A Survey of RFID Authentication Protocols

Drew Scott

1The University of Arizona

April 03, 2024

Abstract

“Radio frequency identification” (RFID) systems are ubiquitous in today’s world. In an RFID system, it is a desirable to attain mutual authentication between a reader and a tag before commencing application-level communications. This is because tags should not share secret information with unknown parties and readers need to defend against tag impersonation. Authentication protocols designed for communication between computers, however, are not appropriate for RFID systems because tags are extremely resource constrained (low energy, small memory, etc.). Thus, there have been many attempts to design secure and practical authentication protocols for RFID systems over the years since RFID systems became prevalent. This survey summarizes and compares these protocols.
A Survey of RFID Authentication Protocols

Drew Scott
The University of Arizona
Submission Date: May 3, 2023

Abstract—"Radio frequency identification" (RFID) systems are ubiquitous in today's world. In an RFID system, it is a desirable to attain mutual authentication between a reader and a tag before commencing application-level communications. This is because tags should not share secret information with unknown parties and readers need to defend against tag impersonation. Authentication protocols designed for communication between computers, however, are not appropriate for RFID systems because tags are extremely resource constrained (low energy, small memory, etc.). Thus, there have been many attempts to design secure and practical authentication protocols for RFID systems over the years since RFID systems became prevalent. This survey summarizes and compares these protocols.

I. INTRODUCTION

This survey presents the RFID authentication protocol research landscape. Before this discussion, Section I and Section II introduce the fundamental properties and security concerns of RFID systems.

A. Background

A "radio frequency identification" (RFID) system involves "tagging" physical items with an RFID chip, which stores data that can be accessed by an RFID reader via radio transmission. The RFID reader has access to a database which stores information about each tagged item. Such a system is comparable to barcode-based systems: each item has a barcode that can be scanned by a barcode reader, which can look up information about the item in a database. The primary advantage of an RFID system over a barcode system is that when an RFID reader scans an item, the item sends a radio signal back to the reader. Reading a radio signal doesn't require a clear view or a precise placement of the reader, which reading a barcode does. Additionally, RFID tags can store more data than barcodes and RFID readers can read many RFID tags at once [1].

There are a wide variety of use cases for RFID systems in the real world. For examples [1]:

- Supply chain management (i.e. tracking the delivery/sale of goods)
- Access control (e.g. entering a private building or public transit station)
- Movement tracking (e.g. tracking newborn babies in hospitals)

RFID tags are significantly resource constrained. Typically, RFID tags only contain about 512 bits to ~8000 bits of memory [2]. An RFID tag can be attached to a power source or a battery (called an "active tag"), or get its power from the reader via electromagnetic waves (called a "passive tag"). Active tags, therefore, can broadcast their own signal, while passive tags can only transmit data when activated by a reader. Active tags can be read from hundreds of feet away, while passive tags can only be read from several feet away [3]. RFID tags have the ability to do limited computation.

The rest of this survey is outlined as follows: Section II introduces security concerns of RFID systems, Section III introduces existing protocols and analyzes their ability to address security concerns, Section IV summarizes the big picture of existing research, and Section V discusses future work in RFID security protocols.

II. RESEARCH PROBLEMS

The aforementioned RFID system use cases introduce a variety of security and privacy concerns. For examples:

- Shipments containing RFID tagged items may be able to be read by an unauthorized actor. For example, a thief could learn if there are valuable items in a truck by scanning it with an RFID reader.
- A person who carries an RFID tag could be tracked by an unauthorized entity. For example, if the person carries an RFID-tagged public transit card in their wallet, they may worry about being tracked by hidden RFID readers as the move through the physical world.
- Transmitted RFID signals can be read by anyone, as long as they have a reader close by. For example, (1) if an RFID tag is used to enter an apartment building, an unauthorized reader could be placed next to the real reader. Then, the unauthorized reader could save transmissions and repeat them to gain unauthorized access to the building later. (2) if secret information is transmitted, it can be stolen by an unauthorized reader.

