Security and Privacy in Large-Scale RFID Systems

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Kazuya Sakai, M.S.

Graduate Program in Department of Computer Science and Engineering

The Ohio State University

2013

Dissertation Committee:

Ten H. Lai, Advisor

Dong Xuan

Feng Qin
Radio Frequency Identification (RFID) is an electronic tagging technology that allows objects to be automatically identified at a distance without a direct line-of-sight using an electromagnetic challenge-and-response exchange of data. An RFID system consists of RF readers and RF tags. RF tags are attached to objects, and used as a unique identifier of the objects. RFID technologies enable a number of business and personal applications, and smooth the way for physical transactions in the real world, such as supply chain management, transportation payment, animal identification, warehouse operations, and more. Though bringing great productivity gains, RFID systems may cause new security and privacy threats to individuals or organizations, which have become a major obstacle for their wide adaptions. Therefore, it is important to address the security and privacy issues in RFID systems.

In this dissertation, we investigate security and privacy issues for large-scale RFID systems. Since any object is uniquely identifiable with an RF tag, the tag’s ID must be protected from adversaries during data communications in keeping with the authenticity of tags. Hence, we first propose private authentication protocols that RF readers to singulate individual tags without disclosing tags’ content to adversaries. To design a secure access protocol, two different approaches are taken, encryption-based and non-encryption-based. In the encryption-based approach, we propose a structured key management with low cost cryptographic operations based on a skip
This can be applied to a large-scale RFID systems. On the other hand, shared key exchanges are not feasible in some contexts. Hence, we develop a distributed RFID architecture for secure data communications without shared secret. With a novel encoding scheme and jamming technique, the distributed RFID authentication scheme protects tags from various types of adversaries.

With a private authentication protocol, readers can securely validate tags’ authenticity. After reading a tag, an RFID system updates object’s status or generates data. Thus, any piece of data in the back-end server is associated with a particular tag. For a high quality RFID-based data service, the authenticity of data is of concern. Therefore, we study the verifiable RFID systems, where a set of data related to a tag can be verified in the sense that the data is associated with the tag and any element of the data cannot be modified without being detected. To realize such a verifiable RFID system, we build a new RFID architecture that integrates multiple RFID systems into single exa-scale RFID system, then formulate data verification problem, and then propose data verification protocols.

The proposed solutions are mathematically analyzed, and computer simulations are conducted to measure all aspects of the RFID systems, including the degree of security and the cost of control overhead. Furthermore, we implement a prototype of a verifiable RFID system. The performance evaluations show that the proposed protocols achieve their design goals. We believe this research serves the foundation for the next generation of RFID systems.
This is dedicated to my family for their love, support, and trust.
Acknowledgments

It would have been impossible for me to finish my Ph.D. study without the support, help and guidance I have received from the professors, colleagues, friends, and my family.

I wish to thank my advisor, Professor Ten-Hwang Lai. I learned from him not only how to conduct top-level research work, but also how to improve myself in every aspect. He is one of the smartest people that I have met in my life. Discussing technical problems with Professor Lai is always meaningful and enjoyable. I learned from him how to find a good research agenda, how to define research problems, how to develop a novel solution, and how to present complicated research results with simple words. My research ability drastically increased under his advisory.

I am also grateful to Professor Dong Xuan and Professor Feng Qin for serving on my candidacy committee and dissertation reading committee. The Ohio State University is an excellent university with world class faculties. I am thankful to all faculties who taught me. With their help, I could broaden my view in computer science and teaching subjects.

There were great helps from people outside of The Ohio State University. I would like to express my gratitude to Professor Min-Te Sun at National Central University and Professor Wei-Shinn Ku at Auburn University for their help for my research. They are always supportive and encourage my Ph.D. study.
Additionally, I would like to show my great appreciation to my mother for her support during my graduate study. Without her, I would not be able to pursue my Ph.D. degree in the United States.
Vita

October 23, 1981 ...................... Born - Osaka, Japan

2004 ............................. B.S. Electronics

2007 ............................. M.S. Electronics

2010 ............................. M.S. Computer Science

2010-present ..................... Ph.D. Candidate,
                                  The Ohio State University.

Publications

Research Publications

Kazuya Sakai, Wei-Shinn Ku, Min-Te Sun, and Roger Zimmermann “Privacy Preserving Continuous Multimedia Streaming in MANETs”. *ACM Transactions on Multimedia Computing Communications and Applications*, vol. 9, issue 4, article no. 24, 2013.


**Fields of Study**

Major Field: Computer Science and Engineering

# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>Vita</td>
<td>vii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xiii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xiv</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 RFID Technologies</td>
<td>1</td>
</tr>
<tr>
<td>1.1.2 RFID Applications</td>
<td>3</td>
</tr>
<tr>
<td>1.1.3 RFID Communication Protocols</td>
<td>4</td>
</tr>
<tr>
<td>1.1.4 RFID Standard</td>
<td>6</td>
</tr>
<tr>
<td>1.2 Research Issues in RFID Systems</td>
<td>7</td>
</tr>
<tr>
<td>1.2.1 Cardinality Estimation</td>
<td>7</td>
</tr>
<tr>
<td>1.2.2 Improving Singulation Protocols</td>
<td>8</td>
</tr>
<tr>
<td>1.2.3 Handling Multiple Readers</td>
<td>8</td>
</tr>
<tr>
<td>1.2.4 Missing Tag Problems</td>
<td>9</td>
</tr>
<tr>
<td>1.2.5 The Optimal Tag Deployments</td>
<td>9</td>
</tr>
<tr>
<td>1.2.6 Private Tag Authentication</td>
<td>10</td>
</tr>
<tr>
<td>1.2.7 Data Verification</td>
<td>11</td>
</tr>
<tr>
<td>1.3 Contribution of This Dissertation</td>
<td>11</td>
</tr>
<tr>
<td>1.4 Organization of This Dissertation</td>
<td>13</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Definition of notations.</td>
<td>23</td>
</tr>
<tr>
<td>3.1 Definition of notations.</td>
<td>58</td>
</tr>
<tr>
<td>3.2 Coding rule for the 1-to-4 RFRJ coding scheme.</td>
<td>64</td>
</tr>
<tr>
<td>3.3 An example of the 2-to-6 coding.</td>
<td>68</td>
</tr>
<tr>
<td>4.1 Definition of notations.</td>
<td>90</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>An example of the Aloha-based protocol.</td>
<td>5</td>
</tr>
<tr>
<td>1.2</td>
<td>An example of the tree-walking protocol.</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>Tree-based.</td>
<td>18</td>
</tr>
<tr>
<td>2.2</td>
<td>Group-based.</td>
<td>20</td>
</tr>
<tr>
<td>2.3</td>
<td>An example of skip lists.</td>
<td>21</td>
</tr>
<tr>
<td>2.4</td>
<td>An example of key issuing.</td>
<td>26</td>
</tr>
<tr>
<td>2.5</td>
<td>An example of authentication.</td>
<td>30</td>
</tr>
<tr>
<td>2.6</td>
<td>Example of unlinkability (Tag 3).</td>
<td>36</td>
</tr>
<tr>
<td>2.7</td>
<td>Example of unlinkability (Tag 5).</td>
<td>37</td>
</tr>
<tr>
<td>2.8</td>
<td>The binary tree corresponding to Figure 2.6and 2.7</td>
<td>38</td>
</tr>
<tr>
<td>2.9</td>
<td>System anonymity.</td>
<td>43</td>
</tr>
<tr>
<td>2.10</td>
<td>Authentication speed.</td>
<td>43</td>
</tr>
<tr>
<td>2.11</td>
<td>Storage cost.</td>
<td>43</td>
</tr>
<tr>
<td>2.12</td>
<td>System anonymity.</td>
<td>43</td>
</tr>
<tr>
<td>2.13</td>
<td>Anonymity with different $k$ values.</td>
<td>44</td>
</tr>
</tbody>
</table>
2.14 Performance with different $k$ values. ........................................ 44
3.1 Distributed RFID systems. .......................................................... 50
3.2 Distributed RFID system deployment. ............................................ 51
3.3 The proposed RFID architecture. .................................................... 55
3.4 The system model and basic idea. .................................................... 60
3.5 The 2 and 3-dimensional hypercube. .............................................. 71
3.6 Anonymity. ................................................................................. 78
3.7 Correct guess probability. ............................................................... 78
3.8 Time to crack tag data. ................................................................. 79
3.9 Correlation attacks $p_j=1.0$. ......................................................... 79
3.10 Correlation attacks $p_j=0.9$. ....................................................... 80
3.11 Correlation attacks $p_j=0.8$. ....................................................... 80
3.12 Anonymity under different $p_j$ values. ......................................... 81
3.13 Anonymity for correlation attacks. ............................................... 81
4.1 The system model. ........................................................................ 87
4.2 Average key storage cost for each client. ........................................ 101
4.3 Average key storage cost for each SP. ............................................. 102
4.4 Average key storage cost for different number of tags. .................... 103
4.5 Valid data rate for different illegal access rate. ............................... 104
4.6 Valid data rate for different illegal access rate. ............................... 104
4.7 Valid data rate for different data modification rate. ......................... 105
4.8 Valid data rate for different percentage of malicious SPs. . . . . . . . 105
4.9 Computational cost for different number of data accesses. . . . . . . 106
4.10 Computational cost for different number of data accesses. . . . . . 106
4.11 The program modules. . . . . . . . . . . . . . . . . . . . . . . . . . . 107
4.12 Turn around time with the loop-back setting. . . . . . . . . . . . . 108
4.13 Response time with LAN and international settings. . . . . . . . . . 108
Chapter 1: Introduction

1.1 Background

Radio Frequency Identification (RFID) is an electronic tagging technology that allows objects to be automatically identified at a distance without a direct line-of-sight using an electromagnetic challenge-and-response exchange of data. An RFID system consists of RF readers and RF tags. An RF tag is attached to an object and used as the unique identifier of the object. RFID systems smooth the way of various physical transactions in the real world.

In this chapter, we first provide an overview of RFID systems. Section 1.1.1 provides the features and configurations of RFID systems. Section 1.1.2 shows existing and potential applications of RFID technologies. RFID communication protocols and standard are provided in Section 1.1.3 and 1.1.4, respectively.

1.1.1 RFID Technologies

An RFID system consists of RF readers and RF tags. RF readers are connected to the back-end database server. The regular computer network security techniques can be applied to protect the communication between a reader and the back-end server. The signal from a reader to a tag is called the forward channel, and the signal from a
tag to a reader is called the \textit{backward channel}. The process that a reader singulates individual tags in its reading region is said to be an \textit{interrogation}.

RF tags are classified either \textit{active} or \textit{passive}. Active tags have their own power sources, and thus computationally more powerful. On the other hand, passive tags do not have a power source, and are very weak devices. Nevertheless, passive tags are very cheap. In fact, passive tags’ price ranges from 5 to 10 cents. Since most RFID applications employ passive tags, in this dissertation, we consider RFID systems with passive tags.

Due to the power constraints of passive tags, the communications between a reader and a tag are very different from the traditional computer communications. In an RFID system, a reader can communicate with tags by a query-and-response exchange. First, a reader needs to energize tags by its signal. On receiving a query, a tag charges power, modulates its ID, and replies by reflecting the reader’s signal. The characteristics of RF communications [1] with passive tags are as follows:

- Limited tuning capability - A tag cannot filter out unwanted frequency. This indicates that tags are capable of communication with any reader in every region in the world. However, interference may degrade the quality of data communications.

- Long forward channel and short backward channel - A reader must transmit a signal with higher power required by communications to energize a tag. In general, the forward channel range is normally five times longer than that of the backward channel.
• Reply by signal reflection - A tag does not generate a signal due to the power constraint. Thus, it simply reflects a reader’s signal with possibly additional modulation for its ID.

1.1.2 RFID Applications

RFID technologies enable a number of business and personal applications. The most common application is the inventory management, including supply chain management [2, 3], library management [4], natural habitat monitoring, and warehouse operations [5], and more. In these applications, tags are attached to items, products, or even animals for monitor objects. A very large number of objects are effectively managed to smooth the physical transaction in the logistic.

Another famous application is the electronic money [6, 7], such as transportation payment and smart cards. In an electronic toll collection (ETC) system [6], a passive RF tag is used as the unique identifier of a vehicle. With the current RFID technology, a reader and a tag can exchange approximately 100 queries and responses in one second. Hence, the driver can pass a tollbooth without stopping. In addition to ETC, RFID systems are applied to electronic money systems, such as subway payment cards and Suica [7]. With this card, a user can go through a ticket gate without the hassle of paying and carrying a ticket each time he/she take the subway.

Recently, RFID-based localization [8–10] is introduced. In the indoor localization [8, 9], tags are deployed on the ground, and these tags are used as reference points. The robot with an RF reader can compute its location by reading tags in its proximity. Kiva system [11] is one of the realization of indoor localization, which
has being used in Amazon.com inventory management. Moreover, RF tags can be utilized for vehicular localization [10].

1.1.3 RFID Communication Protocols

In RFID systems, a reader has to recognize individual tag IDs in its reading region. However, collisions may happen when several tags respond simultaneously to a reader’s query. Therefore, we need anti-collision singulation schemes for a reader to effectively identify tags in its proximity. Current singulation protocols can be roughly categorized into Aloha scheme based protocols and tree-walking scheme based protocols.

Aloha-Based Singulation

In Aloha-based protocols [12–14], named after an early wireless network protocol developed at the University of Hawaii, a reader sends a query frame and each tag randomly chooses a time slot to send its ID information. If more than two tags select the same slot, collisions occur. The colliding tags get to choose another slot to send a response in the next round of query. In addition, the reader can adjust the frame size according to the number of collisions in the previous frame. Although Aloha-based protocols avoid collisions to identify tags, a specific tag may not be identified for a long time. This issue is called the tag starvation problem.

An example of Aloha-based protocol is shown in Figure 1.1, in which there are five tags. A reader first sends a frame, say a frame with 8 time slots, and then each tag randomly selects a time slot to reply. Each time slot could be either in idle, singleton, or collision status. Consider the example shown in Figure 1.1 (a), the second, the fourth, and the seventh time slots are singleton, and thus the reader can identify the
tags that responded with these time slots. However, tag 1 and 4 selects the fifth time slot, leading to a collision in the fifth time slot. Since a collision occurs, a reader enlarge the frame as shown in Figure 1.1 (b). There are 16 time slots in the second frame which is most likely to be enough to singulate five tags. If there is no collision in any time slot in the frame, the reader can identify all tags in its vicinity.

Figure 1.1: An example of the Aloha-based protocol.

Tree-Based Singulation

In tree-walking based protocols [15–18], a reader traverses a binary tag tree, which organizes the entire tag ID space. Each tag ID is mapped to a leaf node in either depth-first or breadth-first order. For singulation, a reader broadcasts a query to all tags in the vicinity for the next bit of their ID. On receiving a query, a tag responds if its ID matches the prefix of the bit string in the query. If more than one tag responds, the reader will be able to detect the collision. Afterward the reader will broadcast a bit indicating whether tags who replied with a 0 or those who replied with a 1 should continue. By applying this mechanism, all tags in the interrogation area will
be identified. While tree-walking based protocols may incur a long singulation delay, they do not suffer from the tag starvation problem that occurs with Aloha-based protocols.

Figure 1.2 illustrates an example of the tree-walking protocol. In this example, there exist five tags, and each tag has a 4-bit ID. The circles represent nodes in the tree. Each node is assigned with prefix except the root. A reader will singulate tags by traveling the query tree. Assume a reader queries tags with the depth first order. The reader first sends a query with the prefix 0. Since Tag 1 and Tag 2 has the prefix 0, both of them replies to the query, and a collision occurs. Then, the reader sends a query with the prefix 00. Again both Tag 1 and Tag 2 has the prefix 00, and a collision happens. Then, the reader sends a query with the prefix 000. Only Tag 2 has the prefix 000, so there will be no collision and thus the reader can identify Tag 2. Likewise, the reader will identify Tag 1 by querying with the prefix 001. Similarly, the reader singulate the rest of the tags by traveling the other half of the query tree. The internal nodes with a solid circle indicates that a collision occurs under the corresponding prefix, the nodes with a gray circle is singleton (i.e., a reader can identify a tag), and the node with dashed circle implies that no node replies to a query.

1.1.4 RFID Standard

The most well-known RFID standard is EPC global Gen 2 [12], and thus based on the current standard the next generation of EPC global is considered in this dissertation. There are three phases in the identification process of an RFID system. In the first phase, a reader singles out tag populations with bit operations, e.g., union
and intersection of IDs. In the second phase, individual tags are identified by queries and responses. In the third phase, the reader accesses to the tags’ content. As an anti-collision mechanism, either Aloha [12–14] or tree-walking protocols [15–18] can be used. Every tag contains its unique ID (96-bit in EPC global Gen 2) and data. A user can obtain object’s information by a tag’s ID from the back-end database server or data in tags’ memory. Note that the memory size of a passive tag is normally less than 512 bits.

1.2 Research Issues in RFID Systems

In this section, we discuss major research issues that are unique to RFID systems.

1.2.1 Cardinality Estimation

In Aloha-based protocol, an appropriate frame size depends on the number of tags in a reading area. For example, a singulation process will slow down when the frame
size is too small with respect to the number of tags. Consider that a reader starts querying with the frame size of 8 for 1000 tags, and the frame size increases twice every rounds. Since the frame size must be more than 1000 to identify all the tags, the first 7 rounds are useless. Note that the frame sizes of the first 7 rounds are 8, 16, 32, 64, 128, 256, and 512, respectively. On the other hand, if the frame size is too large with respect to the number of tags, there will be many redundant time slots in the frame. Thus, an appropriate frame size is critical for the performance of Aloha-based protocols. To tackle this issue, many studies [19–22] are devoted to estimate the number of tags in the reading region.

1.2.2 Improving Singulation Protocols

In general, a reader interrogates tags many times in an inventory management. Although the number of tags in the inventory could be different from time to time, the drastic change in the number of tags is rare. Thus, in the tree-walking protocol, a reader does not have to query starting from 0 or 1, when it knows the distribution of the tag IDs in the inventory. Some works, e.g., [17, 18], improve the performance of the tree-walking protocol by using the information obtained in the previous interrogations.

1.2.3 Handling Multiple Readers

In some context, there exit multiple readers in an RFID system. To effectively singulate individual tags under the interference among readers, slotted scheduling [23] was proposed. Another literature [24] addresses the cardinality estimation in multiple-reader environments.
1.2.4 Missing Tag Problems

To defend objects from missing events, such as theft, a timely detection operation should be performed. This issue is called the missing tag problem. A naive way to find missing tags is to identify all tags in a region and compare with the tags listed in the database. However, this approach may take a long time in a large-scale RFID system. Hence, some researchers [25, 26] adaptively remove identified tags during an interrogation to quickly find out missing tags.

1.2.5 The Optimal Tag Deployments

In RFID-based localization, deployed tags are used as references. Since the communication range of a passive tag is very short, a number of tags must be deployed on a region. Assuming the communication range of a tag is circle, we can define the optimal tag deployment as follow: what deployment pattern results in the minimum number of tags to cover a region? When we simply cover a region with circles with a constant radius, the equilateral triangle with the nearest tags distance $\sqrt{3}r$, where $r$ is the communication range, is optimal [27, 28]. However, when it comes to RFID applications, engineering considerations must be taken into account. In fact, coverage requirement is application dependent. For example, the accuracy of indoor localization [9] will be improved when the entire region is $k$-covered, in which a reader can access at least $k$ tags at any point in the region. The optimal tag deployment pattern differs for each $k$ value. Similarly, human localization [8] and vehicle localization [10] have different coverage requirements.
1.2.6 Private Tag Authentication

Since the tag’s ID is used as a pointer to the corresponding data entry in the back-end server, the ID itself is critical information. As discussed, a tag transmits its ID over the backward channel during an interrogation. Hence, adversaries could eavesdrop the backward channel to obtain tags’ ID. Therefore, to protect tags’ content, private tag authentication must be addressed, where a reader can identify individual tags in keeping with defending tags’ ID. Depending on RFID contexts, there are two approaches, encryption-based and non-encryption-based.

The encryption-based authentication is used for inventory managements, such as RFID library, where the system administrator can assign secret keys to tags before deployment. While this approach guarantees the privacy protection, cryptographic operations induced by a private authentication protocol reduce the performance in term of singulation time. Hence, one of the challenges is how to break the tradeoff between the performance and the degree of security. Unfortunately, none of the existing works [29–34] has accomplished this task.

