettered access to their memories. 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.

Our proposed scheme can still function if tag IDs are absent in tags and readers. If we implement our scheme in such a way, inventorying is unsuccessful even if tags or readers are captured. Nonetheless, if a reader and 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 gains tag ID information, and can perform inventorying. Conversely, tracking is readily achieved. An attacker can capture a tag, password-synchronize with it, and then replace it in the physical space. The tag is now totally controlled by the attacker to perform tracking.

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

5 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 an attack model, 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. 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.

6 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 defends against strong attack scenarios.

Future work includes exploring our proposed 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, and deserves further work.

References

1. Juels, A.: RFID security and privacy: A research survey. IEEE J. Sel. Areas Commun. 24, 381–394 (2006)
2. Langheinrich, M., Marti, R.: Practical minimalist cryptography for RFID privacy. IEEE Systems Journal 1, 115–128 (2007)
3. Shamir, A.: How to share a secret. Comm. of the ACM 22, 612–613 (1979)
4. Juels, A.: Minimalist cryptography for low-cost RFID tags. In: Security in Communication Networks, Amalfi, Italy, pp. 149–164 (September 2004)
5. Law, C., Lee, K., Siu, K.-Y.: 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, August 2000, pp. 75–84 (2000)
6. Vogt, H.: Multiple object identification with passive RFID tags. In: International Conference on Systems, Man and Cybernetics, Hammamet, Tunisia (October 2002)