Therefore, RFID communication protocols must attempt to address these concerns while remaining practical for resource-constrained RFID tags (i.e. small memory, limited power consumption, and limited computation). Desirable properties of such protocols include:

- Mutual authentication: (1) the tag should not send data to an unauthorized reader and (2) the reader should not accept a signal from an unauthorized tag.
- Encryption: to prevent the loss of secret information.
- Resilience to replay attacks (without infringing on the ability to reuse the RFID tag itself): to prevent against reusing transmissions to authenticate at access control points.

This survey focuses on authentication protocols used in communication between RFID tags and readers.
III. AUTHENTICATION PROTOCOLS

This survey considers 7 authentication protocols in two groups ("classical" and "modern" protocols). Classical protocols include the early authentication protocols, and modern protocols include protocols that made improvements on the classical protocols.

A. Classical Protocols

"Classical protocols" establish the fundamental techniques and paradigms for RFID authentication protocols. Each of these protocols experiment with at least one of the following properties:

- Practicality vs. security tradeoff
- Computation vs. memory intensive design
- Key establishment
- Mutual vs. one-way authentication

A discussion of 4 of these protocols follows.

1) Vajda and Buttyán (2003): In 2003, Vajda and Buttyán [4] proposed several protocols to authenticate RFID tags. Their protocols avoid the use of cryptographic primitives like encryption, MACs, and hash functions because the authors assume that they are too costly to compute on RFID tags. Instead, the protocols use more simple operations like XOR, permutation, addition, and multiplication. While the aforementioned cryptographic primitives are normally composed of these operations, the authors’ protocols do not make any cryptographic guarantees. Instead, these protocols offer a system designer to tune their system according to a security/performance tradeoff.

For example, Vajda and Buttyán’s first proposed protocol is:

(1) Reader \( \rightarrow \) Tag : \( x \oplus k_1 \)

(2) Tag \( \rightarrow \) Reader : \( x \oplus k_2 \)

Here, \( k_1 \) and \( k_2 \) are shared keys between the reader and the tag. The tag authenticates by proving it has knowledge of these keys by "decrypting" the challenge using \( k_1 \) sent by the reader, then "decrypting" \( x \) using \( k_2 \). As long as \( k_2 \) is only known by this specific tag, then the reader knows which tag it is talking to. But, this protocol suffers from a key establishment issue since keys cannot be securely reused (as in the "one-time pad" encryption algorithm). To solve this problem, the authors suggest a permutation algorithm to generate \( k_1 \) for each session; \( k_1 \) for the ith session is based on \( k_1 \) from the i-1th session. \( k_2 \) is held constant for all sessions. To generate \( k_1 \) for the first session, use \( k_2 \) as input for the permutation algorithm. So, the goal of the attacker is to generate \( k_2 \) in order to impersonate the tag. The authors note that computing \( k_2 \) may be feasible due to information leakage during each session. As such, they argue there is a security/performance tradeoff with the size of the session key.

2) Feldhofer (2004): In 2004, Feldhofer [5] proposed a challenge-response authentication protocol for tags, whereby the RFID reader sends the tag a random number (challenge) to encrypt using a shared secret key using the AES algorithm. The tag then transmits the ciphertext back to the reader for verification. Clearly, the ability of implement AES on an RFID tag contradicts the assumptions made in Vajda and Buttyán’s work [4]. The author implements a "least-cost version" of AES, which uses an 8-bit architecture and is able to encrypt 128 bits in 850 clock cycles, meaning that 30 RFID tags can be authenticated in 1 second. But, Weis, et. al. would argue that this protocol is not appropriate on two bases: (1) 100-200 RFID tags need to be authenticated each second, not 30, and (2) that a hardware implementation of encryption protocols (including those more simple than AES) requires RFID tags that cost 0.5-1 USD each, while a 0.05-0.10 USD price point is necessary to enable widespread RFID adoption [6]. Thus, while Feldhofer implemented a lightweight AES algorithm, its practicality is certainly under question.