On the other hand, the non-encryption-based authentication is used when exchanging a common secret before communications is not possible. This is the case of the user identification, such as tollbooth [6]. For a reader to securely access tags, a physical layer technique, e.g., jamming, is conducted during a singulation process. While some studies [35–37] successfully protect tags’ content against eavesdroppers, they assume unpractical physical layer assumptions.
1.2.7 Data Verification

Data verification guarantees the authenticity of data. In general, a set of data is said to be verifiable, if the data set cannot be modified without being detected. However, the precise definition of verifiability is different from applications to applications. To the best of our knowledge, there is no clear definition of data verification in RFID-based data service at this moment.

1.3 Contribution of This Dissertation

In this dissertation, we propose solutions to private tag authentication and data verification problems. The contributions of each chapter in this dissertation are as follows:

- Contributions of Chapter 2
  1. We design a new encryption-based private authentication protocol based on skip lists. Unlike the existing solutions, the proposed protocol provides strong privacy protection in keeping with high performance.
  2. We define a new metric to measure the degree of anonymity against a physical attack to an encryption-based authentication protocol.
  3. We quantitatively and qualitatively analyze the new authentication protocol based on skip lists.

- Contributions of Chapter 3
  1. We introduce the distributed RFID architecture, where an RF reader is divided into two components, an RF activator and an RF listener, for
non-encryption-based private authentication problem, in which a common secret between an RF reader and an RF tag is not possible.

2. We develop a non-encryption-based private authentication protocol with a novel coding scheme and jamming environment. Our design relies on the existing physical layer techniques, and thus is practical.

3. It is known that the perfect secrecy is possible [38] by incorporating the physical layer technologies, e.g., jamming. We prove the proposed solution achieves the perfect secrecy when the jamming succeeds.

4. We formulate the degree of anonymity, which is an entropy-based security metric, for the proposed non-encryption-based authentication.

- Contributions of Chapter 4

1. We introduce an integrated RFID architecture, where multiple RFID systems are merged into an exa-scale system and a single tagging system exists.

2. We define a new research problem, data verification in an integrated RFID system. In a verifiable RFID system, data generated by the RFID-based data service is guaranteed its authenticity in the sense that a client can verify a set of its data is associated with a particular tag and any element of the data set cannot be modified without being detected.

3. We propose two data verification protocols for the verifiable RFID system.

4. We build adversary models, the illegal data access and modification, and formulate security measurements to evaluate the proposed verification protocols.
5. We implement a prototype of the verifiable RFID system.

1.4 Organization of This Dissertation

The rest of the dissertation is organized as follows. In Chapter 2, we propose a new encryption-based private authentication, called Randomized Skip Lists-Based Authentication (RSLA). In Chapter 3, we study the distributed RFID architecture and design a non-encryption-based private authentication, namely Random Flipping Random Jamming (RFRJ). In Chapter 4, we introduce verifiable RFID system, and then address data verification problems in an exa-scale integrated RFID system. We conclude this dissertation in Chapter 5.
Chapter 2: Encryption-Based Private Authentication

In this chapter, we present a new encryption-based private authentication protocol, called Randomized Skip Lists-Based Authentication (RSLA).

Radio Frequency Identification (RFID) is widely used to smooth the way of various applications, such as library management [29], transportation payment, natural habitat monitoring, indoor localization [9, 39], and so on. In these systems, the administrator manages and monitors a large number of objects by reading passive RF tags attached to the objects with an RF reader. To protect the tag’s content, low-cost cryptographic operations [40] are conducted during singulation process. Hence, on receiving the tag’s reply, the reader must try all keys in the system to find the corresponding key that the tag used in order to decrypt the content. When it comes to a large-scale RFID system, the authentication process can take a long time.

To accommodate this issue, a number of private tag authentication protocols with structured key management have been proposed. In these approaches, a unique key and a set of group keys are assigned to each tag. The group keys are shared among several tags and are used to confine the search space of the unique key corresponding to a tag’s reply. Based on how group keys are managed, they are categorized into two types: tree-based [29–32, 41, 42] and group-based protocols [33, 34]. In a tree-based protocol, tags are mapped to leaf nodes in the tree and keys are assigned to internal
nodes. Each tag has its unique key and a set of shared keys associated with the nodes from the leaf to the root. By traveling the tree, the reader can securely singulate tags. This results in high authentication efficiency, but discloses a large amount of information once tags in the system are compromised. On the contrary, in a group-based protocol, each tag has two kinds of keys: a unique key and a group key. With this approach, even if one of the group members is compromised, tags in other groups are intact. However, the authentication efficiency of this approach is low.

Therefore, for large-scale RFID systems, the performance and privacy/security of key authentication are commonly seen as tradeoffs. In this research, we propose a scheme that provides both good performance and a high level of privacy/security for a large-scale RFID system. Since both tree-based and group-based structures have pros and cons, we take a different approach based on skip lists [43], a data structure with which operations are performed in a logarithmic order like a balanced tree. In our proposed scheme, an interrogator authenticates a tag by traveling skip lists from top to bottom with a random rotation at each level. The analysis and simulation results prove that the proposed scheme is both efficient on authentication complexity and resistant against compromise attacks. In summary, the contributions of this chapter are as follows.

- We propose a new private tag authentication protocol, named Randomized Skip Lists-based Authentication (RSLA), which provides strong privacy protection and high performance of authentication like the tree-based approach.

- We design the key-updating and system maintenance mechanisms for RSLA to adapt to dynamic environments, where existing tags update their keys, and new tags join or leave the system.
• We conduct performance and security analyses to demonstrate that RSLA achieves its design goals: a high level of security/privacy and good performance.

• We evaluate the proposed RSLA by simulations, and validate that RSLA outperforms the existing solutions.

**Organization:** Section 2.1 reviews the existing works. In Section 2.2, we propose RSLA. Section 2.3 provides the analyses of the proposed scheme, and Section 2.4 demonstrates the simulation results.

2.1 Related Works

2.1.1 Private Authentication

Private RFID authentication protocols are classified into two categories: non-encryption-based and encryption-based. Non-encryption-based authentication [36,37,44] can be applied to some particular contexts, but not to large-scale RFID systems such as RFID-based libraries and super markets. In these applications, it is natural that the provider of the RFID system issues keys to tags before the tag deployment. Therefore, in this paper we focus on encryption-based private authentication. The encryption-based authentication protocols can be further divided into unstructured, tree-based and group-based, which will be elaborated on in the following subsections.

**Unstructured Authentication**

Due to the computational power constraints of passive tags, traditional cryptographic operations are not practical. Many studies put forth low-cost encryption [40], which relies on only simple functions such as hash, concatenation, and XOR. Weis et al. proposed Hash-lock [16] which uses a hash value to identify tags. That is, on
receiving a query from a reader, a tag replies with the hash value of its unique key. However, such an approach requires an RF reader to try all keys in the database to decrypt or compute hash values to validate a tag’s reply, which leads to a slow authentication speed proportional to the number of tags in the system. This motivates private authentication to have a structured key management.

**Tree-Based Authentication**

In tree-based authentication schemes [29,30], unique keys are mapped to the leaf nodes of a balanced tree, and group keys are mapped to non-leaf nodes. In addition, each tag in the system is associated with a leaf node. A tag obtains its keys on the path from the associated leaf to the root. Thus, each tag has one unique key and a set of group keys, denoted as $sk$ and $GK = \{g_{k1}, g_{k2}, \ldots\}$, respectively. A tag computes a set of hash values with $g_i$ and nonce at each level $i$ of the tree. Starting from the root, an RF reader tries all group keys associated with the children of each non-leaf node. When the reader reaches the bottom of the tree, it applies $sk$ corresponding to the leaf node. Thus, a tree-based protocol runs in $O(\log_k N)$, where $k$ is the balancing factor of the tree and $N$ is the number of tags.

While the tree-based approach is fast, it sacrifices privacy against compromise attacks, because two or more tags’ replies can be related using the keys obtained from compromised tags. To measure the degree of security, anonymity [45] is widely used, which is defined as a state of not being identifiable among an anonymous set. An anonymous set is the set of all possible tags from which replies they are indistinguishable.

Figure 2.1 illustrates the key structure with a binary tree, in which 8 tags are mapped to the leaf nodes. For example, Tag 3 has a unique key $sk_3$ and the group
keys $GK_3 = \{g_{k1,1}, g_{k2,2}\}$. Should Tag 3 be compromised, an adversary will have all keys that Tag 3 has. Hence, replies from other tags will be partially disclosed. In addition, tags are divided into 3 disjoint groups, i.e., \{1, 2\}, \{4\}, and \{5, 6, 7, 8\} as shown in Figure 2.1. As a result, anonymity of each tag decreases.

Tree-based protocols have good performance but result in low anonymity should some tags be compromised. This motivates a number of studies to improve the privacy protection mechanism of tree-based authentications. To alleviate compromise attacks, Lu et al. [41] proposed SAP that augments the tree-based authentication with a dynamic key-updating in which shared keys in the tree are periodically updated. In [32], Li et al. successfully reduce the communication overhead based on cryptographic encodings by having tags reply partial bits of the path indicator.

Other studies utilize a tree in different ways. Lu et al. [31] used a sparse tree to reduce the dependency of shared keys. A path indicator is assigned to each tag for fast authentication. However, as pointed out by [32], the possible space of path indicators is small and therefore their protocol is vulnerable to the brute force attack.
against hash values of the path indicator in a tag’s reply. In Yao et al. [42], tags do not share any key with other tags, and non-leaf nodes are used as anchors to the corresponding leaf node. By finding the anchor using the random tree walk, a reader will find a valid tag’s key. Although Yao et al. claimed that their approach reduces the authentication complexity to $O(1)$, tags are required to perform randomized hash functions, which is not suitable for passive tags with low computational power.

**Group-Based Authentication**

In group-based authentication schemes [33, 34], tags are divided into disjoint groups. A reader assigns each tag two keys, a unique key $sk$ and a group key $gk$. A tag’s reply consists of two components encrypted by $gk$ and $sk$, respectively. A reader first tries all group keys to decrypt the first component, which contains the group ID that the tag belongs to. Then, the reader applies the unique keys associated with the group to decrypt the second component in order to verify the tag’s authenticity.

In group-based authentication, if only a few tags are compromised, tags in other groups are intact. For example, in Figure 2.2, 8 tags are divided into 4 groups, each with 2 members. Assume Tag 3, which has $sk_3$ and $gk_2$ is compromised. Although Tag 4’s identity is disclosed, the other tags are still indistinguishable, i.e., Tags 1, 2, 5, 6, 7, and 8, have the anonymous set size of 6.

While group-based protocols improve the anonymity if only a small portion of tags are compromised, they result in low authentication efficiency. The possible authentication complexity is $O(\sqrt{N})$.

To improve the privacy protection with a group-based scheme, Hoque et al. proposed AnonPri [34] where each tag obtains a set of pseudo IDs from the key issuer. Then, a tag replies with one of the pseudo IDs that it has. A reader first scans all
group keys to obtain a pseudo ID in tag’s reply, and then tries all unique keys associated with the pseudo ID. To guarantee that AnonPri works, each tag must share every pseudo ID it has with at least two tags in the same group. By doing so, AnonPri slightly improves the privacy preserving mechanism. However, the low authentication efficiency of the group-based approach has not been addressed.

2.1.2 Skip Lists

Skip lists [43] are a probabilistic data structure that consists of a set of ordered linked lists as shown in Figure 2.3. At the lowest level (labeled by Level 2), the list contains all nodes in increasing order of their keys. A node in the list at a level $i > 0$ appears at level $i - 1$ with probability $p$. Each node in a list has a pointer to each of the previous and next node. The expected number of lists in skip lists is $\log_{1/p} N$, where $N$ is the input size. Search, insert, and delete operations are performed in $O(\log_{1/p} N)$, since the number of steps at each linked list is $\frac{1}{p}$ on average. The space complexity is $O(N)$. Theoretically, skip lists can be considered as an alternative to a balanced tree.
For example, in Figure 2.3, to find Key 13, we start from the top level list. The list at Level 0 has only one node with Key 15. Since $13 < 15$, we travel toward the left at Level 1. At Level 1, we reach the node with Key 8 which is smaller than 13, and then we travel toward the right at Level 2. Finally, we find node with Key 13.

![Figure 2.3: An example of skip lists.](image)

## 2.2 Skip Lists-Based Authentication

### 2.2.1 Protocol Overview

In this chapter, we propose Randomized Skip-Lists based Authentication (RSLA) which consists of four components: key issuing (initialization), private authentication, key-updating, and system maintenance.

In the key issuing process, the system generates skip lists. RF tags are randomly assigned to nodes in the lowest level list. A unique key and a set of group keys are assigned to each tag by traveling from a node at the bottom to the top level list. In the authentication, an RF reader scans group keys to narrow the search space of the corresponding unique key for a tag by traveling from the top list to
the bottom list. The key-updating mechanism makes RSLA more invulnerable, and system maintenance deals with tags enrollment and removal.

The key idea of RSLA is random shifting in a list at each level and dependency among lists. This makes our skip lists-based scheme more secure than existing solutions, in keeping with the high performance of tree-based protocols.

2.2.2 Definitions and Assumptions

In our assumptions, an RFID system consists of $N$ tags and a reader, which is connected to the back-end server. For simplicity, it is assumed that the reader and the back-end server can securely communicate, and thus the reader is the final destination of a tag’s data.

We will denote by $n_r$ and $n_t$ the nonce randomly selected by the reader and a tag, respectively. For a given key $K$ and an input $x$, the hash function $H(x)$ is assumed to be collision resistant, and an encryption function $E(K, x)$ is implemented by low-cost cryptographic operations [40]. A reader is assumed to have enough computational power to run a decryption function $D(K, x)$ with a key $K$ and an input $x$.

2.2.3 Construction of Skip Lists

To construct skip lists for key management, we modify the construction process as follows. Instead of randomly selecting nodes that appear at the list in the upper levels, we deterministically select nodes to keep the number of nodes at each level consistent.

Let $L_i$ be the list at the $i$-th top level. Each list consists of a set of nodes. A node $i$, denoted as $v_i$, has pointers to left and right nodes in the same list, which are denoted by $v_i.left$ and $v_i.right$. The left pointer of the first node and the right
Table 2.1: Definition of notations.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(k)</td>
<td>The balancing factor of skip lists</td>
</tr>
<tr>
<td>(N)</td>
<td>The number of tags in the system</td>
</tr>
<tr>
<td>(\eta)</td>
<td>The height of skip lists, (\lceil \log_k N \rceil)</td>
</tr>
<tr>
<td>(L_i)</td>
<td>The list at Level (i) in skip lists ((0 \leq i \leq \eta))</td>
</tr>
<tr>
<td>(v_i)</td>
<td>Node (i) in a list</td>
</tr>
<tr>
<td>(sk_i)</td>
<td>Tag (i)'s unique secret key</td>
</tr>
<tr>
<td>(GK_i)</td>
<td>A set of group keys of Tag (i), ({gk_1, gk_2, ..., gk_{\eta-1}})</td>
</tr>
<tr>
<td>(R_i)</td>
<td>A set of random numbers of Tag (i), ({r_1, r_2, ..., r_{\eta-1}})</td>
</tr>
<tr>
<td>(n_t, n_r)</td>
<td>Nonces from a tag and a reader</td>
</tr>
<tr>
<td>(\beta)</td>
<td>Tag’s reply, ({\beta_1, \beta_2, ..., \beta_{\eta}})</td>
</tr>
<tr>
<td>(N_c)</td>
<td>The number of compromised tags in the system</td>
</tr>
<tr>
<td>(N_g)</td>
<td>The number of compromised tags in a group</td>
</tr>
<tr>
<td>(E(.), D(.))</td>
<td>The encryption and decryption functions</td>
</tr>
<tr>
<td>(H(.))</td>
<td>The hash function</td>
</tr>
<tr>
<td>(A)</td>
<td>System anonymity</td>
</tr>
<tr>
<td>(S_i)</td>
<td>Anonymous set that Tag (i) belongs to</td>
</tr>
</tbody>
</table>

pointer of the last node are null. In addition, the pointers to the first and last nodes of list \(L_i\) are kept in \(L_i.head\) and \(L_i.tail\).

We generate skip lists that contain \(\eta + 1\) lists. Each list \(L_i\) contains \(k^i\) nodes, where \(\eta\) is defined as \(\lceil \log_k (N) \rceil\) so that we can map all tags to the nodes in the lowest level list. Note that if there are more than \(N\) nodes, some nodes are not assigned a tag. Given the number of tags \(N\) and a balancing factor \(k\), a list \(L_\eta\) with \(k^\eta\) nodes is first created. Then, node \(v_i\) is added into \(L_{\eta-1}\) if \(i \mod k = 0\). For each level \(j\), node \(v_i\) \((0 \leq j \leq \eta - 1)\) is added into \(L_j\) if \(i \mod k^{\eta-j} = 0\). This process is repeated from
\( \eta \) to 0. The top level list always has one node, i.e., \( L_0 = \{ v_0 \} \), since the number of nodes at the lowest level list is \( k^\eta \).

Each node in skip lists has a set of keys. We define \( v_i.key[j] \) as the variable to store node \( v_i \)'s key for Level \( j \). If \( v_i \) does not appear in \( L_j \), \( v_i.key[j] \) is empty. Assuming Tag \( t \) is mapped to \( v_i \), the unique key \( sk_t \) of Tag \( t \) is located at \( v_i.key[\eta] \). Let us denote \( gk_{i,j} \) the group key, which is stored at \( v_i.key[j] \). Thus, all nodes in skip lists have a unique key in \( v.key[\eta] \), and group keys for Level \( j \) (\( 1 \leq j \leq \eta - 1 \)) in \( v.key[j] \) if \( v \) appears in \( L_j \). We do not assign any key to the node in the top level list \( L_0 \), since \( L_0 \) has only one node. Thus, \( v_0.key[0] \) is empty.

Since the construction of skip lists is deterministic, our skip lists with factor \( k \) work in similar fashion as a \( k \)-balanced tree. The reason why we employ skip lists instead of a balanced tree is that the link among the nodes in the same level is utilized for random rotation. Thus, we do not have to modify the data structure of skip lists to achieve the design goals.

### 2.2.4 Key Issuing

In RSLA, Tag \( t \) has three variables, the unique secret key \( sk_t \), a set of group keys \( GK_t \), and a set of random numbers \( R_t \). The high level idea of the key issuing process is as follows: Each tag \( t \) is randomly assigned to a node, say \( v_i \), in the lowest level list \( L_\eta \). Starting from \( v_i \), the key issuer traverses to the top list \( L_0 \) by shifting to the left for \( r_j \) nodes at each \( L_j \) (\( 1 \leq j \leq \eta - 1 \)), where \( r_j \) is randomly chosen between 0 and \( |L_j| - 1 \) (i.e., \( k^j - 1 \)). By doing this, the key of the selected node for each level is assigned to a tag. The key issuing algorithm described in Algorithm 1 is elaborated in the following subsections.
Lines 13 and 14 in Algorithm 1 show the steps in the bottom list. At \( v_i \) in \( L_\eta \), Tag \( t \) obtains the unique key from \( v_i\.key[\eta] \), and then the pointer moves to \( L_{\eta-1} \).

When \( v_i \) in \( L_\eta \) does not appear in \( L_{\eta-1} \), the pointer first moves to node \( v_j \) where \( j = i - (i \mod k) \), and then moves to \( L_{\eta-1} \). In general, for the current node \( v_i \) in \( L_m \), the pointer moves to \( v_j \) where \( j = i - i \mod k^{\eta-m+1} \), and then goes to \( L_{m-1} \).

Thus, this can be seen as a parent and children relation of a \( k \)-balanced tree, i.e., \( v_j \) in \( L_{m-1} \) has \( k \) children \( v_i \) in \( L_m \) (\( j \leq i \leq j + k - 1 \)).

From lines 15 to 24 in Algorithm 1, the key issuer assigns group keys and shift numbers. Every time, the pointer arrives at an upper level list, for instance \( L_j \), the key issuer takes the left shift by \( r_j \) at \( L_j \) (\( 1 \leq j \leq \eta - 1 \)). Here, \( r_j \) is randomly selected between 0 and \( |L_j| - 1 \), and added to set \( R_t \). Note that the left shift is not taken at \( L_\eta \) and \( L_0 \). The shifting can be done by moving the pointer via \( v_i\.left \). If \( v_i\.left = null \), i.e., \( v_i \) is the first node in \( L_j \), the pointer moves to \( L_j\.tail \), i.e., the last node in \( L_j \). Let \( v_i \) be the node in \( L_j \) after shifting. Tag \( t \) obtains the group key from \( v_i\.key[j] \). Then, the pointer moves to the upper level. This process continues until the key issuer reaches \( L_0 \).

At the end of this process, Tag \( t \) has one unique key, \( \eta - 1 \) group keys, and \( \eta - 1 \) random numbers.

**Example** Consider an RFID system with 8 tags that uses skip lists with \( k = 2 \) and \( \eta = 3 \) for key assignment as shown in Figure 2.4. Tags are mapped to the nodes in \( L_3 \). We illustrate how the key issuer assigns group keys and random numbers to a tag, for instance Tag 3. Starting from \( v_3 \), the key issuer traverses to the top level list. First, Tag 3 obtains \( sk_3 \) stored at \( v_3\.key[3] \), and the pointer moves to Level 2 via \( v_2 \). Because \( v_3 \) does not appear in \( L_2 \), the pointer goes to \( v_2 \ (3 - 3 \mod k = 2) \), and then
Algorithm 1 Key Issue

1: /* Key Issuer does following */
2: Issuer locates all tags $t$ to node $v_i$
3: /* For each tag $i$ Key Issuer does following */
4: for for each tag $t$ in the system do
5: KeyIssue($i$, $v_i$)
6: end for
7: /* The function to assign keys to Tag $t$ */
8: KeyIssue($t$, $v_i$)
9: /* $v_i$ is the current node */
10: $R_t = \phi$ /* Initialize the random numbers list */
11: $GK_t = \phi$ /* Initialize the group keys list */
12: /* At the lowest level list $L_0$ */
13: $sk_t v_i : key[j]$  
14: $v_i \leftarrow v_m$ where $m = i - (i \mod k)$
15: for $(j$ from $\eta - 1$ to $1$) do
16: /* Random shifting by $r$ and add a group key */
17: $r \overset{uniform}{\leftarrow} [0, |L_j| - 1]$
18: Add $r$ to $R_t$
19: $v_i \leftarrow$ shift to the left by $r$
20: Add $v.key[j]$ to $GK_t$
21: /* Move to upper level */
22: $v_i \leftarrow v_m$ where $m = i - (i \mod k^{\eta - j + 1})$
23: $j = j - 1$
24: end for

Figure 2.4: An example of key issuing.

moves to Level 2. Assume the key issuer randomly selects $r_2 = 3$ and the pointer shifts to the left by 3. At the same time, 3 is added to $R_3$. The current pointer is
now at $v_4$ in $L_2$. The issuer assigns $gk_{4,2}$ stored in $v_4.key[2]$ to Tag 3. This process continues until the issuer reaches $L_0$. Assume Tag 3 selects $r_1 = 1$ at Level 1. It obtains $sk_3$, $GK_3 = \{gk_{0,1}, gk_{4,2}\}$, and $R_3 = \{1, 3\}$.

### 2.2.5 Authentication

After issuing keys, the reader can securely communicate with tags. In RSLA authentication protocol, the reader first sends a query with nonce $n_r$, then a tag generates a reply message with nonce $n_t$, and then the reader decrypts the tag’s reply.

The pseudo code of the replying process at tags’ side is illustrated in Algorithm 2. Assume Tag $t$ has the unique key $sk_t$, a set of group keys $GK_t = \{gk_{1}, gk_{2}, \ldots, gk_{\eta-1}\}$, and a set of random numbers $R_t = \{r_1, r_2, \ldots, r_{\eta-1}\}$. On receiving a query with nonce $n_r$ from the reader, Tag $t$ generates a reply message with nonce $n_t$. Let $\beta = \{\beta_1, \beta_2, \ldots, \beta_\eta\}$ be the reply message. Here, $\beta_i$ ($1 \leq i \leq \eta$) consists of a hash value $\beta_i.hash$ and encrypted number $\beta_i.num$ at each level $i$. Lines from 5 to 9 in Algorithm 2 elaborate on how $\beta_i$ ($1 \leq i \leq \eta - 1$) is computed by group keys. The hash value $\beta_i.hash$ is obtained by $H(gk_i||r_{i-1}||n_t||n_r)$ with the base $r_0 = empty$. In other words, $\beta_1.hash = H(gk_1||n_t||n_r)$ because there is no rotation at $L_0$. The reason that we include the number at the previous level, i.e., $r_{i-1}$ for $\beta_i.hash$, is to enforce dependency between the levels to keep high anonymity. The random number $r_i$ is encrypted by $E(gk_i, r_i)$ and set to $\beta_i.num$. For the last element $\beta_\eta$, the hash value $\beta_\eta.hash$ is defined by $H(sk_t||r_{\eta-1}||n_t||n_r)$ where the unique key is used as shown in line 10 in Algorithm 2, and $\beta_\eta.num$ is empty. Finally, the tag sends $n_t$ and $\beta$ to the
reader. Note that $\beta$ contains $\eta$ elements. One of them is computed using $sk$; the other $\eta - 1$ elements are computed using $gk_i$ ($1 \leq i \leq \eta - 1$).

**Algorithm 2** ReplyToReader($n_r$)  

```
1: /* Assume Tag $t$ has $sk_t$, $GK_t$, and $R_t$/
2: /* where $GK_t = \{gk_1, gk_2, ..., gk_{\eta-1}\}$/
3: /* and $R_t = \{r_1, r_2, ..., r_{\eta-1}\}$/
4: Generate nonce $n_t$
5: for $i$ from 1 to $\eta - 1$ do
6:   $\beta_i.hash \leftarrow H(gk_i||r_{i-1}||n_t||n_r) /* r_0 = empty */$
7:   $\beta_i.num \leftarrow E(gk_i, r_i)$
8:   Add $\beta_i$ to $\beta$
9: end for
10: $\beta_{\eta}.hash = H(sk_t||r_{\eta-1}||n_t||n_r)$
11: reply $n_t$ and $\beta$
```

On receiving Tag $i$’s reply, the reader scans group keys associated to nodes from the top level list as shown in Algorithm 3. At the beginning, the pointer is at node $v_0$ in $L_0$. In $L_1$, there are $k$ nodes, and one of them has the group key $v_i.key[1]$ ($v_i \in L_1$) that matches the group key used for $\beta_i.hash$. After finding the corresponding key used for $\beta_i.hash$, the reader decrypts $\beta_i.num$ with the key. Then, we first move the pointer to $L_1$ from $L_0$, and shift the pointer to the right by $\beta_i.num$. If the pointer reaches the tail during shifting, it moves to the head of the same list. Note that the left shift was taken for key assignment by traveling from the lowest level, and on the contrary, the authentication process takes the right shift since the reader travels skip lists from the top. Assume $v_i$ is the current node after shifting right by $r_1$. The list $L_2$ has $k^2$ nodes, but only $k$ nodes $v_j$ ($i \leq j \leq i + k$) need to be scanned. This is because one of the $k$ nodes has the group key for $\beta_2$. This process continues until the reader reaches the bottom. Since the key at $L_\eta$ is unique for a tag, the reader singulates the tag from $\beta$. The reader scans no more than $k$ keys at each level $1 \leq i \leq \eta$, hence
our skip lists imitate the search operation of a $k$-balanced tree. During this process, should the reader be unable to find a group key at any level, the tag’s reply is invalid and the reader returns a \textit{FAIL} message.

\begin{algorithm}
\caption{Authentication($n_r, n_t, \beta$)}
\begin{algorithmic}[1]
  \State // $\beta = \{\beta_1, \beta_2, ..., \beta_\eta\}$\*/
  \State $v_0 \leftarrow \text{head}$ \/license* the pointer to the current node \*/
  \For {$j$ from 1 to $\eta$}
    \State /* Scan \texttt{v.key}[$j$] for $k$ nodes from $v_i$ */
    \For {$m$ from 1 to $k$}
      \State /* Note that the base $r_0 = \text{empty} */
      \If {$H(v_i.key[$j$]|r_{j-1}|n_r|n_t) = \beta_j.hash$}
        \If {$j == \eta$}
          \State Identify Tag $t$ by the unique key $v_i.key[$j$]$
        \Else
          \State $r \leftarrow D(v_i.key[$j$], \beta_j.num)$
          \State $v_i \leftarrow \text{shift to the right by } r$
          \State $j \leftarrow j + 1$
        \EndIf
      \EndIf
      \State $m \leftarrow m + 1$
    \EndFor
    \State if The key is not found for $L_j$ then
      \State return \textit{FAIL}
  \EndIf
\EndFor
\State return $t$
\end{algorithmic}
\end{algorithm}

\textbf{Example} We provide an example to demonstrate how the reader authenticates Tag 3 as shown in Figure 2.5. Tag 3’s parameters are $sk_3$, $GK_3 = \{gk_{0,1}, gk_{4,2}\}$, and $R = \{1, 3\}$. Thus, the reply of Tag 3 $\beta$ as follows:

\begin{align*}
  \beta_1 &= H(gk_{0,1}|n_t|n_r), E(gk_{0,1}, 1) \\
  \beta_2 &= H(gk_{4,2}|1|n_t|n_r), E(gk_{4,2}, 3) \\
  \beta_3 &= H(sk_3|3|n_t|n_r), \text{empty}
\end{align*}
On receiving the tag’s reply $n_t$ and $\beta$, the reader travels skip lists as shown in Figure 2.5. First, the reader scans $v_1.key[1]$ and $v_4.key[1]$ in $L_1$, i.e., $gk_{0,1}$ and $gk_{4,1}$, to compare the obtained hash value with $\beta_1.hash$. As the key $gk_{0,1}$ works, the reader applies $D(gk_{0,1}, \beta_1.num)$ and obtains $r_1 = 1$. The reader takes the right shift by 1, and moves to $v_4$. For Level 2, the reader scans two nodes as $k = 2$ in this example, i.e., $v_4.key[2]$ and $v_6.key[2]$. The reader will validate that $gk_{4,2}$ works for $\beta_2.hash$ and obtains $r_2 = 3$ from $\beta_2.num$. This process continues until the reader reaches the lowest level list $L_3$. At Level 3, the reader scans the unique keys stored at $v_2.key[3]$ and $v_3.key[3]$. The hash value obtained with $sk_3$ returns the same value with $\beta_3.hash$. Since $v_3$ corresponds to Tag 3, the reader finally concludes the reply comes from Tag 3.

2.2.6 Key Update

Secure RFID systems should periodically update shared keys to avoid tag compromise attacks. In SPA [41], the reader first updates keys at a tag when it accesses the tag, and then updates the corresponding keys in the key tree.
However, our RSLA updates keys in the opposite order to simplify the key updating mechanism. First, the reader updates all the keys in all of the skip lists, and updates tag side information upon accessing tags.

The pseudo code is given in Algorithm 4. First, the key issuer generates a random number $r$, and then all keys in skip lists are updated by visiting each node of each list. Note that the random number $r$ generated by the key issuer is kept in secret. By doing this, an attacker cannot compute a new key, even though she obtains the current key and the hash function from a compromised tag.

The new key is obtained by $H(r, v.key[i])$ ($0 \leq i \leq \eta$) where $r$ is a random number. The old key at a node, for instance $v.key[i]$ ($1 \leq i \leq \eta$), is kept as $v.old.key[i]$, so that tags with old keys can be singulated. For tags’ side, the reader updates a tag’s unique key, group keys, and random numbers by Algorithm 1 when it accesses a tag. A tag maintains only the latest set of keys and numbers, and the old ones are discarded upon updating. Therefore, our key-updating mechanism can successfully renew the keys in the system while the reader can still access tags with old keys.

\textbf{Algorithm 4 KeyUpdate}

1: Generate a random number $r$
2: for for each $i$ from 0 to $\eta$ do
3: \hspace{1em} $v \leftarrow L_i\_head$
4: \hspace{1em} while $v.right \neq \text{null}$ do
5: \hspace{2em} $v.old\_key[i] \leftarrow v.key[i]$
6: \hspace{2em} $v.key[i] \leftarrow H(r, v.key[i])$
7: \hspace{1em} $v \leftarrow v.right$
8: \hspace{1em} end while
9: end for
2.2.7 System Maintenance

In RFID applications, it is natural that tags join and leave the system. Thus, RSLA also provides tag enrollment and removal mechanisms.

When a new tag joins the system, the system first tries to find a node in the lowest level list \( L_n \) such that no tag is assigned to. If found, the key issuer assigns a unique key, group keys, and random numbers, by Algorithm 1. If there is no such node, a new set of skip lists with the same size as the original one will be created and assign keys to the new tag. Thus, the reader needs to scan \( 2^k \) nodes at the beginning. There are \( k \) keys in the original skip lists; the other \( k \) keys are in the new skip lists. After this, the reader narrows the search space to either skip lists. This does not affect authentication efficiency, since only \( k \) more keys need to be scanned by the reader only for the first element of the reply, i.e., \( \beta_1 \).

Note that adding a new list to the existing skip lists causes a large amount of overhead, because the key issuer must assign one more group key to all tags and thus the entire key structure must be reconstructed. Therefore, the proposed maintenance procedure creates a new set of skip lists if there is no node to map a new tag to.

The tag removing process is simple. The system removes the tag from the corresponding node in \( L_n \).

2.2.8 Implementation Issues

One of the implementation issues is the small domain of the random numbers \( R = \{r_1, r_2, \ldots\} \), as each element \( r_i \) is randomly selected between 0 and \( k^i - 1 \). For example, assume that an attacker has a group key \( gk_i \) for level \( i \) of an uncompromised tag, but does not have \( gk_{i-1} \) and \( r_{i-1} \). To compute the valid hash value at level \( i \), i.e.,
\( H(gk_i||r_i-1||n_t||n_r) \), the attacker must have both \( gk_i \) and \( r_{i-1} \). Even though attacker does not have \( r_{i-1} \), she can compute the valid hash value with \( gk_i \) by trying all integers from 0 to \( k^i - 1 \) as \( r_{i-1} \).

To avoid the brute force attack, the domain should be large enough. Thus, we can select \( r_i \) between 0 and \( 2^{32} \) (\( r_i \) will be 32 bits for all \( i \)), and then take the left shift by \( r_i \mod k^i \) at Level \( i \). This virtually protects a tag’s replies from the brute force attack.

### 2.3 Analyses

#### 2.3.1 Performance Analysis

Theorems 1 and 2 show the upper bound and the average time complexity of authentication in RSLA, respectively. The authentication running time is determined by the number of the computations of hash and decryption functions by a reader. Note that in the proposed protocol, skip lists are created deterministically, and thus the worst case complexity is the same as that of a balanced tree structure.

**Theorem 1** Given the number of tags \( N \) and the balancing factor \( k \), RSLA runs in \( O(\log_k N) \).

**Proof:** The number of lists in skip lists is \( \eta = \lceil \log_k N \rceil \leq \log_k N + 1 \). The reader scans \( k \) nodes for \( \beta_i.hash \) at Level \( i \), and decrypts \( \beta_i.num \). Thus, the number of computations is at most \( (k + 1)(\log_k N + 1) \). Therefore, RSLA singulates a tag in \( O(\log_k N) \). \( \blacksquare \)

**Theorem 2** Given \( N \) and \( k \), the average running time of RSLA is \( \frac{k+3}{2}\log_k N \).
Proof: Let $X_i$ be the random variable which represents the number of times a reader computes the hash values at Level $i$. Since the reader scans at most $k$ nodes in $L_i$ where $1 \leq i \leq \eta$, we have $1 \leq X \leq k$. Thus, the expected value $E[X]$ is obtained by

$$\sum_{i=1}^{k} \frac{i}{k} = \frac{k+1}{2}.$$ 

The reader applies the decryption function to obtain the random number, and thus the average number of computations at each level equals $\frac{k+1}{2} + 1$. There are $\eta$ lists excluding the top level list where there is no computation. $\eta$ can be approximated by $\log_k N$. Therefore, the average running time is $\frac{k+3}{2} \log_k N$. This completes the proof.

We deduce Theorems 3 and 4 for key storage cost of the system and tags.

Theorem 3 Given the number of tags $N$ and the balancing factor $k$, the number of keys in the system is bounded by $O(N)$.

Proof: Skip lists have $\eta$ levels, and $|L_i| = k^i$ number of nodes for each level $i$. Since the node $n_0$ in $L_0$ does not have a key for Level 0, i.e., $n_0.key[0] = \text{empty}$, the number of keys in the system is $\sum_{i=1}^{\eta} k^i = \frac{k^\eta - 1}{k-1} - 1$. Note that $\frac{k}{k-1}$ and $\frac{1}{k-1}$ are constants because $k$ is a constant. Therefore, the key storage cost is $O(N)$.

Theorem 4 Given the number of tags $N$ and the balancing factor $k$, the storage cost for a tag is bounded by $O(\log_k N)$.

Proof: A tag has one unique key, $\eta - 1$ group keys, and $\eta - 1$ random numbers. As $\eta \leq \log_k N + 1$, the storage cost for a tag is $O(\log_k N)$. This concludes the proof.
2.3.2 Unlinkability

In this subsection, we demonstrate unlinkability of RSLA. Unlinkability is defined as a state that two or more tags are no more or no less related after observing tags’ replies. That is, the probability for adversaries to identify (or relate) tags’ authenticity from two or more tags’ replies does not increase by observing the tags’ replies.

**Lemma 5** Without having the key to $\beta_i$, an interrogator (a reader or an adversary) cannot find the corresponding key to $\beta_{i+1}$.

**Proof:** The proof is by contradiction. For a given set of keys $GK = \{gk_1, gk_2, ..., gk_{\eta-1}\}$ and $\beta$, assume an interrogator can find the key $gk_{i+1}$ for $\beta_{i+1}$ without the key $gk_i$ for $\beta_i$. Recall that $\beta_{i+1}.hash$ is computed by $H(gk_{i+1}||r_i||n_i||n_r)$. This indicates that even if the adversary has the key $gk_{i+1}$, she cannot conclude the key is valid for $\beta_{i+1}.hash$ without the random number $r_i$, which can only be obtained by decrypting $\beta_i.num$ with the key $gk_i$. Hence, the interrogator must have $gk_i$ to find $gk_{i+1}$ for $\beta_{i+1}$. This is a contradiction. Therefore, the claim must be true. □

**Theorem 6** Given a compromised tag $t$ and uncompromised tag $t'$, the unlinkability of two tags holds as long as $GK_t \neq GK_{t'}$, where $GK_t = \{gk_1, gk_2, ..., gk_{\eta-1}\}$ and $GK_{t'} = \{gk'_1, gk'_2, ..., gk'_{\eta-1}\}$

**Proof:** The proof is by contradiction. Assume an adversary can tell the location of the nodes in $L_n$ that Tags $t$ and $t'$ are mapped to when $GK_t \neq GK_{t'}$. Note that we have $\eta - 1 = |GK_t| = |GK_{t'}|$. If there exists the case such that $gk_i \neq gk'_i$ for some $gk_i \in GK_t$ and $gk'_i \in GK_{t'}$, the adversary cannot distinguish $gk_{\eta-1}$ and $gk'_{\eta-1}$ by
Lemma 5. This is a contradiction. Therefore, the unlinkability of two tags holds as long as $GK \neq GK'$. This concludes the proof.

By Theorem 6, two tags, say $t_1$ and $t_2$, are indistinguishable as long as $GK_{t_1} \neq GK_{t_2}$. This means that even if Tag $t_1$ is compromised, the anonymous set size of $t_2$ remains $N - 1$. On the other hand, should $GK_{t_1} = GK_{t_2}$, the adversary can conclude the node, to which $t_2$ is mapped in $L_n$, is one of $k$ nodes. Note that this group with $k$ nodes in skip lists is similar to a branch with $k$ children in a tree. Thus, the anonymity set size of Tag $t_2$ is $k - 1$ if it is in the same group as Tag $t_1$. Otherwise, the anonymity set size is $k$. However, the probability is very small. This results in a high anonymity under fast authentication.

![Figure 2.6: Example of unlinkability (Tag 3).](image)

**Example** Assume that Tag 3 has $sk_3$, $GK_{t_3} = \{gk_{0,1}, gk_{4,2}\}$, and $R = \{1, 3\}$, and Tag 5 has $sk_5$, $GK_{t_5} = \{gk_{0,1}, gk_{6,2}\}$, and $R = \{1, 3\}$ as shown in Figure 2.6 and 2.7. The example will show that Tags 3 and 5’s replies are unliked since $GK_{t_3} \neq GK_{t_5}$.
Assume that Tag 3 is compromised and an attacker obtains all keys associated with Tag 3, i.e., $sk_3$, $gk_{0,1}$, and $gk_{4,2}$. Tag 5’s reply is computed as follows:

\[
\beta_1 = H(gk_{0,1}||n_t||n_r), E(gk_{0,1}, 1) \quad (2.4)
\]
\[
\beta_2 = H(gk_{6,2}||1||n_t||n_r), E(gk_{6,2}, 3) \quad (2.5)
\]
\[
\beta_3 = H(sk_5||3||n_t||n_r), \text{empty} \quad (2.6)
\]

Since the attacker has $gk_{0,1}$, she can compute $\beta_1$ in Equation 2.4. However, the attacker does not have $gk_{0,2}$ and thus cannot obtain the shift number $\beta_2.num$. Hence, there is no hint where the pointer to move in the list in Level 2 in Figure 2.7. By the dependency among lists, the attacker cannot compute $\beta_i$ without valid $\beta_{i-1}$. This implies that Tag 5 could be any of seven tags, $T_0$, $T_1$, $T_2$, $T_4$, $T_5$, $T_6$, or $T_7$, from the attacker’s perspective. Therefore, the anonymous set that Tag 5 belongs to is of size $N-1$, and the replies of Tag 3 and Tag 5 are unlinked.