3) Weis, et. al. (2004): In 2004, Weis, et. al. [6] proposed a locking/unlocking protocol for tags to authenticate a reader. This is in contrast to the previously mentioned protocols which authenticate the tag, not the reader. In a locked state, a tag does not transmit any valuable information. Tags wait until the reader has authenticated itself before "unlocking" to share this valuable information. In this system, each tag stores a "metaID", which is the hash of a unique key the reader associates with a tag (metaID := hash(key)). Additionally, the reader stores a mapping for each metaID and key pair. The unlocking protocol is:

(1) Reader \( \rightarrow \) Tag : Unlock request

(2) Tag \( \rightarrow \) Reader : metaID := hash(key)

(3) Reader : Lookup metaID to find key

(4) Reader \( \rightarrow \) Tag : key

(5) Tag : Verify metaID = hash(key) to unlock

Due to the one-way property of a hash function, the pre-image "key" cannot be generated by simply knowing the image "metaID." So, the tag authenticates the reader. This protocol’s strength is, therefore, fully dependent on the efficacy of the selected hash function. Vajda and Buttyán argue, however, that cryptographic hash functions such as MD5 and SHA-1 are not able to be implemented on RFID tags [4]. As such, there is a practicality/security tradeoff when selecting a hash function to use in this protocol.

4) Juels (2005): In 2005, Juels [7] proposed a protocol for mutual authentication between an RFID tag and an RFID reader, which uses a method called "pseudonym throttling." In this system, an RFID tag is preloaded with a list of pseudonyms. Each pseudonym is associated with a "secret key" and an "authentication key"; that is, each tag stores a list of 3-tuples: (pseudonym, secret key, authentication key). Readers store these lists for all tags. In this protocol, the secret
key is used to authenticate the reader and the authentication key is used to authenticate the tag.

The mutual authentication protocol is:

1. Reader → Tag : Read request
2. Tag → Reader : Next pseudonym \( p_i \) in list
3. Reader : Lookup corresponding secret key \( s_i \) for \( p_i \)
4. Reader → Tag : \( s_i \)
5. Tag : Verify \( s_i \) matches \( p_i \)
6. Tag : Lookup corresponding authentication key \( a_i \) for \( p_i \)
7. Tag → Reader : \( a_i \)
8. Reader : Verify \( a_i \) matches \( p_i \)

Once the pseudonym list (of length \( k \)) has been exhausted, the tag will start back at the beginning of the list. To prevent recycling pseudonyms, a reader may update a tag with new pseudonyms at any point (i.e. whether the list has been exhausted or not). For each data item (each pseudonym, each secret key, and each authentication key), the tag stores a list of pads each with length \( m \). The pseudonym list can only be updated following the mutual authentication protocol. This pseudonym list update protocol is:

1. Reader → Tag : Pad update matrix
   
   (3 × \( k \) sublists, each with \( m \) elements)
2. Tag : For each pad list \( l \), set \( l[i] = l[i+1] \), set \( l[k] = 0 \)
3. Tag : Update each pad list to be the element-wise XOR with itself and its corresponding list in the pad update matrix
4. Tag : Update each item by computing bit-wise XOR with itself and the first element in its corresponding pad list

This protocol’s security is based on the difficulty of an adversary guessing/knowing the next (pseudonym, secret key, authentication key) tuple. If an adversary knows this information, they will be able to impersonate both the reader and the tag. So, this difficulty is based on the length of the pseudonym list and the length of the pad lists. Once the \( k \) elements of the pseudonym list have been exhausted, an adversary will be able to calculate the 3-tuple if they have observed the last \( m \) padding updates. Therefore, this system’s security is tightly coupled with the amount of memory available on the RFID tag.

To limit the intense memory usage of this protocol, the authors propose several simplifications to their protocol which directly trade off memory with security. These simplifications include:

- Using small secret keys and authentication keys (as few as 20 bits)
- Using small pseudonyms (as few as 100 bits)
- Using small pad lists (as few as 0 or 1 elements)
- Using small pseudonyms list (as few as 4 or 5 elements)

### B. Modern Protocols

While "classical protocols" established the fundamental techniques and paradigms used to design RFID authentication protocols, "modern protocols," in general, make technical improvements to improve classical protocols. These improvements include:

- Reducing memory usage
- Reducing computation cost
- Improving security

A discussion of 3 of these protocols follows.