Note that the skip lists structure in Figures 2.6 and 2.7 can be seen in another way of a binary tree in Figure 2.8, but the key assignment and authentication process
are different. In a tree-based protocol, Tag 3 and Tag 5 always obtain the group keys \( \{g_{k_{0,1}}, g_{k_{2,2}}\} \) and \( \{g_{k_{4,1}}, g_{k_{4,2}}\} \), respectively. Should Tag 3 be compromised, the attacker can divide tags into disjoint sets, and she can deduce Tag 5 belongs to the right sub tree with size 4 in Figure 2.8.

![Figure 2.8: The binary tree corresponding to Figure 2.6 and 2.7](image)

### 2.3.3 Analysis for Compromise Attacks

A privacy protection mechanism against the compromise attack can be measured by anonymity. Anonymity is a state of not being identifiable within an anonymous set. Let \( S_i \) be the anonymous set that Tag \( i \) belongs to. According to [34], the system anonymity, denoted as \( A \), can be formulated as Equation 2.7.

\[
A = \frac{1}{N} \sum_{i} |S_i|^2 
\]  

(2.7)

When no tag is compromised, \( |S_i| \) for any tag \( i \) equals \( N \) and therefore \( A \) equals 1. The system anonymity decreases as the number of compromised tags increases.
When a tag is compromised, the adversary will have the unique key, group keys, and random numbers. Since RSLA takes a random shift at each level of skip lists, any pair of two tags cannot be linked unless they have all their group keys in common. Let $T_c$ be the set of compromised tags, and $N_c (1 \leq N_c \leq N-1)$ be $|T_c|$. A compromised tag $t$ in $T_c$ always has $|S_t| = 1$. On the other hand, we can obtain $|S_{t'}|$ which an uncompromised tag $t' (t' \notin T_c)$ belongs to as follows. By the definition of search over skip lists, the nodes in the lowest level are divided into a number of groups with each having $k$ nodes (which is similar to branches in a balanced tree). If there is at least one compromised tag $t$ in $T_c$ that has all group keys in common as Tag $t'$, the adversary knows the group that $t'$ belongs to. Let $N_g (0 \leq N_g \leq k-1)$ be the number of compromised tags in the same group as $t'$. We can derive two cases as follows.

1. If $\exists t \in T_c$ s.t. $GK_t = GK_{t'}$, $|S_{t'}| = k - N_g$.

2. If $\forall t \in T_c$ $GK_t \neq GK_{t'}$, $|S_{t'}| = N - N_c$.

Note that $P[\exists t \in T_c \ s.t. \ GK_t = GK_{t'}]$ is $1 - (1 - \frac{1}{k^{N_c}})^N$ and thus is very small. By computing an anonymous set size for each tag, we can obtain the system anonymity by Equation 2.7.

**Example** Assume tag 3 is compromised in Figure 2.5, where $N = 8$ and $k = 2$. Tag 2’s anonymous set size will be 1 in Case 1, and 7 in Case 2. The anonymous set size of other tags (Tag 0, 1, and 4 to 7) will be 2 in Case 1, and 7 in Case 2.

Next, we formulate the average anonymous set size of a tag when $N_c$ tags are compromised. Let us say $\sigma = (1 - \frac{1}{k^{N_c}})^{N_c}$. The expected anonymous set size of an uncompromised tag $E[|S|]$ is computed by Equation 2.8.
\[ E[|S|] = (1 - \sigma)(k - E[N_g]) + \sigma(N - N_c) \] (2.8)

Here, \( E[N_g] \) can be obtained by

\[ E[N_g] = \sum_{i=0}^{k-1} (k - i) \binom{N_c}{i} \left( \frac{k}{N} \right)^i \left( \frac{N - k}{N} \right)^{k-1-i}. \] (2.9)

### 2.3.4 Qualitative Security Analysis

In this subsection, we analyze how RSLA achieves the security/privacy requirements.

**Privacy** - Privacy of tags preserved by encrypting data with a tag’s unique key.

**Untraceability** - With key-updating mechanism and nonces by the reader and a tag, the result of a tag’s reply changes from time to time. Hence, adversaries cannot distinguish two different replies from the same tag by one-way properties of a hash function with different keys. Therefore, adversaries cannot track tags.

**Cloning attack resistance** - In this attack, an adversary obtains a tag’s reply and then sends it to a reader, i.e., cloning tag’s reply. Similar to existing works, the use of nonces by the reader and a tag avoids cloning attacks.

**Forward security** - This requirement prevents an adversary from obtaining the contents in the previous interrogations by the current keys of a compromised tag. Our key-updating mechanism guarantees the forward security, since adversaries cannot deduce the old key from the current key of compromised tags.
2.4 Performance Evaluation

To evaluate the performance of the proposed RSLA, simulations are conducted with existing solutions, including static tree [29], SPA [41], group-based [33], and AnonPri [34].

2.4.1 Simulation Configuration

In the simulations, an RFID system contains one RF reader and a number of tags. The number of tags is set to be 4096, or ranges from 256 ($2^8$) to 16384 ($2^{14}$) if specified. During simulations, $N_c$ tags are randomly selected as being compromised, where $N_c$ ranges from 0 to 512.

The parameters for each protocol is set to be as follows. Unless specified, the balancing factor $k$ in RSLA is set to be 2, i.e., the skip lists behave like a balanced binary tree. For fair comparison, the static tree and SPA is implemented with a balanced binary tree. In group-based protocols, the size of each group is 64, which is the same setting as [34]. For AnonPri, the size of pseudo ID pool in the system and the number of pseudo IDs that each tag has are set to be 1000 and 10, respectively. In addition, we assume that AnonPri always succeeds, i.e., we initialize key issuing to guarantee that a tag shares its pseudo ID with at least two members in the group.

We consider three scenarios, static systems, dynamic systems, and the optimization of skip lists.

**Static Systems** - In the static system scenario, tags do not update their keys. To assess the degree of privacy, the system anonymity is computed under the assumption that the adversary obtains the unique key as well as all the group keys from the compromised tags. In addition, the singulation efficiency and cost are measured by
the average authentication speed for a tag and the number of keys in the system, respectively. Authentication speed is defined as the number of executions of hash and encryption functions. Note that SPA provides a key updating mechanism, and the other parts are the same as the static tree. Therefore, we compared our RSLA without the key-updating with the static tree, the group-based, and AnonPri.

Dynamic Systems - In the dynamic systems scenario, tags periodically update their keys. First, \( N_c \) tags are randomly selected as being compromised. Second, another set of \( N_c \) tags is randomly selected to update their keys. This process is repeated to reflect the dynamic nature of the system. The system anonymity is measured each time before tags update their keys, since the system is more vulnerable when tags are just compromised. Note that static tree, group-based, and AnonPri do not have a key updating mechanism, and thus we exclude them from consideration in this scenario.

The Optimization of Skip Lists - To investigate how the balancing factor \( k \) affects the performance, we conducted simulations of RSLA with the \( k \)-balanced skip lists, where \( k = 2, 4, 8, 16 \).

2.4.2 Simulation Results of Static Systems

Figure 2.9 illustrates the system anonymity with respect to the number of compromised tags. Clearly, RSLA achieves much higher anonymity than other protocols, and significant improvement from the existing solutions can be seen. As indicated in [46], the anonymity of AnonPri and the group-based protocol is similar.

Figure 2.10 demonstrates the authentication speed with respect to the number of tags. Since skip lists and a tree structure run in \( \log_k N \), both RSLA and the static tree can quickly singulate a tag. In addition to computing a hash function, RSLA
is required to decrypt a random number at each level of skip lists, and so it incurs slightly higher overhead compared with the static tree. In contrast, AnonPri and the group-based protocol take a much longer time for authentication as the scale of the system increases.
Figure 2.11 presents the number of unique keys and group keys in the system. RSLA has the same amount of key storage cost as the static tree, since the construction of our skip lists creates the same number of nodes as a balanced tree. Although AnonPri and the group-based protocol do not require as much storage cost as ours, the difference is small.

### 2.4.3 Simulation Results of Dynamic Systems

Figure 2.12 shows the anonymity of RSLA and SPA with respect to the number of compromised tags. From the figure, we can see that RSLA improves the anonymity compared with SPA, especially when a large number of tags are compromised. Therefore, we can say that our RSLA is the best alternative for a tree-based authentication protocol.

![Anonymity with different k values.](image1)

![Performance with different k values.](image2)
2.4.4 The Optimization of Skip Lists

Figure 2.13 depicts the anonymity of RSLA with different balancing factors with respect to the number of compromised tags. By Theorem 6, two tags \( t \) and \( t' \) are indistinguishable as long as \( GK_t \neq GK_{t'} \). Thus, the balancing factor \( k \) should minimize \( P[GK_t = GK_{t'}] \). As \( \eta = [\log_k N] = \log_k N + c \) where \( 0 \leq c \leq 1 \), we can derive \( P[GK_t = GK_{t'}] = \frac{1}{k^{\eta+1}} = \frac{k^{1-c}}{N} \). Since \( k \geq 2 \), the optimal value is \( k = 2 \) for high anonymity, and anonymity decreases as \( k \) increases. This figure validates our analysis.

Figure 2.14 shows the authentication time and the number of keys in the system required by RSLA with different balancing factors with respect to the number of tags. Although the balancing factor affects authentication speed, the increase of authentication time is small. On the contrary, the value of \( k \) is critical to the key storage cost. For example, the number of keys is large when \( k = 14 \). Recall in this simulation, \( N \) is set to be 4096. We have \( 14^3 < 4096 < 14^4 \) and thus skip lists must contain 4 lists. This indicates that the system has 7050 keys (2744 shared keys and 4096 unique keys). On the contrary, When \( k = 16 \), \( 16^3 = 4096 \) and the number of keys in the system will be 4368. This implies that the balancing factor has a significant impact on the authentication speed and the degree of privacy.
Chapter 3: Non-Encryption-Based Private Authentication

In this chapter, we address non-encryption-based private authentication problem. Since passive tags are computationally weak devices, encryption-based secure sigulations [47] are not practical. Instead of relying on the traditional cryptographic operations, recent works [35–37] employ physical layer techniques, i.e., jamming [48], to protect tags’ data. With this approach, tags could be securely identified without pre-exchanged shared keys.

The issue of the existing solutions, the privacy masking [35], Randomized Bit Encoding (RBE) [36], and Dynamic Bit Encoding (DBE) / Optimized DBE (ODBE) [37], is the impractical assumption. In these solutions, all the bits transmitted by a tag are masked (jammed) under the assumption of an additive channel, where the receiver can read a bit only when two bits (the data bit and mask bit) are the same. When the two bits are different, it is assumed that the receiver is unable to recover the corrupted bit. However, this assumption is too strong since a reader should be able to detect signals from two different sources. In reality, a receiver of a data bit will decode it as either 0 or 1 without knowing the bit collision. If there is a bit collision, either the signal strength of data bits from the tag is stronger than that of the jamming bits, or vice versa. In other words, depending on the location of the reader, it can either read all the data bits or all the jamming bits. Also, masking requires the
perfect synchronization between data bits and mask bits, which is difficult to achieve in practice.

In addition to this, DBE and ODBE have two drawbacks. One is encoding collision, where two different source data bits could be encoded into the same codeword. This causes the singulation process to fail. The other drawback is more serious. Tags’ data encoded by DBE or ODBE could eventually be cracked, should an adversary repeatedly listen to the backward channel (i.e., signals from a tag to a reader). This approach is called the correlation attack. Moreover, none of the aforementioned solutions protect tags against ghost-and-leech attacks, i.e., impersonation of RF tags, similar to man-in-the-middle attacks.

To tackle these issues, we put forth a new RFID architecture and a novel coding scheme for privacy protection against various adversary models. The contributions of this chapter are as follows:

- We redesign the system architecture of the non-encryption-based private tag access where an RF reader is divided into an RF activator and a TSD. The proposed architecture can be built by the current physical layer technologies, and thus our assumptions are much more practical than those of the existing solutions.

- The proposed distributed RFID architecture physically defends tags against ghost-and-leech attacks.

- We propose a novel coding scheme, named Random Flipping and Random Jamming (RFRJ), to protect the backward channel from passive adversaries, i.e., the random guessing attack, correlation attack, and eavesdropping. In our scheme,
a tag/TSD randomly flips/jams a bit in a codeword and keeps the index of the these bits in secret. RFRJ guarantees that the TSD can recover a tag’s content with one of the secrets, but an adversary cannot obtain the content of tags.

- Since the backward channel is protected by the RFRJ coding scheme, we can protect the forward channel (i.e., signals from a reader to a tag) by having an RF activator querying based on encoded data (or pseudo ID) space by RFRJ.

- We generalize the RFRJ coding scheme with the arbitrary source bits and code-word lengths. In addition, we prove the maximum information rate of our RFRJ scheme that achieves the perfect secret is 0.25.

- We conduct theoretical analyses for security of the proposed scheme, and prove that RFRJ provides perfect protection against passive attacks as long as jamming is successful.

- We evaluate our RFRJ coding scheme with the existing solutions by extensive simulations, and illustrate that the new architecture and coding scheme achieve our design goals.

**Organization:** Section 3.1 provides background knowledge for this chapter. In Section 3.2, we review existing works for non-encryption-based protocols. We design a new RFID architecture in Section 3.3, and propose the RFRJ coding scheme in Section 3.4. Generalization of the RFRJ coding scheme is discussed in Section 3.5. Security analyses are provided in Section 3.6 and simulation results are demonstrated in Section 3.7.
3.1 Preliminary

3.1.1 Physical Layer Security

Jamming is widely used for secure communications at the physical layer level, in which jamming signals corrupt receiving signals. Although this indicates that a legitimate receiver cannot decode received signals due to jamming, the full-duplex mode of wireless antennas allows the receiver to simultaneously transmit jamming signals and receive data. This can be done by canceling self-interference, in which transmitting signals interrupt receiving signals. According to [49], the current implementation can cancel self-interference up to 45 dB across 40MHz. Therefore, with jamming techniques, an eavesdropper cannot steal communications unless it is in close proximity to a jamming source node.

It is known that perfect secrecy is possible without shared secrets by degrading the signal at an eavesdropper relative to that at the legitimate receiver [38]. Thus, jamming is a physical layer security technique suitable to wireless sensor networks where encryption-based security systems are not practical due to the power constraints of sensor nodes. Dialog code [48] is proposed that provides secure communications without shared secrets for wireless sensor networks. In this scheme, each source bit is encoded to a codeword, and jamming is performed during the transmission of the codeword. To achieve this, two assumptions must be held. One is that bit level jamming is possible; the other is that an eavesdropper cannot know which bit is jammed. Their implementation with sensor motes shows that both assumptions can be held by simulating a byte as a bit.

Another application of physical layer security with jamming is the protection of medical devices. In [50], a shield is developed to intermediate all the communications
between a medical device of a patient and a reader from a doctor. A shield is capable of full-duplex communications, and protects the channel between a medical device and itself by jamming. Furthermore, the shield and the reader communicate with an encrypted channel. On detecting an unauthorized reader’s access, the shield interrupts the communication by jamming all transmitted bits. The authors implemented the shield with a small portable device that looks like a necklace, and thus eavesdropping is almost impossible since an adversary must be at a very close position to the shield. By doing this, the proposed architecture does not need to modify medical devices in the markets.

Figure 3.1: Distributed RFID systems.

3.1.2 Bit Level Jamming Models

Let $b$ be a source bit, $b_j$ be a jamming bit, and $b'$ be the outcome of a bit $b$ transmitted under jamming $b_j$. In [48], jamming channel models are categorized as follows.

Figure 3.1: Distributed RFID systems.
- **Probabilistic Flipping Model** - no matter what value $b_j$ has, the source bit $b$ flips with the probability $p_j$, i.e., $P[b' \neq b] = p_j$.

- **AND Channel Model** - the receiver will decode $b' = 1$ when either $b$ or $b_j$ is 1. Otherwise, $b' = 0$.

- **XOR Channel Model** - the receiver will decode $b' = 1$ when $b \neq b_j$. Otherwise, $b' = 0$. It is known that one-time pad in this model can achieve perfect secrecy if the jamming bits are truly random in [48].

- **General Model** - in this model, $P[b' = 0|b = 0, b_j = 0] + P[b' = 0|b = 0, b_j = 1] = 1$ and $P[b' = 0|b = 1, b_j = 0] + P[b' = 0|b = 1, b_j = 1] = 1$. The probability of that $b' = 1$ is similar. This jamming model achieves perfect secrecy, since the probability of that the receiver decodes $b' = 0$ is 0.5 whenever the jamming bits are truly random [48].

Figure 3.2: Distributed RFID system deployment.
3.1.3 Distributed RFID Systems

In the traditional RFID system, an RF reader has two components, a transmitter (i.e., query transmission/energizing tags) and a listener (i.e., listening to a tag’s reply) as shown in Figure 3.1(a), where a diamond represents the transmission function of a reader, a circle represents the listening function of a reader, and a rectangle represents a tag. The communication range of the backward channel is much shorter than that of the forward channel, and thus readers must be deployed based on the short-range backward channel to access all tags in the region as shown in Figure 3.2(a). A recent study proposes Distributed RF Sensing model [51] that employs two kinds of devices (a single RF transmitter and a number of RF listeners) for each function of a reader as shown in Figure 3.1(b). The model contributes to cost reduction of RFID system deployment. For example, in Figure 3.2, the traditional RFID system requires 9 transmitters and 9 listeners, while the distributed RFID system requires 1 transmitter and 9 listeners.

3.2 Related Works

In this section, non-encryption-based authentication protocols are reviewed.

3.2.1 Forward Channel Protection

In tree-walking-based protocols, each node is mapped to a leaf node of a binary tree comprised of the entire ID space, and a reader travels the tree in depth-first or breadth-first order by querying a prefix corresponding to an internal node in the tree. Thus, by eavesdropping the query, an adversary may obtain the tag’s ID, and at least
the tag’s ID is partially disclosed. To protect the forward channel, the blinded tree-walking protocol [16] and the randomized tree-walking protocol [52] are proposed. In the blinded tree-walking protocol, instead of querying with a prefix that could be the entire ID in the worst case, a reader sends a next ID bit to avoid sending all bits in an ID. In the randomized tree-walking protocol, each tag maintains two IDs: a read tag ID and a pseudo ID generated by manufacturers or by the tag itself. A reader traverses the tree with the prefix of a pseudo ID and tags reply with their real ID. These techniques protect the forward channel, but not the backward channel.

3.2.2 Backward Channel Protection

The most related studies to non-encryption-based protocol design are secure tree-walking-based singulations. Since tags can perform only simple functions, the protection of a tag’s reply is much more difficult than the forward channel protection. To protect the backward channel without shared secrets, the physical layer security techniques are incorporated to the private tag access [35–37]. In privacy masking [35], a tag’s reply is intentionally corrupted by mask bits, (i.e., jamming under the additive channel). However, if the data sent by a tag and the mask bits are exactly the same, an adversary successfully eavesdrops the tag’s content, called the same bits problem. RBE [36] alleviates the same bits problem by encoding by source bit to a codeword with a longer length. Nevertheless, RBE is vulnerable to the correlation attack, where an adversary listens to a tag’s reply over several interrogations and recovers the source bits from scratches. To tackle this issue, DBE and ODBE [37] utilize the dependency among the source bits during their encoding process, and the information obtained in the previous interrogation is meaningless for the current interrogation. Note that
RBE, DBE, and ODBE are used under privacy masking, and a reader composes a binary tree with pseudo IDs generated by these encoding schemes.

### 3.2.3 Ghost-and-Leech Attacks

Forward/backward channel protection techniques defend a tag’s ID from passive adversaries, but not active adversaries. Ghost-and-leech attacks [44] are one of the active attacks, in which an adversary impersonates a tag by forwarding a reader’s query to the tag and the tag’s reply to the reader. This attack is similar to the man-in-the-middle attacks in the study of cryptography. In [44], the author proposed Secret Handshake, where the user of a tag owner defines a motion signature, e.g., motion of a circle, a triangle, an alpha, etc., and unlocks the tag before a reader accesses it. However, this solution only works for the applications in which a tag is used for the owner’s identification, such as ID cards, since the motion signature must be defined for individual tags. Hence, this approach cannot be applied to RFID systems where tags are attached to products, e.g., supermarkets, library, supply chains, and more.

### 3.3 Proposed Distributed Architecture

In this section, we propose a new RFID system architecture for a secure singulation as shown in Figure 3.3.

#### 3.3.1 Assumptions

We begin with listing physical layer assumptions as follows.

- Bit level jamming is feasible.

- An eavesdropper does not know if a bit is jammed.
Figure 3.3: The proposed RFID architecture.

- Probabilistic flipping model is used for a jamming environment.

As we discussed in Section 3.1, the first and third assumptions are already implemented and validated in [48]. On the other hand, there is no implementation of the backward channel protection methods in [35–37]. Therefore, our assumptions are much more practical than the past research.

3.3.2 New RFID System Architecture

Similar to [51], an RF reader is divided into two components, an RF activator and a Trusted Shield Device (TSD). In our new architecture, an RF activator queries a tag with a long-range signal (i.e., the forward channel) and energizes the tag. A TSD receives a tag’s reply with a short-range signal (i.e., the backward channel), and it sends the reply to the activator via an encrypted channel which we define as the relay channel. In typical RFID applications, a reader forwards tags’ data to the back-end server. For simplicity, in this chapter we consider the RF activator as the final destination of a tag’s data by assuming the activator forwards collected data.
to the back-end server. A TSD works as an RF listener and it is capable of bit level jamming during reception of a tag’s reply. Therefore, our new RFID system architecture consists of three components: an RF activator, a TSD, and RF tags.

In this chapter, we introduce a new coding scheme, namely Random Flipping Random Jamming (RFRJ), for the backward channel protection. A tag will send encoded data (i.e., pseudo IDs) to a TSD under the jamming environment. This prevents adversaries from passive attacks, i.e., the random guessing attacks, correlation attacks, and eavesdropping. As we will show later, the RFRJ coding scheme ensures that adversaries cannot decode the original tag’s ID from incomplete data due to jamming while the TSD successfully recovers the data from imperfect information.

A TSD is conceptually similar to the trusted masking device in [36] and a medical device shield implemented in [50], but different in the following functions.

- On overhearing a query from an activator to a tag, a TSD jams a bit in a codeword. As mentioned in the assumption, bit level jamming is possible.

- If an unauthorized reader tries to access a tag, a TSD jams against all bits of codewords so that the unauthorized reader cannot read the content of the transmitted data. A similar function is implemented in [50], where a shield device jams the whole communication on detecting unauthorized accesses. This can be done by letting an authorized activator communicate with a TSD before a singulation process.

- Unlike the trusted masking device and medical shield, a TSD intermediates only the backward channel.

With our new architecture, we can achieve the following design goals:
- The forward channel is protected by having an activator querying tag based on the pseudo ID space encoded by the RFRJ coding scheme.

- The RFRJ coding scheme protects the backward channel against the random guessing attacks, correlation attacks, and eavesdropping, as we will show in Section ??.

- Since we assume both an activator and a TSD have computational power, the relay channel can be protected by the traditional cryptographic operations.

- The proposed architecture defends against ghost-and-leech attacks. First, an adversary cannot forward an activator’s query to a tag since a TSD blocks all unauthorized accesses. Second, an adversary cannot obtain a tag’s reply due to the jamming by TSD. Therefore, an adversary cannot impersonate a tag.

- The physical layer assumptions are much more practical than the existing solutions [35–37], as we discussed in Section 3.3.1

### 3.4 Random Flipping Random Jamming Coding

In this section, we present the Random Flipping Random Jamming (RFRJ) coding scheme.

#### 3.4.1 Definition

Let $r$ be an RF activator, $s$ be a TSD, and $t$ be an RF tag. An activator which intends to obtain data from a tag sends a query on the forward channel. When the tag replies to the TSD, it encodes every $l_b$ bits in the data into a $l_c$ bits codeword with an encoding function $E(.)$. Note that $l_b$ is not the length of an ID, but the unit
to be encoded into a codeword. A coding scheme for private tag access is defined by the parameters, $l_b$, $l_c$, and $C$. Here, $C$ is a set of codewords that could be used for encoding. During the transmission of a pseudo ID on the backward channel, the TSD conducts bit level jamming. On receiving the tag’s reply, the TSD decodes the received codeword by a decoding function $D(.)$, and forwards the data to the activator via the relay channel.

In general, we call $l_b$-to-$l_c$ the RFRJ coding scheme. For instance, the coding scheme with $l_b = 1$ and $l_c = 4$ is said to be the 1-to-4 RFRJ coding scheme. The notations utilized in this chapter are listed in Table 3.1.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>The RF Activator $r$</td>
</tr>
<tr>
<td>$s$</td>
<td>The TSD $s$</td>
</tr>
<tr>
<td>$t$</td>
<td>The RF tag $t$</td>
</tr>
<tr>
<td>$b$</td>
<td>The bit $b$</td>
</tr>
<tr>
<td>$B$</td>
<td>The source bits ${b_1, b_2, \ldots}$</td>
</tr>
<tr>
<td>$c$</td>
<td>The codeword $c$</td>
</tr>
<tr>
<td>$C$</td>
<td>A domain of codewords $C = {c_0, c_1, \ldots}$</td>
</tr>
<tr>
<td>$l_c$</td>
<td>The length of a codeword $</td>
</tr>
<tr>
<td>$l_b$</td>
<td>The length of source bits $</td>
</tr>
<tr>
<td>$I$</td>
<td>The index of a bit in a codeword</td>
</tr>
<tr>
<td>$E(\cdot)$</td>
<td>The function $E : {0, 1}^l_b \rightarrow {0, 1}^l_c$</td>
</tr>
<tr>
<td>$D(\cdot)$</td>
<td>The function $D : {0, 1}^l_c \rightarrow {0, 1}^l_b$</td>
</tr>
<tr>
<td>$H(b, b')$</td>
<td>The Hamming distance between $b$ and $b'$</td>
</tr>
<tr>
<td>$H(b, b', i)$</td>
<td>The Hamming distance between $b$ and $b'$ after removing the $i$-th bit of $b$ and $b'$</td>
</tr>
<tr>
<td>$p_j$</td>
<td>The probability that a jammed bit is flipped</td>
</tr>
</tbody>
</table>
3.4.2 Private Tag Access Protocol

The proposed private tag access protocol works as follows. Suppose an RF activator \( r \) plans to read an RF tag \( t \) without disclosing the tag’s ID to an eavesdropper. In this section, we first consider the length of the encoding unit \( l_b \) to be 1 in this chapter. Our idea can be applied to arbitrary values of \( l_b \) and \( l_c \), where \( l_b < l_c \). On receiving a request, the tag \( t \) extends a bit into an \( l_c \)-bit codeword, where \( l_c \geq 4 \) must hold. When the tag transmits data over the backward channel, it randomly selects a bit in a codeword and intentionally flips it. Note that this process is done before the tag sends out the codeword, so the data sent by the tag always contains a one-bit error. On the other hand, the TSD, which is an RF listener with jamming capability, jams a single bit in the codeword. The jamming causes the selected bit to flip. Let \( p_j \) (0 ≤ \( p_j \) ≤ 1) be the probability that the bit jammed by the TSD is flipped. We denote \( I_s \) and \( I_t \) as the indexes of the selected bits by the TSD and the tag, respectively. The TSD randomly selects any bit in the first half of the \( l_c \) bits codeword, i.e., \( 1 \leq I_s \leq \lfloor \frac{1}{2}l_c \rfloor \), while a tag randomly selects a bit in the second half of the codeword, i.e., \( \lfloor \frac{1}{2}l_c \rfloor + 1 \leq I_t \leq l_c \). By doing this, we can guarantee that the TSD and the tag do not select the same bit. Thus, the codeword received by the TSD or an eavesdropper contains a two-bit error when jamming flips the \( I_s \)-th bit and a one-bit error when jamming fails.

For instance, in Figure 3.4, a source bit is encoded into a 4-bit codeword. The tag flips the third bit in the codeword, which is colored gray, and the TSD selects the first bit for jamming, which is crossed off.

Assume the original codeword is 1010. Since the tag flips the third bit, it will send 1000 over the backward channel. Meanwhile, the TSD jams the first bit. Hence, the
TSD and the eavesdropper will receive $X000$, where $X$ could be decoded to either 0 or 1. The TSD knows $I_s$, and thus it knows one of the three bits may contain an error after excluding the jammed bit. However, the eavesdropper does not know which bit the TSD jammed or which bit the tag flipped. For the eavesdropper, two of four bits may contain errors. Thus, the TSD and the eavesdropper have a different amount of information to decode the original codeword. In general, for 1-to-$l_c$, TSD knows that there is a one-bit error out of $(l_c - 1)$ bits while the eavesdropper knows there is a two-bit error out of $l_c$ bits at best.

Both the TSD and the tag keep the indexes of the bits they jammed/flipped in secret. The TSD has one of the secrets, but the eavesdropper knows neither of them. Therefore, with the coding scheme the receiver can decode a source bit when one of the $(l_c - 1)$ bits is flipped but not when two of the $l_c$ bits are flipped. Our new system architecture and our proposed private access protocol allow for an RF activator to securely collect RF tags’ content without shared secrets.
3.4.3 The Single Bit RFRJ Coding Scheme

We propose the RFRJ coding scheme with the parameter $l_b = 1$ and $l_c = 4$. Note that $l_c = 3$ does not work and $l_c = 4$ is the most efficient in terms of communication cost, which will be shown later. Let $b$ be a source bit and $c$ be a codeword. The encoding function $E: \{0, 1\} \rightarrow \{0, 1\}^4$ is defined by $E(b) = c_0$ if $b = 0$ and $E(b) = c_1$ if $b = 1$.

The encoding function $E(.)$ must ensure that the Hamming distance between $c_0$ and $c_1$, denoted by $H(c_0, c_1)$, is four. There are 16 such $(c_0, c_1)$ pairs that can be used for private tag access. We call them \textit{valid 4-bit codeword pairs}.

\textbf{Valid 4-bit Codeword Pairs} When $l_c = 4$, a codeword pair $(c_0, c_1)$, corresponding to a source bit pair $(0, 1)$, is said to be valid when the Hamming distance between $c_0$ and $c_1$ is four, i.e.,

$$(0000, 1111), (0001, 1110), (0010, 1101), (0100, 1011), (1000, 0111), (0011, 1100), (0110, 1001), (0101, 1010),$$

and $(c_1, c_0)$.

Let $c'$ be the received codeword in which up to two bits could be flipped. We define the decoding function as $D: \{0, 1\}^4 \rightarrow \{0, 1\}$. Since a TSD knows the index of the jammed bit, the decoding function ignores the jammed bit. A tag also flips a bit which is unknown to the TSD, and the three bits contain the flipped bit after the TSD removes the jammed bit. Let $H(b, b', i)$ be the Hamming distance between $b$ and $b'$ after removing the $i$-th bit from $b$ and $b'$. $D(c')$ outputs 0 when $H(c', c_0, I_s) < H(c', c_1, I_s)$ and 1 when $H(c', c_0, I_s) > H(c', c_1, I_s)$. Note that $H(c', c_0, I_s) = H(c', c_1, I_s)$ never happens.
Next, we prove that the 1-to-4 RFRJ coding scheme successfully achieves our design goal.

**Theorem 7** When the RFRJ coding scheme with a valid codeword pair is used, the receiver can successfully decode the source bit, but the eavesdropper cannot.

**Proof:** The TSD knows the value of $I_s$, so it can exclude the $I_s$-th bit for the decoding process. Since a tag flips the $I_t$-th bit where $I_s \neq I_t$, one of the three bits is flipped. Hence, this problem is reduced to whether or not the TSD can recover the original codeword sent by the tag, even if one out of three bits contains an error, while the eavesdropper cannot do it if two out of four bits contain errors.

Let $(c_0, c_1)$ be a codeword pair and $c'$ be the codeword that the TSD and the eavesdropper receive. Since $H(c_0, c_1) = 4$, excluding the $I_s$-th bit, $H(c', c_0, I_s)$ and $H(c', c_1, I_s)$ are both three. For instance, after removing the first bit of a codeword pair $(1100, 0011)$, we have $H(100, 011) = 3$. This implies that either $c_0$ or $c_1$ must be closer to $c'$ than the other. Thus, the TSD can always decode it.

On the contrary, the eavesdropper does not know both $I_s$ and $I_t$. All valid codeword pairs have the Hamming distance of four, and the 4-bit codeword received by the eavesdropper may contain a two-bit error. This indicates that $H(c_0, c') = H(c_1, c') = 2$, and the eavesdropper cannot decode it. Therefore, the claim is true. □

**Example** Consider a bit pair $(0, 1)$ is mapped to one of a valid codeword pair, say $(c_0, c_1) = (0101, 1010)$, as shown in Figure 3.4. A tag sends a bit 1 which will be encoded to 1010, and it selects the third bit to be flipped, i.e., $I_t = 3$. Afterward, the TSD selects the first bit for jamming, i.e., $I_s = 1$. Hence, the TSD will receive $X000$. 62
Let us mark the jammed bit by $X$. Since a tag flips a bit in the second half of the codeword, $X000$ contains a one bit error. With the one bit error in the second half of $c_0$ and $c_1$, we will have $c_0 = \{X100, X111\}$ and $c_1 = \{X000, X011\}$.

Clearly, sets of possible values of $c_0$ and $c_1$ are exclusive, and hence $H(X000, c_0, I_s) = H(X000, c_1, I_s)$ never happens. Thus, the TSD can always obtain the original codeword by taking the closer Hamming distance to $X000$. The decoding function takes $c_1$, and outputs 1.

On the contrary, the eavesdropper can neither derive the original codeword nor the source bit. When two of four bits have errors, i.e., 0000, the eavesdropper cannot distinguish whether the second and fourth bits of 0101 or the first and third bits of 1010 are flipped.

### 3.4.4 The 1-to-4 RFRJ Coding Scheme

We have illustrated how the RFRJ coding scheme encodes a single source bit to a 4-bit codeword. In general, an RF tag has data with arbitrary length or a constant length ID (e.g., 96-bit defined in EPC Class1 Gen2 [12]). In this section, we elaborate on the complete 1-to-4 RFRJ coding scheme.

In real RFID applications, a tag is likely to transmit the same data, such as its ID, to a TSD several times. Should an eavesdropper continuously listen, it can recover the content of the tag response by the help of the previous interrogations (the correlation attack [37]). To avoid the attack, we incorporate dependency by using different valid codeword pairs to each source bit.
Let $b_k$ be the $k$-th source bit that a tag intends to encode. To encode $b_k$, our coding scheme employs the previous source bits, $b_{k-1}, b_{k-2}, b_{k-3},$ and $b_{k-4}$. To be specific, we use the coding table in Table 3.2, where $b_k = 0$ if $k \leq 0$.

For example, the source bits with length four, 1010, will be encoded into four codewords with each having 4 bits, i.e., 1111 1100 1001 1110.

The decoding process is basically the same, but uses different codeword pairs for each source bit. The corresponding codeword for the $b_k$-th source bit is obtained by Table 3.2. The decoding function $D(.)$ is applied to the received codeword $c'$, computes $H(c', c_0, I_s)$ and $H(c', c_1, I_s)$, and then outputs 0 or 1.

The correctness of RFRJ is given by Lemma 8 and Theorem 9.
Lemma 8 To **successfully decode** the \( k \)-th source bit, a TSD must successfully decode the \((k-1)\)-th source bit.

**Proof:** First, note that to decode the \( k \)-th source bit, a TSD must know the previous source bits, \( b_{k-1}, b_{k-2}, b_{k-3}, \) and \( b_{k-4} \), which means that the receiver must have successfully decoded the \((k-1)\)-th source bit.

The proof is by contradiction. Assume that the TSD does not know \( b_{k-1} \) but knows all \( b_{k-2}, b_{k-3}, \) and \( b_{k-4} \); then the TSD can decode \( b_k \). Let \( b_{k-4}b_{k-3}b_{k-2}b_{k-1} \) and \( b'_{k-4}b'_{k-3}b'_{k-2}b'_{k-1} \) be two possible previous bit pairs, and the corresponding valid codeword pairs are \( c = \{c_0, c_1\} \) and \( c' = \{c'_0, c'_1\} \). By the assumption, \( b_{k-4} = b'_{k-4} \), \( b_{k-3} = b'_{k-3} \), \( b_{k-2} = b'_{k-2} \), but \( b_{k-1} \neq b'_{k-1} \). To decode \( b_k \) without decoding \( b_{k-1} \), the Hamming distance between \( H(c_0, c'_0) \), \( H(c_0, c'_1) \), \( H(c_1, c'_0) \), and \( H(c_1, c'_1) \) must be more than two. However, all such codeword pairs have the Hamming distance two as shown in Table 3.2. This indicates the TSD cannot decode when one of two bits is flipped. Therefore, the TSD cannot decode \( b_k \) without decoding \( b_{k-1} \), which leads to a contradiction. This concludes the proof. \( \blacksquare \)

**Example** Consider a TSD which successfully decodes 000\( X \) for \( b_{k-4}b_{k-3}b_{k-2} \), but not \( b_{k-1} \), where \( X \) could be 0 or 1. The two possible codeword pairs used to encode \( b_k \) are \((0000, 1111)\) and \((0011, 1100)\), and their corresponding source bits are 0000 and 0001, respectively. Clearly, \( H(0000, 0011) \), \( H(0000, 1100) \), \( H(1111, 0011) \), and \( H(1111, 1100) \) are all two. Thus, the TSD cannot decode the source bit \( b_k \) without decoding \( b_{k-1} \).

**Theorem 9** A TSD can **successfully decode** all source bits encoded by the RFRJ coding scheme.
**Proof:** The proof is by induction on $k$.

**Induction base:** For the first source bit, the TSD knows the valid codeword pair since the base $b_{k-i} = 0 \ (1 \leq i \leq 4)$ as shown in Table 3.2. From Theorem 7, the TSD successfully decodes the first source bit.

**Induction step:** Assuming the TSD successfully decodes the $k$-th source bit, we need to show it can decode the $(k+1)$-th source bit. According to the RFRJ coding scheme, the TSD knows the previous bits for $k$ to $k-4$, when it decodes the $k$-th source bit. Thus, the TSD knows the valid codeword pair for the $(k+1)$-th source bit from Table 3.2. From Theorem 7, the receiver successfully decodes the $(k+1)$-th source bit. Therefore, the above claim is true.

**Theorem 10** When $l_b = 1$, the RFRJ coding scheme with $l_c = 4$ is the most efficient in terms of communication cost.

**Proof:** We can prove the above claim by showing that the encoding with $l_c = 3$ does not work. A TSD will receive a 3-bit codeword where one of which is jammed and one of which is flipped. The proof is by contradiction. Assume the RFRJ encoding with $l_c = 3$ is the most efficient in terms of communication cost, then the TSD can decode the original codeword. If the TSD was able to decode the source bit, it would be able to recover the original codeword from the two bits where one of which is flipped after removing the $I_s$-th bit from consideration. However, the Hamming distance between any pair of two bits is at most two, i.e., $H(00, 11), H(11, 00), H(01, 10)$, or $H(10, 01)$. Thus, when one of two bits is flipped, the TSD cannot recover the original codeword. This is a contradiction. The RFRJ coding scheme with $l_c = 3$ does not work. This completes the proof. ■
There are $8!$ coding tables that satisfy the property described in Lemma 8. Therefore, during initialization of an interrogation, an activator can send a query with the coding table number between $[1, 8!]$ to tags to prevent eavesdroppers from utilizing the disclosed bits from codewords in the previous interrogations.

### 3.5 Generalization of RFRJ Coding

In this section, we consider the general cases, the $l_b$-to-$l_c$ coding scheme, where $1 \leq l_b < l_c$. Let $E_{l_b,l_c}$ be an encoding function for $l_b$-to-$l_c$ coding scheme which is defined by $E_{l_b,l_c} : \{0,1\}^{l_b} \rightarrow \{0,1\}^{l_c}$, and $D_{l_b,l_c}$ be the corresponding decoding function. If two bits jamming and two bits flipping are considered, we can develop the $2-8$ coding scheme based on the $1$-to-$4$ coding scheme. However, it is not interesting. Since the information rate is defined as $\frac{l_b}{l_c} (0 < \frac{l_b}{l_c} \leq 1)$, the coding efficiency, in terms of the information rate of the $2$-to-$8$ coding scheme, is the same as that of the $1$-to-$4$ coding scheme. Therefore, the purpose of this section is to investigate the existence of any $l_b$-to-$l_c$ coding scheme such that $\frac{l_b}{l_c} > \frac{1}{4}$.