1. Chien (2007): In 2007, Chien [8] proposed a mutual authentication protocol using a pseudonym transmission method, similar to what Juels [7] developed. Each tag is associated with a 4-tuple (ID, pseudonym, K1, K2), which both the tag and reader knows. Over time, this tuple is changed, so the tag tracks its current and previous tuples. The protocol is:

   1. Reader → Tag : Read request
   2. Tag → Reader : Pseudonym
3. Reader : Lookup tuple corresponding to pseudonym
4. Reader : Generate randoms \( n_1 \) and \( n_2 \)
5. Reader : Compute \( A := \text{pseudonym} \oplus K_1 \oplus n_1 \)
6. Reader : Compute \( B := \text{pseudonym} \lor K_2 + n_2 \)
7. Reader : Compute \( K_1' = \text{rotateleft}(K_1 \oplus n_2, \text{K1}) \)
8. Reader : Compute \( C = (K_1 \oplus K_2') + (K_2 \oplus K_1') \)
9. Reader → Tag : A, B, C
10. Tag : Extract \( n_1, n_2 \) from A and B, compute K1', K2'
11. Tag : Compute \( C' = (K_1 \oplus K_2') + (K_2 \oplus K_1') \)
12. Tag : Verify \( C = C' \)
13. Tag : Compute \( D = (K_2' + \text{ID}) \oplus ((K_1 \oplus K_2) \lor \text{K1}') \)
14. Tag → Reader : D
15. Tag → Reader : Verify D

Then, the reader and tag update the (ID, pseudonym, K1, K2) tuple as follows:

- \( \text{ID} := \text{ID} \)
- \( \text{pseudonym} := (\text{pseudonym} + \text{ID}) \oplus (n_2 \oplus \text{K1}') \)
- \( \text{K1} := \text{K1}' \)
- \( \text{K2} := \text{K2}' \)

In this protocol, the tag authenticates the reader by verifying the computation of \( C \) and the reader authenticates the tag by verifying the computation of \( D \). The tag trusts the reader after verifying \( C \) because its computation requires knowledge of K1 and K2 which correspond to its pseudonym–only the reader and the tag knows these. Similarly, the reader trusts the tag after verifying \( D \) because it requires knowledge of ID, K1, K2 which correspond to a valid pseudonym–only the
reader and the tag knows these. The secrecy of these 4-tuples, however, is dependent on the security of the update protocol. In other words, these assumptions only hold if it is difficult to extract $n_1$ and $n_2$ from $A$ and $B$. Additionally, if $K_1$ and $K_2$ are ever revealed, all future 4-tuples could be computed by an adversary. The author argues that the protocol adequately protects these values.

Although this is a pseudonym protocol like Juels’ [7], this protocol does not consume as much memory to update pseudonym data. Even better, this protocol doesn’t make this improvement by requiring computationally intensive operations. Thus, this protocol demonstrates an opportunity to work outside of the computationally expensive vs. memory intensive design paradigm seen in the “classical protocols.”

2) Aghili, et. al. (2019): In 2019, Aghili, et. al. [9] proposed a mutual authentication protocol that improves upon a mutual authentication protocol from Fan, et. al. [10]. The improvements made by Aghili, et. al. include:

- Reduces communication flow between the reader and tag to improve efficiency.
- Simplifies the information stored on the backend database.
- Uses random numbers to improve resistance to a variety of attacks, including replay and impersonation attacks.
- Develops a new function, called "MRot" (a keyed, modular rotation function), which is lightweight and more secure than other rotation functions.

The most technically interesting contribution of this work is the development of the "MRot" function. The authors developed MRot to replace the "cro" (cross rotate) function created by Fan, et. al. The purpose of these functions is to provide an efficient, yet secure, "encryption" algorithm to protect the transmission of secret data between the reader and the tag. Both functions take two inputs, $x$ and $y$. Unfortunately, $cro(x,y)$ and $x$ (or $y$), they can compute $y$ (or $x$). This presents a clear vulnerability in the case that $cro(x,y)$ and $x$ (or $y$) are public and $y$ (or $x$) is private. MRot, however, does not leak this information because it is keyed (i.e. on top of using inputs $x$ and $y$, there is another input $K$). The authors prove that, on average, each input $(x,y)$ pair can be mapped to every possible output value. Thus, knowledge of both $MRot(x,y,K)$ and $x$ (or $y$) does not reveal any information about $y$ (or $x$).