First, we need to find valid codeword sets $C$ that can be used for $E_{l_b,l_c}$. Note that we call $C$ codeword sets instead of codeword pairs, since $C$ contains more than two codewords when $l_b > 1$. In general, $|C| = 2^{l_b}$.

Intuitively, if every pair of codewords in $C$ has the Hamming distance of four, $C$ seems to be a valid codeword set. However, there is one restriction we need to enforce. Recall that a TSD jams the first half of a codeword and a tag flips the second half of the codeword to prevent the TSD and tag from selecting the same bit in the codeword. Considering this restriction, the following two properties are introduced to define a valid codeword set.
**Property 1** For a given codeword set $C$ ($|C| = 2^b$), $\forall c, c' \in C$, $H(c_1, c'_1) \geq 2$ and $H(c_2, c'_2) \geq 2$, where $c = c_1||c_2$ and $c' = c'_1||c'_2$.

**Property 2** For a given codeword set $C$ ($|C| = 2^b$), $\forall c, c' \in C$, $H(c_1, c'_1) \leq 2$ and $H(c_2, c'_2) \leq 2$, where $c = c_1||c_2$ and $c' = c'_1||c'_2$.

Property 1 is to ensure that a TSD can decode a codeword, and Property 2 is to prevent an eavesdropper from decoding under the RFRJ authentication. Note that any 4-bit codeword pairs has Properies 1 and 2. Now, we can define valid codeword sets, which can be used for the $l_b$-to-$l_c$ coding scheme.

**Valid Codeword Sets** Given $l_b$ and $l_c$, a set of codeword with the properties 1 and 2 is said to be a valid codeword set.

For example, the 2-to-6 coding scheme with Table 3.3 is valid. Consider the case that a tag replies a codeword 000000. Assume a TSD jams the first bit and the tag flips the fourth bit. If the jamming succeeds, an eavesdropper will receive 100100. All 000000, 110110, and 101101 with two bits flipping could be 100100, and thus the eavesdropper cannot decode the original codeword.

<table>
<thead>
<tr>
<th>Source bits</th>
<th>Codewords</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>000000</td>
</tr>
<tr>
<td>01</td>
<td>110110</td>
</tr>
<tr>
<td>10</td>
<td>011011</td>
</tr>
<tr>
<td>11</td>
<td>101101</td>
</tr>
</tbody>
</table>

Table 3.3: An example of the 2-to-6 coding.
While the above 2-to-6 coding scheme is more efficient because its information rate is $\frac{1}{3}$, the perfect secrecy cannot be achieved. Since $l_b = 2$, there are four source bits, i.e., 00, 01, 10, and 11. Thus, the random guessing probability by an eavesdropper must be 0.25 for the perfect secret. In general, the random guessing probability must be $\frac{1}{2^k}$. However, in the aforementioned example, on receiving 100100 the eavesdropper can guess the original codeword to be either 000000, 110110, and 101101, but exclude the possibility of 011011. Hence, the eavesdropper narrows the source bits to be 00, 01, and 11. In other words, the correct guess probability is approximately 0.33.

When an eavesdropper receives a codeword, both the first and second half of the codeword contain one bit error. If the first and second half of any pair of codewords in a valid set has the Hamming distance of one, the original codeword could be any codeword in the valid set. This indicates that eavesdropper cannot guess the original source bits with probability greater than $\frac{1}{2^k}$. To formally provide a valid codeword set for the perfect secrecy, we introduce Property 3.

**Property 3** For a given codeword set $C$ ($|C| = 2^k$), let $i$ ($1 \leq i \leq l_c$) be the index of a bit in a codeword $c \in C$. For $1 \leq i \leq \lfloor \frac{1}{2}l_c \rfloor$, $\forall c, c' \in C$, $H(c_1, c'_1, i) = 1$, and for $\lfloor \frac{1}{2}l_c \rfloor + 1 \leq i \leq l_c$, $H(c_2, c'_2, i) = 1$. Here, $c = c_1 || c_2$ and $c' = c'_1 || c'_2$.

While the 1-to-4 coding scheme has Property 3, the aforementioned 2-to-6 coding scheme does not. In fact, there is no $l_b$-to-$l_c$ coding scheme with $\frac{l_b}{l_c} > \frac{1}{4}$ that achieves the perfect secrecy. We prove this by Theorem 11.

**Theorem 11** The information rate of the $l_b$-to-$l_c$ coding scheme that achieves the perfect secrecy is at most $\frac{1}{4}$.
Proof: First, note that \( l_b < l_c < 4l_b \) must hold for the information rate being greater than \( \frac{1}{4} \). The case when \( l_b = 1 \) is proved in Theorem 10. We will prove the case of \( l_b > 1 \) by showing there is no valid codeword set with Property 3 that achieves the perfect secrecy when \( l_c < 4l_b \). The proof is by contradiction. Assume there exists such a valid codeword set \( C \) for some \( l_b \) and \( l_c \) with the information rate higher than \( \frac{1}{4} \). Let us construct a graph with \( 2^{\lfloor \frac{l_c}{2} \rfloor} \) vertices, where each vertex has a \( \lfloor \frac{l_c}{2} \rfloor \) bits length key and two vertices, say \( v_i \) and \( v_j \), are connected if the Hamming distance of their keys is one. Any key has \( \lfloor \frac{l_c}{2} \rfloor \) keys to which the Hamming distance is one, and so as a result each vertex has exactly \( \lfloor \frac{l_c}{2} \rfloor \) neighboring vertices. Hence, each vertex is equivalent from each other vertices, and there are \( 2^{\lfloor \frac{l_c}{2} \rfloor} \) vertices in the graph. According to the definition of a hypercube [53], such a graph is a \( \lfloor \frac{l_c}{2} \rfloor \)-dimensional hypercube. The flipping of one bit in the key of a vertex will lead us to one of its neighboring vertices. Let \( v_i \) be a vertex whose key is the same as the first half (or second half) of a codeword. Let \( v_j \) be a neighboring vertex of \( v_i \). Recall the first half (or second half) of a codeword received by an eavesdropper contains one bit error due to jamming (or flipping). To satisfy Property 3, the neighbor set of \( v_j \) must contain all vertices whose keys are the same as the first half (or second half) of codewords in \( C \). Since each vertex has \( \lfloor \frac{l_c}{2} \rfloor \) neighbors and there are \( 2^{l_b} \) codewords in \( C \), \( \lfloor \frac{l_c}{2} \rfloor \geq 2^{l_b} \) must hold. From the condition \( l_c < 4l_b \), we can derive \( 2l_b - 1 \geq 2^{l_b} \), which never hold for any integer \( l_b \geq 1 \). Thus, there is no valid codeword set for the perfect secrecy if \( l_b > 1 \) and \( l_c < 4l_b \). This is a contradiction. Therefore, the above claim must be true.

Example For the 1-to-4 coding scheme, the source bit could be 0 or 1, and there are two codewords with length 4 in a valid codeword set. We can map the first half
(or second half) of codewords to 2-dimensional hypercube \( \lceil \frac{l_b}{2} \rceil = 2 \) as shown in Figure 3.5 (a). The vertex pairs (00, 11) and (10,01) satisfy Property 3. Thus, any combination of them can be a valid codeword pair which is listed in Definition 3.4.3 in Section 3.4.3. Consider the case \( l_b = 2 \). Theorem 11 indicates there is no coding scheme that achieves the perfect secrecy for \( l_c < 2l_b = 8 \). When \( l_c = 6 \) or \( l_c = 7 \), either the first or second half of a codeword is of length 3. Thus, we map the first half (or second half) of codewords to 3-dimensional hypercube as shown in Figure 3.5 (b). Assume the first half of a codeword is 000. After making one bit error by jamming or flipping, the bit string refers to one of the neighbors. For instance, there is an error at the third bit, and we move to the vertex 001. An eavesdropper can guess the original first half of codeword is either 000, 011, or 101 (neighbors of the vertex 001) from the 3-dimensional hypercube. Although there are four codewords, the eavesdropper can narrow the corresponding codeword to three. Since each vertex in 3-dimensional hypercube has only three neighbors, it is impossible to find a valid codeword set for \( l_c < 8 \). A similar argument holds for arbitrary \( l_b \) and \( l_c \), where \( l_b > 1 \).

![Figure 3.5: The 2 and 3-dimensional hypercube.](image)
Therefore, the 1-to-4 coding scheme is the best in terms of the information rate and degree of security in our distributed RFRJ authentication. Note that we could develop \( i \)-to-\( 4i \) coding schemes with \( i \) bits jamming and \( i \) bits flipping by using the 1-to-4 coding scheme. However, such a discussion is trivial and less significant.

### 3.6 Security Analyses

In this section, we provide security analysis for the proposed coding scheme. Every source bit is assumed to be 0 or 1 with the same probability 0.5.

#### 3.6.1 The 1-to-4 Coding Security

Let \( X \) be a random variable that represents the number of flipped bits in a codeword. The \( I_t \)-th bit selected by a tag is always flipped with the probability 1 since this is done before the data is transmitted. On the other hand, the \( I_s \)-th bit selected by a reader is flipped with the probability \( p_j \), since the jamming does not guarantee that a target bit is flipped. In RFRJ, one or two bits in a codeword could be flipped depending on \( p_j \). The probability that the events \( X = 1 \) and \( X = 2 \) occur is obtained by:

\[
P[X = 1] = 1 - p_j \tag{3.1}
\]
\[
P[X = 2] = p_j \tag{3.2}
\]

Since \( X \) is either 1 or 2, \( P[X = 1] + P[X = 2] = 1 \). In our 1-to-4 RFRJ coding scheme, an eavesdropper cannot decode when two bits are flipped. Thus, the eavesdropper cannot decode the source bit with the probability \( p_j \). This rule is only
applied to the first source bit, but not to the $k$-th bit for $k > 1$ because it is encoded with a dependency.

Let $X_k$ be a random variable that represents the number of flipped bits in the codeword corresponding to the $k$-th source bit. Again $X_k$ could be 1 or 2. Since a valid codeword pair used for the $k$-th source bit is defined by the previous source bits, an eavesdropper must decode the $(k - 1)$-th source bit to successfully decode the $k$-th source bit. Thus, the probability that the eavesdropper can decode the $k$-th source bit is $P[X_k = 1|X_{k-1} = 1]$ with the base $P[X_0 = 1] = 1$. Although the selection of a valid codeword pair is dependent, $X_k = 1, 2$ and $X_{k-1} = 1, 2$ are independent events.

\[
P[X_k = 1|X_{k-1} = 1] = P[X = 1] \cdot P[X_{k-1} = 1]
= P[X = 1]^k
= (1 - p_j)^k
\]

Hence, an eavesdropper has a very small chance to successfully decode the $k$-th source bit when $k$ is large.

### 3.6.2 Random Guessing Attacks

When the eavesdropper cannot decode, they may guess the source bit to be either 0 or 1 with even probability (i.e., the random guessing attacks). In this subsection, we consider the security of our coding scheme against an eavesdropper with random guessing capability. When a bit flipping by jamming fails, the eavesdropper decodes with the probability 1. Otherwise, it can successfully decode with the probability 0.5 by random guessing. Let $b'$ be the bit decoded by the eavesdropper. Thus, the probability that the eavesdropper successfully decodes the source bit $b$ is given by:
Let $b_k$ and $b'_k$ be the $k$-th source bit and a bit decoded by the eavesdropper, respectively. We can obtain the probability that the random guessing succeeds at the $k$-th source bit as follows.

\[
P[b = b'] = P[X = 1] + \frac{1}{2}P[X = 2]
\]  

\[
(3.4)
\]

3.6.3 Anonymity Analysis

We will use the entropy based anonymity analysis that has been developed for coding schemes in [37]. Let $n_b$ be the length of source bits (e.g., the data or tag ID length), and $n_u$ be the number of source bits uncompromised by an eavesdropper. Then, the anonymity of the source bit is given by:

\[
\sum \frac{1}{2^{n_u \log_2(\frac{1}{2^{n_u}})}} \cdot \frac{1}{n_b} = \frac{n_u}{n_b}
\]

\[
(3.6)
\]

The average anonymity of our 1-to-4 RFRJ coding scheme is computed from the expected number of bits that an eavesdropper will decode. Let $Z$ be the random variable that represents the number of compromised source bits. We will have $n_u = n_b - E[Z]$. From Equations 3.3 and 3.6, the average anonymity is computed by:
\[
\frac{n_b - E[Z]}{n_b} = 1 - \frac{1}{n_b} \left( \sum_{k=1}^{n_b-1} kp_j(1-p_j)^k + n_b(1-p_j)^{n_b} \right) \tag{3.7}
\]

Next, we formulate the anonymity at the \(i\)-th interrogation cycle. Let \(Z_i\) be the number of compromised source bits at the \(i\)-th interrogation cycle. \(E[Z]\) in Equation 3.7 can be simplified by \(\sum_{k=1}^{n_b} kp_j(1-p_j)^k\). We can derive \(E[Z_i]\) as follows.

\[
E[Z_1] = \sum_{k=1}^{n_b} kp_j(1-p_j)^k \tag{3.8}
\]
\[
E[Z_2] = E[Z_1] + \sum_{k=1}^{n_b-E[Z_1]} kp_j(1-p_j)^k \tag{3.9}
\]
\[
E[Z_i] = E[Z_{i-1}] + \sum_{k=1}^{n_b-E[Z_{i-1}]} kp_j(1-p_j)^k \tag{3.10}
\]
\[
\approx i \sum_{k=1}^{n_b} kp_j(1-p_j)^k \tag{3.11}
\]

Therefore, the anonymity at the \(i\)-th interrogation cycle is approximately \(\frac{1 - iE[Z]}{n_b}\).

### 3.6.4 Analytical Results

According to [37], DBE and ODBE may generate the same pseudo ID from two different source IDs. Although such a possibility is very small, pseudo ID collisions cause the singulation process to fail. This is not acceptable. Contrarily, our RFRJ coding scheme does not have pseudo ID collisions by Lemma 12.

**Lemma 12** When \(B \neq B'\) where \(B\) and \(B'\) are two sets of bits, \(E(B) \neq E(B')\) always holds.

**Proof:** The proof is by contradiction. Assume there exist two sets of bits, \(B\) and \(B'\), such that \(E(B) = E(B')\), then there must exist \(E(b_k)\) and \(E(b'_k)\) where \(b_k \neq b'_k\).
and $b_{k-i} = b'_{k-i}$ for $1 \leq i \leq 4$. But, according to Table 3.2, this never occurs, since $B \neq B'$, there exists at least one bit pair $b_k \in B$ and $b'_k \in B'$ such that $b_k \neq b'_k$. This is a contradiction. Therefore, the claim must be true.

DBE and ODBE have significantly improved the performance of the privacy masking [35] and RBE [36], especially against correlation attacks. Nevertheless, both DBE and ODBE cannot completely avoid correlation attacks. Hence, eventually the source bits are cracked. According to [37], encoded 96-bit data by ODBE with codeword length 4 and $p_j = 1$ is cracked in 800 interrogation cycles. However, our RFRJ is different. One of the important results in this dissertation is that the RFRJ coding scheme perfectly protects source bits from passive attacks when $p_j = 1$. This is proved by Theorem 13.

**Theorem 13** When $p_j = 1$, RFRJ achieves perfect secrecy against passive eavesdropping, random guessing, and correlation attacks.

**Proof:** We will prove the above claim by showing that RFRJ coding results in the theoretical upper bound of anonymity and lower bound of random guessing probability.

**Eavesdropping** - the probability that an eavesdropper can obtain source bits is given in Equation 3.3. When $p_j = 1$, Equation 3.3 results in $(1 - p_j)^k = 0$. Thus, RFRJ provides perfect protection against passive eavesdropping.

**Random guessing attacks** - when all bits in a codeword are disclosed (jamming/masking fails for a codeword), an eavesdropper with the random guessing capability can decode the corresponding source bit with the probability 1. Otherwise, the source bit is successfully guessed with probability 0.5. Hence, the lower bound of the
random guessing probability is $0.5^k$ for $k$-bit data. The random guessing probability for RFRJ is provided in Equation 3.5. When $p_j = 1$, we will have $(1 - \frac{1}{2} p_j)^k = 0.5^k$. This validates that RFRJ achieves the lower bound of the random guessing probability.

Correlation attacks - the upper bound of anonymity is 1. The anonymity of RFRJ for $n_b$ bits source data is obtained by Equation 3.7. When $p_j = 1$, $P[X = 1] = 0$ and thus $E[Z] = 0$. Hence, the anonymity is 1. This holds for any $n_b \geq 1$, and encoded data by RFRJ is never cracked as long as $p_j = 1$. Thus, RFRJ avoids the correlation attacks.

Therefore, the claim is true. 

3.7 Performance Evaluation

In this section, we demonstrate the performance of RFRJ with the existing secure coding schemes for RFID backward channels, including RBE [36], DBE, and ODBE [37].

3.7.1 Simulation Configurations

We have implemented the $1 - to - 4$ RFRJ coding scheme along with RBE, DBE, and ODBE. For fair comparisons, the codeword length for RBE, DBE, and ODBE is set to be four, which results in the same control overhead as the $1 - to - 4$ RFRJ coding scheme. In this simulation, data exchanged between an RF reader and RF tags are 96-bit tag IDs. Each tag encodes its ID with an encoding scheme and transmits it. 100 RF tags are deployed in the reading range of an RF activator and TSDs. The reader executes a tree-based singulation protocol against encoded IDs. The successful
jamming rate $p_j$ varies from 0.1 to 1.0. For correlation attacks, a tag sends its ID under the RFRJ access protocol (or the privacy masking environment for RBE, DBE, and ODBE), and an eavesdropper keeps the scratches of disclosed data from previous interrogations. The number of interrogations for correlation attacks is set to be 1000. For each configuration, 1000 simulations were conducted.

![Figure 3.6: Anonymity.](image)

![Figure 3.7: Correct guess probability.](image)

### 3.7.2 Simulation Results

Figure 3.6 shows the average anonymity of a pseudo ID by different encoding schemes with respect to the successful jamming rate $p_j$. All encoding schemes except RBE achieve very high anonymity. This implies that RFRJ has a strong protection against eavesdropping. In addition, we would like to emphasize that the physical layer assumptions used in our model are weaker than those in the privacy masking environment.
Figure 3.7 illustrates the random guessing probability with respect to the successful jamming rate. Although RFRJ has a slightly higher random guessing probability than DBE and ODBE even when $p_j$ is smaller than 0.7, it already provides a very strong protection. To be specific, when $p_j = 0.5$, the random guessing probability of RFRJ is $10^{-28}$. It is clear that a random guessing eavesdropper has a very small probability of decoding the source bits.

![Figure 3.8: Time to crack tag data.](image)

Figure 3.8 demonstrates the time required to crack all source bits by the correlation attacks with respect to the successful jamming rate. It is known that data encoded by RBE, DBE, or ODBE is eventually cracked due to design faults of the schemes. Contrarily, our RFRJ perfectly protects tags’ IDs from the correlation attacks when $p_j = 1$. Note that the figure plots the results for $p_j$ up to 0.95.

![Figure 3.9: Correlation attacks $p_j=1.0$.](image)

Figures 3.9, 3.10, and 3.11 present the average anonymity of a pseudo ID by different encoding schemes with respect to the interrogation cycles for the successful jamming rate, 1.0, 0.9, and 0.8, respectively. For $p_j = 1.0$ (Figure 3.9), RFRJ always
has the maximum anonymity 1.0 because its design completely avoids the correlation attacks. This is one of the significant results of RFRJ. When \( p_j = 0.9 \), RFRJ achieves a similar anonymity to that of DBE and ODBE, and a much higher anonymity than that of RBE. When \( p_j = 0.8 \), RFRJ results in a slightly lower anonymity than that of DBE and ODBE. However, the difference is not significant.

Figure 3.10: Correlation attacks \( p_j=0.9 \). Figure 3.11: Correlation attacks \( p_j=0.8 \).

### 3.7.3 Comparisons between Analytical and Simulation Results

In this subsection, the analytical and simulation results of the 1-to-4 coding scheme are compared to validate our analyses.

Figure 3.12 shows the anonymity at the first interrogation cycle with respect to the jamming successful rate \( p_j \). Figure 3.13 illustrates the anonymity for different \( p_j \) with respect to the number of interrogation cycles. As can be seen in the figures, the
analytical and simulation results are very close to each other. Thus, the simulation results validate our security analyses.

Figure 3.12: Anonymity under different $p_j$ values.

Figure 3.13: Anonymity for correlation attacks.
Chapter 4: Data Verification

Radio Frequency Identification (RFID) has emerged as an electronic tagging technology, where RF tags are used as the unique identifier of objects. Its wide adoption significantly reduces the cost of inventory management and facilitates a number of transactions in the physical world, such as library management [29], indoor localization [8, 39], warehouse operations [54], and so on. The key to the success of RFID technology is the availability of inexpensive passive RF tags. Although passive tags do not have a power source, they can be energized by signals from RF readers and are capable of simple computations, e.g., 16-bit pseudo random generator, a collision resistant hash function, etc.

While RFID drives a number of personal and business applications, security and privacy threats are always a concern for individuals and organizations. Hence, many studies have been devoted for an RF reader to securely obtain tag IDs by private authentications [34, 37], and to verify an owner’s credential by a motion signature [44] or tag activation [55]. A securely obtained tag ID is used as a pointer to the data entry in the back-end server. However, to the best of our knowledge, there is no study on the authenticity of data. Therefore, we are interested in the data verification problem in RFID systems.
In RFID systems, the back-end database server stores the information about objects or information generated based on objects’ status. Thus, any piece of data is associated with a particular tag. A set of data \( D_T = \{d_1, d_2, ..., d_i\} \) associated with Tag \( T \) is said to be verifiable if we can prove that \( D_T \) is the information about the object referred by \( T \) and any element of \( D_T \) cannot be modified without being detected by the owner of \( T \). Therefore, we first formulate the formal definition of the data verification problem in RFID systems as follows: A challenger provides data set \( D_T \) associated with Tag \( T \), and a verifier can verify that all elements in \( D_T \) are associated with \( T \) and none of them are modified.

One of the applications is integrated RFID systems. At present, different RFID systems use different tagging systems. However, individual RFID systems will converge into a single tagging system in the near future, since any object with a tag is uniquely identifiable in an Internet-like structure. In the integrated RFID system, RFID technology is not only an identification system, but also the source of valuable information. In other words, an RFID system generates huge amount of sensitive data by reading tags.

For example, a bookstore manages its products with RF tags. Consider the scenario that the bookstore ships books to a customer or another branch via a ground transportation service. Two tagging systems are involved. One is the RFID-based inventory management system in the bookstore; the other is the system employed by the carrier which attaches tags to books for delivery services. In an integrated RFID system, only one tagging system exists.

Integrating multiple RFID systems into an exa-scale system reduces the number of tags to be deployed. However, the authenticity of data is of concern, since users
and providers of RFID-based data may belong to different organizations. Thus, data verification problem must be addressed. In this chapter, we propose a verifiable RFID system, where data generated by a semi-trusted party is verifiable in terms of the authenticity. The contributions of this chapter are as follows.

- We define the data verification problem in RFID systems.
- We model an integrated RFID system architecture, where the ownership of RF tags remains but other organizations can read these tags and generate valuable information.
- We propose a data verification protocol, called 1-1 protocol, for verifiable RFID system to ensure the authenticity of data generated by service providers.
- We generalize the proposed data verification protocol into the $m$-$n$ protocol, where $m$ clients and $n$ Service Providers (SPs) exist. The proposed general model is practical in terms of key storage cost and computational cost at each party.
- We implement a prototype of the $m$-$n$ protocol, and complete testbeds to demonstrate the feasibility of the verifiable RFID system.

**Organization:** Related works are studied in Section 4.1. In Section 4.2, verifiable integrated RFID system architecture is introduced. We propose data verification protocols for integrated RFID systems in Section 4.3. The performance of the proposed system is evaluated by computer simulations in Section 4.5 and by testbeds in Section 4.6.
4.1 Related Work

The problem of data verification in integrated RFID systems is related to verifiable database systems. In general, for a query from a client, the server provides data and its proof (i.e., the authenticator of the data). A database is said to be verifiable if a client can check that her data in the untrustworthy database server is correct in the sense that any other party can not add/delete/modify her data without being detected. As authenticated data structures, Merkle tree [56], distributed Merkle tree [57], one-way accumulators [58], skip-lists [59], and hash tables [60] are widely used. For example, in the tree-based approach, data records in the database are mapped to leaf nodes and each node maintains the authenticator for a data record.

Requirements of data verification are different from application to application. In some database systems, data records should be stored in a non-erasable and non-rewritable format to establish the irrefutable proof and accurate details of past events [61]. Li et al. [62] proposed a Merkle hash tree-based data retention and verification mechanism with write-once and read-many properties in rewritable storage media. In their tree structure, the authenticator of the root is directory updated without the authenticator of internal nodes when a leaf node is updated due to data addition.

On the other hand, in cloud computing environments, it is natural for a client to update data in a server. Banabbas et al. [63] developed a verifiable computation scheme that allows a client to efficiently update data and its proof in the database server. In verifiable data streaming [64], the order of streamed data (e.g., a client streams data to a storage server) is considered, and data verification is guaranteed in the database with an unbounded size.
The data verification problem in RFID-based database is somewhat similar to verifiable database systems, but different due to the following reasons. First, an integrated RFID system has write-once and read-many properties. Second, data is generated by reading tags, and the amount of data in the database can increase exponentially. Third, the order of data generated by an RFID system is of concern. In addition, the data verification in RFID systems differs from general verifiable database systems in the definition of verifiability. Existing verifiable database systems guarantee that the data entry in the database is not modified without the permission of its client and index of the data entries is correct. In this research, the verifiability is defined in the sense that each data entry in the database is associated with a particular tag.

4.2 Problem Formulation

4.2.1 Integrated RFID Architecture

At present, different RFID systems use different tag populations. In other words, each tag is assigned for a particular application, and more than one tag is attached to an object during its life cycle. For example, a bookstore manages its products with RF tags. Consider the scenario that the bookstore ships a book to a customer or another branch via a ground transport service. Two tags are associated to the book. One is the tag attached by the bookstore for the inventory management; the other is the tag attached by the carrier for delivery service.

Since objects with a tag are uniquely identifiable (i.e., the Internet of Things idea), in the near future, individual RFID systems will converge into a single exascale RFID system, i.e., an integrated RFID system. In the bookstore example shown in
Figure 4.1, the bookstore has the ownership of the book and the tag, and the carrier generates data by reading the tag during the delivery service. In other words, a single tagging system exists between two different RFID applications.

In our verifiable RFID systems, two kinds of parties exist, clients and service providers (SPs). A client is an organization that owns objects with an RF tag, such as supermarkets and bookstores. A client has its own RFID systems for inventory management purposes. During supply chain, its products are processed and managed by other organizations, called service providers. A SP is a semi-trusted organization that reads RF tags, generates data, and stores tags’ information in its database. Thus, the organization that owns objects with a tag is a client with respect to the organization that provides information service by reading these tags and generating data. The system architecture is illustrated in Figure 4.1.

![System Model Diagram](image)

Figure 4.1: The system model.

An organization could be either a client or a SP for a particular tag population. In addition, many organizations are involved in an integrated RFID system. Thus,
for particular tag populations, we define the \( m \)-client and \( n \)-SP model, where each client has a number of tags and they are processed by SPs.

### 4.2.2 Data Verification Problem

The data stored in the back-end server is the information about objects or information generated based on objects’ status. Thus, any piece of data is associated with a particular tag. In a traditional RFID system, tags are read and generated data is used by the same organization. An organization can always ensure that data generated by reading tags is associated with a particular tag. Thus, the authenticity of generated data has not been of concern. However, when it comes to integrated RFID systems, clients and SPs are different organizations. Therefore, the clients must be able to ensure the authenticity of data provided by SPs.

The authenticity of data is defined as the verifiability. A set of data \( D_T = \{d_1, d_2, ..., d_i\} \) associated with Tag \( T \) is said to be \emph{verifiable}, if we can prove that \( D_T \) is the information about the object referred by \( T \) and any element of \( D_T \) cannot be modified without being detected. An integrated RFID system is said to be \emph{verifiable} if a client can verify the authenticity of any data set \( D_T \) generated by SPs, where \( T \) is any tag that the client owns.

### 4.2.3 Assumptions

In addition to passive RF tag functions defined by EPC Global Gen 2 [12], tags are assumed to be able to execute the \emph{synchronization} command. That is, a tag is capable of computing a hash value of a key and updating its key, i.e., \( Key \leftarrow H(Key) \), where \( Key \) is a key and \( H(\cdot) \) is a collision resistant hash function. The synchronization technique is used in many studies [41] to prevent adversaries from tracking a tag.
An SP is semi-trusted in the sense that the SP does not physically compromise tags. For example, as defined by EPS Global Gen 2 [12], a tag has unreadable memory space by readers, where access and kill passwords are stored. We assume an SP neither changes the password of a tag nor kills a tag by physical attacks. Consider that a client, say a bookstore, ships a book with a tag to a customer via a semi-trusted ground transport service provider. Should the SP physically compromise or kill the tag, it will be penalized by the law or lose credits of the organization. Hence, there is no motivation for the SP to do such things. However, the SP could generate data without reading tags or modify information in the database. For example, the SP may generate information “Tag T is at New York” without reading Tag T to reduce operational cost of reading tags for the SP.

4.3 Data Verification Protocol

In this section, we propose a data verification protocol for verifiable integrated RFID systems. The notations used in this chapter are listed in Table 4.1.

4.3.1 Overview of Data Verification Protocol

The data verification is achieved by exchanging signatures between an SP and a tag. The proposed verification protocol consists of three phases. The first phase is system initialization, where the client generates two keys, a tag’s key \(TK\) and a reader’s key \(RK\). \(TK\) is assigned to a tag, and \(RK\) is assigned to an SP. In addition, the client provides a counter \(C = 0\) to the SP. The second phase is data generation. In this stage, the SP reads tags and generates data \(d\). Based on \(TK\), \(C\), and \(d\), the SP computes a signature for data \(d\), and by the query-and-response, the tag also
Table 4.1: Definition of notations.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_i$</td>
<td>The client $i$</td>
</tr>
<tr>
<td>$T_i$</td>
<td>The tag $T_i$</td>
</tr>
<tr>
<td>$SP_i$</td>
<td>The service provider $SP_i$</td>
</tr>
<tr>
<td>$TK$</td>
<td>Tag’s secret key</td>
</tr>
<tr>
<td>$RK$</td>
<td>SP’s secret key</td>
</tr>
<tr>
<td>$D_T$</td>
<td>Data set associated with Tag $T$</td>
</tr>
<tr>
<td>$C$</td>
<td>A counter</td>
</tr>
<tr>
<td>$N_r$</td>
<td>The random number generated by a reader</td>
</tr>
<tr>
<td>$N_t$</td>
<td>The random number generated by a tag</td>
</tr>
<tr>
<td>$H(.)$</td>
<td>A hash function</td>
</tr>
<tr>
<td>$\pi$</td>
<td>The signature of a reader</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>The signature of a tag</td>
</tr>
</tbody>
</table>

computes a signature and replies to the SP. In the third phase, the client verifies the authenticity of the data generated by the SP by the signature of the SP and tag.

First, we introduce the 1-1 protocol as a baseline for a simplified integrated RFID system, where one client and one SP exist. Then, we will propose a practical data verification protocol for the $m$-client and $n$-SP model for arbitrary $m$ and $n$ values.

### 4.3.2 The 1-1 Protocol

In the 1-client and 1-SP model, one client and one SP exist. The 1-1 protocol for the data verification in this simplified model consists of three phases, which is elaborated in the following subsections.

**System Initialization**

The client initializes the system by assigning a key $TK$ for tags and $RK$ for the SP. In addition, the client provides a counter $C$ with initial value 0 to the SP. The
tag’s secret key $TK$ is stored in unreadable memory space in the tag, and thus the SP cannot obtain $TK$ from a tag unless it physically compromises the tag.

**Data Generation**

When $SP$ accesses $T$, an RF reader is involved in the communication. For simplicity, we just say $SP$ sends a query to $T$, $T$ replies to $SP$, and so on. In the data generation phase, both $SP$ and $T$ generate a signature for the data verification. $RK$ and $TK$ are used to compute a signature, and for each interrogation, both $RK$ and $TK$ are updated by the $synch$ command. Note that in the literature [41], the synchronization command is used to update the common secret between a reader and a tag, hence the name is $synch$. $RK_i$ and $TK_i$ are computed by applying a hash function $i$ times, and the bases are $RK_0 = RK$ and $TK_0 = TK$.

For each interrogation, $SP$ reads Tag $T$ and generates $d_i$ which is the $i$-th data associated with $T$. $SP$ randomly chooses a random number $N_{r,i}$ and computes a signature $\pi_i$ for $d_i$. Note that the use of random numbers prevents the reply attack, in which an adversary clones a tag’s reply seen before. $\pi_i$ is obtained by a hash function $H(RK_i, d_i || C || N_{r,i})$, where $||$ represents the concatenation of two binary strings. Then, $SP$ sends $\pi_i$ and $N_{r,i}$ to $T$. On receiving a query, $T$ also generates a random number $N_{t,i}$ and computes a signature $\sigma_i$ by $H(TK_i, ID || \pi_i || N_{r,i})$, where $ID$ is $T$’s identifier. Then, $T$ sends $\sigma_i$ and $N_{t,i}$ to $SP$. On receiving $T$’s reply, $SP$ stores data $d_i$ and the proof $(\pi_i, \sigma_i, C, N_{r,i}, N_{t,i})$ to the database. $SP$ updates the key by $RK_{i+1} \leftarrow H(RK_i)$, and increments the counter by 1, i.e., $C \leftarrow C + 1$. Finally, $SP$ sends the $synch$ command to $T$. With the $synch$ command, $T$ computes $TK_{i+1} \leftarrow H(TK_i)$ and stores $TK_{i+1}$ in the memory. Note that the old key is overwritten and replaced by the new key. The pseudo code is given in Algorithm 5.
Algorithm 5 Data generation phase.

<table>
<thead>
<tr>
<th>SP</th>
<th>Tag</th>
</tr>
</thead>
</table>
| generates data $d_i$
| generates a random number $N_{r,i}$
| computes $\pi_i \leftarrow H(RK_i, d_i || C || N_{r,i})$ | generates a random number $N_{t,i}$
| computes $\sigma_i \leftarrow H(TK_i, ID || \pi_i || N_{r,i})$ | stores data and the signatures $d_i$, $(\pi_i, \sigma_i, C, N_{r,i}, N_{t,i})$
| updates the secret key and the counter $RK_{i+1} \leftarrow H(RK_i)$ and $C \leftarrow C + 1$ | update the secret key $TK_{i+1} \leftarrow H(TK_i)$

Data Verification

In the data verification phase, Client $V$ obtains data $D_T$ from $SP$ and verifies the data authenticity in terms that all data in $D_T$ are associated with Tag $T$. Should $SP$ modify any data or add data without reading $T$, $V$ is able to detect.

First note that $V$ knows $RK$, $TK$, and ID of $T$. $V$ requests the $i$-th data of $T$, and $SP$ replies with $d_i$ and its proof $(\pi_i, \sigma_i, C, N_{r,i}, N_{t,i})$. Based on the counter $C$, $V$ computes the keys $RK_i'$ and $TK_i'$ by applying $H(RK)$ and $H(TK)$ $i$ times, respectively. With these keys, $V$ computes two signatures $\pi_i'$ by $H(RK_i', d_i || C || N_{r,i})$ and $\sigma_i'$ by $H(TK_i', ID || \pi_i || N_{r,i})$. Then, $V$ checks whether $\pi_i'$ equals $\pi_i$ and $\sigma_i'$ equals $\sigma_i$. If so, $d_i$ is valid. Otherwise, $d_i$ is invalid. The pseudo code is provided in Algorithm 6.

4.3.3 The m-n Protocol

In this section, we propose the $m$-$n$ protocol for data verification in $m$-client and $n$-SP model. Let $V_i$ be Client $i$, and $SP_j$ be Service Provider $j$. Assume each client $V_i$ owns $l_{V_i}$ tags, and these tags could be processed by all SPs. The straightforward
Algorithm 6 Data verification phase.

<table>
<thead>
<tr>
<th>Client</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>requests the $i$-th data of Tag $T$</td>
<td>find $d_i$ of Tag $T$</td>
</tr>
<tr>
<td></td>
<td>$d_{i,(\pi_i,\sigma_i,C,N_{r,i},N_{t,i})}$</td>
</tr>
</tbody>
</table>

computes the keys and signatures

\[
\begin{align*}
RK'_i & \leftarrow \{H(RK)\}^C \\
TK'_i & \leftarrow \{H(TK)\}^C \\
\pi'_i & \leftarrow H(RK'_i, d_i)|C||N_{r,i}) \\
\sigma'_i & \leftarrow H(TK'_i, ID||\pi_i||N_{r,i})
\end{align*}
\]

if $\pi'_i = \pi_i$ and $\sigma'_i = \sigma_i$, then $d_i$ is valid
otherwise, $d_i$ is invalid

approach based on the 1-1 protocol requires the key storage cost of $n \times l_{V_i}$ for clients, $n$ for tags, and $\sum_{i=0}^{m} l_{V_i}$ for SPs. Thus, this approach is impossible, since tags can store only a few keys due to the storage constraint. For instance, in EPC Global Gen 2, 32-bit keys are used and a tag normally has less than 512 bits memory space.

Hence, we propose the $m$-$n$ data verification protocol with the key storage costs of $m + l_{V_i}$ for clients, 1 for tags, and $m$ for SPs, respectively. The proposed protocol consists of three phases, system initialization, data generation, and data verification.

System Initialization

Let $RK_{j,i}$ be the key assigned by a client $V$ to compute the signature for the $i$-th data $d_{j,i}$ generated by $SP_j$. For data $d_{j,i}$ associated with $T$ owned by $V$, Client $V$ must be able to compute the corresponding $TK_k$ and $RK_{j,i}$ from $TK$ and $RK$. Note that the number of synchronization commands applied to $RK_{j,i}$ and $TK_k$ is different, since a number of SPs may read Tag $T$. Hence, in addition to SPs, each tag $T$ needs to keep a counter $C_T$.
Each client, say \( V_i \), generates \( RK_{j,0} \) for each service provider \( SP_j \), and \( TK_0 \) for each tag \( T \). In addition, \( V \) initializes the counter \( C_{SP_j} \) to be 0 for each \( SP_j \) and the counter \( C_T \) for each \( T \). Thus, Client \( V_i \) stores \( n + l_{V_i} \) keys, SP \( SP_j \) stores \( m \) keys and \( m \) counters, and Tag \( T \) of any client stores one key and one counter.

**Data Generation**

In the \( m-n \) protocol, Tag \( T \) owned by Client \( V \) will be processed by a number of SPs, say \( SP_j \). The counter \( C_T \) and \( C_{SP_j} \) for all SPs that generated data from \( T \) are incremented. Thus, we have \( C_T = \sum_{SP_j} C_{SP_j} \).

Let \( D_{j,T} = \{d_{j,1}, d_{j,2}, ..., d_{j,i}\} \) be the data set generated by \( SP_j \) and associated with \( T \). Similar to 1-client and 1-SP model, \( SP_j \) generates the \( i \)-th data \( d_{j,i} \), generates a random number, and computes a signature \( \pi_{j,i} \) by using \( RK_{j,i} \), \( C_{SP_j} \), and \( N_{r,i} \). Then, \( SP_j \) sends the signature and the random number to Tag \( T \). When tag creates a signature, it incorporates its counter \( C_T \). The counter value will be \( C_T = k \), where \( k = \sum_j C_{SP_j} \) for all \( SP_j \) that reads \( T \) so far. On receiving the signature \( \sigma_i \), \( C_T \) and \( N_{t,i} \), \( SP_j \) stores data \( d_i \) and its proof \((\pi_{j,i}, \sigma_k, C_{SP_j}, C_T, N_{r,i}, N_{t,i})\). Finally, \( SP_j \) and \( T \) update their key and counter. The pseudo code is provided in Algorithm 7.

**Data Verification**

In the \( m-n \) protocol, Client \( V \) obtains the \( i \)-th data \( d_{j,i} \) and its signature from \( SP_j \). Since each SP updates its secret key and counter independently, even if one of SPs adds or modifies data without reading a tag, other SPs are intact.

In the data verification protocol, \( V \) first requests the \( i \)-th data associated with \( T \) to \( SP_j \), and then \( SP_j \) returns \( d_{j,i} \) and \((\pi_{j,i}, \sigma_i, C_{SP_j}, C_T, N_{r,i}, N_{t,i})\). \( V \) computes the corresponding keys \( RK_{j,i}' \) for \( \pi_{j,i} \) and \( TK_k \) for \( \sigma_k \) by applying the hash function
Algorithm 7 Data generation phase.