3) Izza, et. al. (2021): In 2021, Izza, et. al. [11] proposed a mutual authentication protocol using cryptographically secure primitives, including elliptic curve cryptography (ECC) and one-way hashing. Classical works from both Feldhofer [5] and Weis, et. al. [6] used cryptographically secure primitives, like symmetric key encryption (AES) and one-way hashing, too. These works, however, were criticized for being impractical to implement on cheap RFID tags. The work of Izza, et. al., however, is geared toward wireless body area networks (WBAN) for healthcare applications. In the healthcare field, rigorous cryptographic guarantees for encryption, privacy, and integrity are necessary. As this work demonstrates, the long held assumption of requiring authentication protocols to be designed for the cheapest RFID tags is no longer always the case. In other words, there are real world applications that create a market for more expensive RFID tags. Thus, it is necessary to design authentication protocols for these applications.

IV. CONCLUSION

Table I summarizes the key properties of each of the authentication protocols discussed in Section III. The properties of these protocols demonstrate the tradeoff between practicality and security for protocol design. Over time, the design of RFID authentication protocols became more robust:

- Early on, protocols only implemented one-way authentication. Later, mutual authentication became ubiquitous.
- Early on, protocols made a tradeoff between computation and memory intensive design. Later, protocols were able to break out of this dichotomy.
- Early on, cryptographically secure protocols were deemed impractical for all applications. Later, applications were developed that required cryptographically secure protocols.

V. FUTURE WORK

In the future, I would expect for even more applications that require cryptographically secure protocols to become popular. Right now, these applications appear to be generally confined to the medical field, but they may expand into banking/payment, ticketing, and elsewhere. In preparation for the development of these applications, I would suggest future work to be concentrated on creating new or improving existing cryptographically secure protocols.

Additionally, there have been several works to "patch" shortcomings in published protocols. With such glaring security issues, it seems prudent to create a standardized framework to analyze the security level of newly designed protocols to prevent weak protocols from being published and adopted. Such a framework would require standards for both analyzing the strengths and weaknesses of encryption algorithms and for analyzing the vulnerabilities of a protocol’s flow to attacks like replay, impersonation, etc.

REFERENCES

<table>
<thead>
<tr>
<th>Protocol Name</th>
<th>Authenticated</th>
<th>Mechanism</th>
<th>Practical</th>
<th>Resource</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Juels (2005) [7]</td>
<td>Both</td>
<td>Pseudonym verification</td>
<td>Yes</td>
<td>Memory</td>
<td>Tunable</td>
</tr>
<tr>
<td>Chien (2007) [8]</td>
<td>Both</td>
<td>Pseudonym verification</td>
<td>Yes</td>
<td>Memory</td>
<td>Tunable</td>
</tr>
<tr>
<td>Aghili, et. al. (2019) [9]</td>
<td>Both</td>
<td>Pseudonym verification</td>
<td>Yes</td>
<td>None</td>
<td>Tunable</td>
</tr>
<tr>
<td>Izza, et. al. (2021) [11]</td>
<td>Both</td>
<td>Public key encryption (ECC) and hashing</td>
<td>No</td>
<td>Compute</td>
<td>Cryptographic</td>
</tr>
</tbody>
</table>

**TABLE I**

Summary of all surveyed authentication protocols and their properties.

LEGEND: "Protocol Name" describes the author(s) and proposal year of the protocol, "Authenticated" indicates which entity is authenticated in the protocol, "Mechanism" describes the key algorithmic mechanism of the protocol, "Practical" indicates if the protocol can be practically implemented on cheap RFID tags, "Resource" indicates the limiting resource on the RFID tag for the protocol, and "Security" indicates the protocol’s security properties.