Service Provider $SP_j$

generates data $d_{j,i}$
generates a random number $N_{r,i}$
computes $\pi_{j,i} \leftarrow H(RK_{j,i}, d_{j,i}||C_{SP_j}||N_{r,i})$

$\sigma_{j,i} \leftarrow H(TK_{k}, ID||\pi_{j,i}||C_T||N_t,k)$

stores data and the signatures $d_{j,i}, (\pi_{j,i}, \sigma_{k,SP}, C_{SP}, C_T, N_{r,i}, N_{t,k})$

updates the secret key and the counter $RK_{i+1} \leftarrow H(RK_i)$ and $C_{SP} \leftarrow C_{SP} + 1$

$TK_{k+1} \leftarrow H(TK_k)$ and $C_T \leftarrow C_T + 1$

$C_{SP_j}$ and $C_i$ times, respectively. Here, $k$ is the number of reads by SPs, i.e., $k = \sum_{\forall SP_j} C_{SP_j}$. If $\pi_{j,i} = \pi_{j,i}$ and $\sigma_k = \sigma_k$, then the data $d_{j,i}$ is valid. Otherwise, it is invalid. The pseudo code is provided in Algorithm 8.

Algorithm 8 Data verification phase.

Client $V$
requests the $i$-th data of Tag $T$ from $SP_j$

$T_i \leftarrow d_{j,i}(\pi_{j,i}, \sigma_{k,SP}, C_{SP}, C_T, N_{r,i}, N_{t,k})$

computes the keys and signatures

$RK_{j,i}^{'} \leftarrow \{H(RK)\}^{C_{SP_j}}$

$TK_k^{'} \leftarrow \{H(TK)\}^{C_T}$

$\pi_{j,i}^{'} \leftarrow H(RK_{j,i}, d_{j,i}||C_{SP_j}||N_{r,i})$

$\sigma_k^{'} \leftarrow H(TK_{k}, ID||\pi_{j,i}||C_T||N_{t,k})$

if $\pi_{j,i}^{'} = \pi_{j,i}$ and $\sigma_k^{'} = \sigma_k$, then $d_{j,i}$ is valid
otherwise, $d_{j,i}$ is invalid

95
4.3.4 Optimization

To verify the authenticity of data, a client must compute a number of hash functions, which may take a long time. Let $N_d$ be the number of data generated from a tag. A client can request the $i$-th data generated by the $j$-th SP, and $ij <= N_d$ always holds. Without the key caching, $i \times j$ computations are required for each data verification. To save the computational cost, we propose an optimization mechanism by means of the key caching. Our key caching mechanism minimizes the computational cost with a bounded size of key caching.

Let $S$ be a set of keys, and $S_{max}$ be the number of keys that will be stored at a client for each tag. The current cache size is denoted as $|S|$. If $|S| < S_{max}$, a client simply stores the current key in the cache. When $|S| = S_{max}$, the client needs to discard the current key after data verification or replace an existing key with the new one. Note that $|S| > S_{max}$ should not happen because $S_{max}$ is the bounded size of the cache.

Each key in $S$ corresponds to the $i$-th data generated by the $j$-th SP in some ways. We define the distance between two keys, $d(key_1, key_2)$, as the number of computations to obtain $key_2$ from $key_1$ by a hash function $H(.)$. That is, $key_2$ is obtained by applying the hash function $d(key_1, key_2)$ times. If $key_2$ cannot be obtained from $key_2$, $d(key_1, key_2) = \infty$. Let $X$ be the random variable defined as $d(s_k, key)$, where $key$ is the current key and $s_k$ is in $S$. Our goal is to minimize $\sum_{i=1}^{N_d} \frac{X}{i}$. This can be done by scanning all keys in the cache. Note that the cache size is considered as a constant, as the cache size is normally very small compared to a sampling population.

If two keys $key_1$ and $key_2$ are valid, $d(key_1, key_2)$ can simply be computed by $i_2j_2 - i_1j_1$, where $key_1$ is the $i_1$-th data generated by the $j_1$-th SP and $key_2$ is the
i_2\text{-th data generated by the } j_2\text{-th SP. Assume } k\text{-th element in } S \ (1 \leq k \leq |S|) \text{ is the key for } i_k\text{-th data generated by } j_k\text{-th SP. In the proposed optimization mechanism, we first add a new key to } S. \text{ Since } S \text{ contains } |S|+1 \text{ keys at this time, we will remove one of the keys in } S \text{ so that } \sum_{i=1}^{N_d} X_i \text{ is minimized. To optimize the computational cost, we need to find the } k\text{-th key } (1 \leq k \leq |S|) \text{ such that } i_{k-1}j_{k-1} - i_{k+1}j_{k+1} \text{ is minimized, where } i_0j_0 = 0 \text{ and } i_{|S|+1}j_{|S|+1} = N_d. \text{ Note that } i_1j_1 \text{ may be } i_0j_0, \text{ and } i_kj_k \text{ may be } i_{|S|+1}j_{|S|+1}. \text{ The pseudo code is provided in Algorithm 9.}

**Algorithm 9** Key caching algorithm.

1: /* Client does following */
2:  if (|S| < S_{max}) then
3:    add the current key to \( S \).
4:  else
5:    add the current key to \( S \).
6:    find \( s_k \) in \( S \) such that \( \min_{1 \leq k \leq S_{max}} (i_{k-1}j_{k-1} - i_{k+1}j_{k+1}) \).
7:    remove \( s_k \) from \( S \).
8:  end if

4.4 Analyses

4.4.1 Valid Data Rate Analyses Against Illegal Data Access

The valid data rate is an indicator to show how well a verification protocol protects tags’ data against potential malicious SPs. Let \( p_{sp} \) be the probability that an SP is malicious, and \( p_d \) be the probability that a malicious SP illegally generates data, i.e., the SP does not obey the verifiable tag access protocol when it generates data. We denote as \( N_d \) the average number of data generated for a tag and \( N_{sp} \) the number of SPs that process a tag.
First, we analyze the valid data rate for the 1-1 protocol. In the 1-1 protocol, once invalid data is added to the data set, the other SPs cannot generate a valid signature for data generation. Let $X$ be the random variable that the $k$-th SP illegally accesses a tag first, and $Y$ be the minimum index of the invalid data generated by the $k$-th SP. Since all of the $j$-th SPs ($1 \leq k \leq X - 1$) follow the protocol, all data generated by the $k$-th SP is valid. In addition, the $i$-th data ($1 \leq i \leq Y - 1$) is valid, but the $i$-th data ($i \geq Y$) is invalid. Thus, the valid data rate can be formulated by Equation 4.1.

$$\frac{N_r E[X] + E[Y]}{N_d} \quad (4.1)$$

For simplicity, $M_{sp} = \lfloor N_{sp}p_{sp}\rfloor$ and $M_{d} = \lfloor N_{r}p_{d}\rfloor$. The expected values of $X$ and $Y$ are computed by the following Equations.

$$E[X] = \frac{1}{N_{sp}} \sum_{i=1}^{N_{sp}} \left( \frac{M_{sp}}{1} \right) \left( \frac{i}{N_{sp}} \right) \left( \frac{M_{sp} - i - 1}{N_{sp}} \right)^{M_{sp} - 1}$$

$$E[Y] = \frac{1}{N_{d}} \sum_{i=1}^{N_{d}} \left( \frac{M_{d}}{1} \right) \left( \frac{i}{N_{d}} \right) \left( \frac{M_{d} - i - 1}{N_{d}} \right)^{M_{d} - 1} \quad (4.2)$$

Next, we analyze the valid data rate for the $m$-$n$ protocol. In this protocol, even though malicious SPs illegally access to a tag, data generated by other SPs are intact. Thus, the valid data rate is independent of the random variable $X$. We deduce Equation 4.4 for the valid data rate of the data verification protocol.

$$1 - N_{sp}p_{sp} + \frac{E[Y]}{N_r} N_{sp}p_{sp} \quad (4.4)$$
4.4.2 Analyses of Computational Cost

We build an analytical model of the number of executions of a hash function in a data verification protocol. Without a key caching mechanism, a client must compute the corresponding key for a tag and an SP from the current keys. To analyze the computational cost, a random data access is considered.

Let $X_c$ be the random variable that represents the distance between a base key and the current key, and $X_n$ be the random variable that represents the distance between a base key and the next key. Assuming both current key and next key are valid, the number of computations can be obtained by $X_n$ when $X_n < X_c$ and $X_n - X_c$ when $X_n > X_c$. Since a client is assumed not to request the same data, $X_n = X_c$ should never happen. Thus, we can derive the computation cost in Equation 4.5.

\[
\frac{(X_c - 1)X_n}{N_d} + \frac{(X_d - X_c)(X_n - X_c)}{N_d} \tag{4.5}
\]

Next, we analyze the computational cost with the key caching. In our caching mechanism, the index of keys in the cache is uniformly distributed. Thus, given the size of the key cache $S_{max}$, each pair of the closest keys in the cache is distanced by approximately $N_d/S_{max}$. Therefore, the expected number of hash function computations can be obtained by Equation 4.6.

\[
\frac{1}{N_d/S_{max}} \sum_{k=1}^{N_d/S_{max}} k = \frac{N_d/S_{max} - 1}{2} \tag{4.6}
\]
4.5 Performance Evaluation

We conducted computer simulations to evaluate the performance of the proposed data verification protocols, the 1-1 protocol and $m$-$n$ protocol, along with the tree-based protocols [62].

4.5.1 Simulation Configurations

The integrated RFID system consists of 10 to 100 clients and service providers (SPs) joins. Each client has 4096 tags, and each tag is processed by multiple SPs. Each SP reads a tag and generates data 10 to 100 times. 10% to 90% of SPs are malicious and they add data without reading tag with probability $p$. The value of $p$ ranges from 0.1 to 0.9. As attack models, illegal data access and illegal data modification are considered. The illegal data addition, where a malicious SP adds data without reading a tag, causes the signatures for subsequence data to be invalid. In the illegal data modification attack, a malicious SP modifies existing data entry in the database. For each tag, randomly selected 10 SPs that read the tag and generate data during its life cycle. 1000 system realizations are generated and the average are taken as simulation results.

In this performance evaluation, the following metrics are considered.

- **Valid data rate** - during the life cycle of a tag, SPs generate data including invalid data. A client randomly accesses data on an SP, and the proof for the data may or may not be valid. Valid data rate is defined as the number of data with valid proof divided by the number of data access.
- **Number of keys** - as key storage cost, the number of keys in the system is employed, including service provider and tag keys that each organization (i.e., client or SP) maintains.

- **Computational cost** - a client has to compute the corresponding keys from the base $RK$ and $TK$ in the data verification phase. The number of hash functions applied to obtain the keys is used as computational cost.

### 4.5.2 Analytical Results

Figure 4.2 demonstrates the key storage cost of each client with respect to the number of SPs. From analyses, it is clear that a client maintains $n + l_V$ on average in the $m-n$ protocol. Here, $n$ is the number of SPs and $l_V$ is the average number of tags that clients own. On the other hand, the 1-1 protocol incurs $n \times l_V$ keys cost.

![Figure 4.2: Average key storage cost for each client.](image-url)
Figure 4.3 shows the key storage cost of each SP with respect to the number of clients. Theoretically, an SP keeps $\sum_{i=0}^{m} l_v$ keys on average in the 1-1 protocol and $m$ keys on average in the $m$-$n$ protocol, where $m$ is the number of clients. As shown in Figure 4.2 and 4.3, the $m$-$n$ protocol significantly reduces the key storage cost as indicated by the analyses.

![Figure 4.3: Average key storage cost for each SP.](image)

Figure 4.4 depicts the key storage cost with respect to the number of tags in the system. In this configuration, there are 100 clients and SPs. For the 1-1 protocol, each client and SP maintains the same number of keys, as the number of keys depends on the total number of tags in the system. With the same reasons for Figure 4.2 and 4.3, the storage cost per client (or per SP) with the $m$-$n$ protocol is much smaller than that with the 1-1 protocol.
4.5.3 Simulation Results

Figure 4.5 illustrates the valid data rate for the illegal data access attack with respect to illegal data access rate. As can be seen in the figure, even when the illegal access rate is 0.9, at least 90% of data has a valid proof in the $m-n$ protocol and the tree-based protocol. This is because SP and tag independently update their key and counter. Hence, illegal data access affects signatures computed for data generated by one SP, and other data and its proof are intact. On the other hand, in the 1-1 protocol, once an SP adds data without reading a tag, all data generated after illegal access has invalid proof. As a result, the 1-1 protocol has poor valid data rate.

Figure 4.6 presents the valid data rate for illegal data access attack with respect to the percentage of malicious SPs. In this configuration, the percentage of malicious SPs ranges from 10% to 90%, and each malicious SP illegally adds data with the probability 10%. In the tree-based protocol, once a malicious SP adds data without reading a tag, it updates the signature for all internal nodes in the key tree. This causes a client not to access any data located in the tree. Hence, the valid data
rate in the tree-based protocol drastically decreases according to the percentage of malicious SPs. With the same reason as Figure 4.5, the 1-1 protocol results in poor performance even when there are only few malicious SPs. From Figures 4.5 and 4.6, it is clear that the proposed $m$-$n$ protocol is more reliable than other protocols.

Figure 4.5: Valid data rate for different illegal access rate.

Figure 4.6: Valid data rate for different illegal access rate.

Figure 4.7 shows the valid data rate for the data modification attack with respect to the data modification rate. Note that 1-1 protocol and the $m$-$n$ protocol has the same performance against the data modification attack, and thus we omit the 1-1 protocol. In this setting, 10% of SPs are malicious and modify existing data for a tag. In the $m$-$n$ protocol, only modified data is affected, and thus the valid data rate linearly decreases as the illegal data modification rate increases. On the other hand, in the tree-based protocol, all signatures are updated in the internal nodes in the tree should one of its data for a tag be modified, and hence a client cannot obtain valid signature for any data entry for the tag in the malicious SP.
Figure 4.8 presents the valid data rate for the data modification attack with respect to the percentage of malicious SPs. In this scenario, 10 to 90% of SPs are malicious and each malicious SP modifies 10% of data in its database. If data for a tag is modified in a malicious SP, a client cannot access any data for the tag. Thus, the valid data rate of the tree-based protocol decreases as the percentage of malicious SPs increases. On the other hand, our $m-n$ protocol results in high valid data rate, since only modified data is affected and others are intact.

Figure 4.7: Valid data rate for different data modification rate.

Figure 4.8: Valid data rate for different percentage of malicious SPs.

Figure 4.9 shows the computational cost with respect to the number of data accesses for a particular tag. In this scenario, a client requests $i$-th data from $j$-th SP that processes the client’s tag, where $i$ and $j$ are randomly selected. Without the key caching, the client must compute the corresponding keys from the base keys for each data verification, which causes heavy computational cost. By the key caching mechanism, the computational cost is alleviated by 90% as shown in Figure 4.9.
Figure 4.10 presents the computational cost with respect to the number of data generated by each SP. From this figure, we can see the key caching reduces 60% of computations compared with the $m-n$ protocol. Though the computational cost linearly increases as the number of generated data increases, the key caching is still effective.

Figure 4.9: Computational cost for different number of data accesses.

Figure 4.10: Computational cost for different number of data accesses.

4.6 Implementation and Testbeds

We have implemented a prototype of the $m-n$ protocol to demonstrate the feasibility of our proposed data verification scheme in an integrated RFID system.

The prototype consists of a number of modules. The interaction between different modules are shown in Figure 4.11. The system consists of the client side and server side modules. Both the client and server have setup, data verification, and RF interface modules. The setup module initializes a SP with the secret key and a
counter. The data verification module is an implementation of Algorithm 8. The RF interface module is the program interface for an RF reader. At the client side, the RF interface module initializes the key and a counter of a tag. On the other hand, at the server side, RF interface reads a tag and executes the synchronization command. In addition, the client module has GUI.

![Diagram of program modules](image)

Figure 4.11: The program modules.

### 4.6.1 Testbed Environment

The testbed is composed of up to two computers. Note that the verification process is conducted between a client and a server. One of the computers acts as a client; the other is a server (an SP). The data generated by an SP when it reads a tag is application dependent. Thus, in this testbed, the server module generates bulk information, such as “SP reads Tag 1 at 10:00 pm”.

Three kinds of network configurations, the loop-back, LAN, and international accesses, are considered. In the loop-back configuration, both the client and server
programs are executed in the same computer (MacBook Air), and the client program accesses the server program through the loop-back address, i.e., 127.0.0.1. The client (SP) computer acts as different clients (SPs) at different times, so that the m-n model is simulated overall. For the LAN setting, a Windows PC is used as a server in IEEE 802.11g wireless LAN controlled by a wireless broadband router (Linksys WRT54GL). In the international scenario, the server (Ubuntu PC) is located at National Central University, Taiwan, and the client accesses the server from the Ohio State University, Ohio, USA. In all of the settings, MacBook Air is used as a client.

Since the purpose of this testbed is to show the feasibility of our verifiable RFID system, the response time is considered. The response time is defined as the required time for a client to request data and verify its authenticity. For each configuration, 100 experiments are performed, and the average values are collected as the results.

Note that the system initialization is an offline process. Hence, we conducted several rounds of the testbed experiment to measure the performance of the online processes.

Figure 4.12: Turn around time with the loop-back setting.

Figure 4.13: Response time with LAN and international settings.
4.6.2 Testbed Results

Figure 4.12 illustrates the response time (msec) for the loop-back setting with respect to the number of data entries in the server. Since only one SP exists in this testbed, the number of data entries for a tag is set to be up to 1000. Comp. cost and Comp. cost w/ caching refer to the required time to compute the hash value from the base key, which does not contain the network-related delay. As shown in the figure, the response time increases as the number of data entries increases. This is because more data in the server implies that the client needs to apply the hash function more times to the base tag/SP keys for the proof. According to Figure 4.12, the data verification process does not take much time. In addition, the key caching mechanism significantly reduces the response time.

Figure 4.13 presents the response time (msec) for the LAN and international settings with respect to the number of data entries in the server. For both the LAN and international configurations, the key caching slightly reduces the response time, but the reduction is not significant. This implies that the response time with the data verification is mostly dominated by the network delay. Therefore, we can conclude that the computational delay introduced from the $m$-$n$ protocol is very small, and the data verification in an integrated RFID system is feasible for real deployment in terms of the computational cost.
Chapter 5: Conclusion

Security/privacy issue in RFID is one of the most significant concerns when we deploy RFID applications to the real world. Therefore, in this dissertation, we address private tag authentication and data verification problems. The private authentication safeguards tags’ content during a singulation process and verify the authenticity of tags. Data verification verifies the authenticity of data generated by RFID-based data service. The summary of this dissertation is as follows:

First, we propose a new encryption-based private authentication scheme, namely Randomized Skip Lists-Based Authentication (RSLA). Our solution provides both high authentication efficiency and strong privacy protection mechanism. RSLA relies on skip lists, a different data structure from the existing solutions. In addition, performance and security/privacy analyses are conducted. Our simulations demonstrate that our RSLA outperforms existing solutions in terms of authentication speed and degree of privacy. Thus, the proposed scheme can be applied to a large-scale RFID system.

Second, we consider the case of the non-encryption-based private authentication. To tackle this issue, we first introduce a novel distributed RFID architecture which divides the RF reader into two parts: an RF activator and a TSD, each tailoring for a specific function of an RF reader. Then, we develop the RFRJ coding scheme,
which when incorporated with the new architecture, works against a wide range of adversaries including the random guessing attack, correlation attack, ghost-and-leech attack, and eavesdropping. The physical layer assumptions of the proposed RFID architecture and the encoding scheme are readily available. In addition, the hardware cost of the new architecture is theoretically cheaper than the existing RFID systems. Moreover, we prove that the proposed system results in the perfect secrecy when jamming successfully flips a bit.

Finally, we study the RFID data verification problem. While many RFID applications are deployed at present, each individual system uses a different tagging system. In the age of Internet of Things, we envision that these RFID systems will soon be merged into an exa-scale RFID system and use a single tagging system. In such a system, the authenticity of data must be addressed to improve the quality of the RFID-based data service. We first propose an integrated RFID system, where a number of organizations are involved and a single tagging system exists. Then, we formulate the data verification problem, in which RFID-based data is verifiable in the sense that data is associated with a particular tag. To achieve this, we design two verification protocols, the 1-1 and the \( m-n \) protocols. In addition, analytical models are built and computer simulations are conducted to measure the degree of security and performance of a verifiable integrated RFID system. Moreover, we have implemented a prototype of the \( m-n \) model. From the testbeds for different network configurations, we conclude that the proposed verifiable RFID system is highly feasible.

Our work has both theoretical and practical significance. Our results can be applied to the existing RFID applications to enhance the degree of security, reliability, and scalability, with no sacrifice of the system performance. Moreover, our new RFID
architectures, the distributed and integrated ones, could be the catalyst of a number of new business and personal RFID applications. We believe the proposed protocols and architectures will serve as the foundation of the next-generation RFID systems.
Bibliography


