Improving Security and Privacy in Large-Scale RFID Systems

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Master of Engineering in the Graduate School of The Tokyo Metropolitan University

By

Yudai Komori, B.S.

Graduate Program in Department of
Information and Communication Systems

The Tokyo Metropolitan University

2018

Dissertation Committee:

Satoshi Fukumoto, Advisor

Kazuhiko Iwasaki

Kiyoshi Nishikawa

Abstract

Radio Frequency Identification (RFID) technologies lay in the very heart of Internet of Things (IoT), in which every physical objects are tagged and identified in an internet-like structure. High performance and privacy-preserving interrogations of individual tags, generally called private tag authentication, is crucial for effective monitoring and management of a large number of objects with RFID tags. 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 two important security and privacy issues for large-scale RFID systems.

First, we discuss the private tag authentication problems. In a singulation process, an RF reader first sends a query and energizes an RF tag, and then the tag replies its ID or data to the reader. As the tag's ID itself is sensitive information, the reply from tags must be protected against various threats, such as eavesdropping

and compromise attacks, where tags are physically tampered and the keys associated with compromised tags are disclosed to adversaries. Fast and secure object identification, generally called private tag authentication, is critical to efficiently monitor and manage a large number of objects with Radio Frequency Identification (RFID) technologies. In a singulation process, an RF reader queries an RF tag, and then the tag replies its ID or data to the reader. Since the tags ID itself is private information, the reply must be protected against various threats, such as eavesdropping and compromised attacks, where tags are physically tampered and the keys associated with compromised tags are disclosed to adversaries. Hence a large amount of efforts have been made to protect tags replies with low-cost operations, e.g., the XOR operation and 16-bit pseudo random functions (PRFs). In the primitive solution, a tag sends a hashed ID, instead of its real ID, to a reader, and then, the reader searches the corresponding entry in the back-end server. While this approach defends tags replies against various attacks, the authentication speed is of O(N), where N is the number of tags in the system. Hence, such a straightforward approach is not practical for large-scale RFID systems. In order to efficiently and securely read tags content, private authentication protocols with structured key management have been proposed. In these schemes, each tag has its unique key and a set of groups keys. Groups keys are shared by several tags and used to confine the search space of a unique key. With efficient data structures, the tag authentication completes within O(log k N). However, private authentication protocols with structured key management unfortunately reduce the degree of privacy, should some tags in the system be compromised. This is because group keys are shared by several tags, and physical tampering of some tags makes the other tags less anonymous. How to remedy this issue is equivalent

to reducing the probability that two tags share common group keys (hence after we refer to it as the correlation probability). The introduction of random walking over a data structure, e.g., randomized tree-walking and randomized skip-lists, significantly reduces the correlation probability. Nevertheless, two tags are still correlated should they have same groups keys at all the levels of in a balanced tree or skip lists. In our study, we design a private tag authentication protocol, namely Randomized Skip Graphs-Based Authentication (RSGA), in which unique and group keys are maintained with a skip graph. The RSGA achieves lower correlation probability than the existing scheme while maintaining the same authentication speed as the tree structure.

Second, we discuss the fast and secure grouping problems. In the large-scale RFID systems, categorization and grouping of individual items with RF tags are critical for efficient object monitoring and management. For example, when tags belonging to the same group share a common group ID, the reader can transmit the same data simultaneously to the group ID, and it is possible to save considerably the communication overhead as compared with the conventional unicast transmission. To this end, Liu et al. recently propose a set of tag grouping protocols, which enables multicast-like communications for simultaneous data access and distribution to the tags in the same group. In the reality, not only the performance issue, but also security and privacy-preserving mechanisms in RFID protocols are important for protecting the assets of individuals and organizations. Although a number of works have been done for protecting tag's privacy, to the best of our knowledge, the problem of private tag grouping is yet to be addressed.

To address the problem of private tag grouping in a large-scale RFID system, we first formulate the problem of private tag grouping and define the privacy model based on the random oracle model. As a baseline protocol, we design a private traditional polling grouping (PrivTPG) protocol based on traditional tag polling protocol. Since PrivTPG is a straightforward approach, it can take a long time. Hence, based on the idea of broadcasting group IDs, we propose a private enhanced polling grouping (PrivEPG) protocol. To further improve the efficiency of tag grouping, we propose a private Bloom filter-based grouping (PrivBFG) protocol. These protocols broadcast unencrypted group IDs. Therefore, we propose a private Cuckoo filter-based polling grouping (PrivCFG) protocol, which is a more secure protocol using a data structure called a cuckoo filter.Then, the protocol-level tag's privacy of the proposed PrivTPG, PrivEPG, PrivBFG, and PrivCFG is proven by random oracles. In addition, computer simulations are conducted to evaluate the efficiency of the proposed protocols with different configurations.

This is dedicated to my family for their love, support, and trust.

Acknowledgments

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

I would like to thank my advisor, Professor Satoshi Fukumoto for useful discussions. I wish to thank Professor Kazuya Sakai. I learned form 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.

I am also grateful to laboratory members. Thanks to them, my student life has become very fulfilling.

There were great helps from people outside of The Tokyo Metropolitan University. I would like to express my gratitude to Professor Masayuki Arai at The Nihon University and Professor Mamoru Ohara at The Tokyo Metropolitan Industrial Technology Research Institute for their help for my research.

Vita

July 12, 1993 ..... Born - Saitama, JPN

2015 ..... B.S. Engineering

Publications

Research Publications

Y. Komori, K. Sakai, S. Fukumoto "Randomized Skip Graph-Based Authentication for Large-Scale RFID Systems". *WASA*, Vol. 9798, 2016, pp. 112

Y. Komori, K. Sakai, S. Fukumoto "RFID Tag Grouping Protocols Made Private". *DSN*, June 2017

Y. Komori, K. Sakai, S. Fukumoto "Fast and secure tag authentication in large-scale RFID systems using skip graphs". *Computer Communications*, Vol. 116, January 2018, pp. 7789

Fields of Study

Major Field: Department of
Information and Communication Systems

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Background

Radio Frequency Identification (RFID) technologies lay in the very heart of Internet of Things (IoT), in which every physical objects are tagged and identified in an internet-like structure. 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 (ex. distribution management, supermarket, public transport, etc...) in the real world.

1.1.1 RFID Technologies

The RFID system consists of an RF tag, an RF reader, and a back-end server. Communications between RF reader and backend server uses encryption method as used for the existing Internet, so we will not discuss it in this paper. RF tags are classified either active or passiv. Active tags have a transmission module and a battery for data transmissions. On the other hand, passive tags do not have a power source, and have very weak calculation power. However, since passive tags are very inexpensive, mass production is possible. Hence, most RFID applications employ passive tags, in this dissertation, we consider RFID systems with passive tags.

1.2 Contributionof This Dissertation

In this dissertation, we propose solutions to private tag authentication and dgrouping 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 graph. Unlike the existing solutions, the proposed protocol provides strong privacy protection in keeping with high performance.
    2. We quantitatively and qualitatively analyze the new authentication protocol based on skip graph.

- Contributions of Chapter 3

    1. We introduce the grouping problem and fast grouping protocols.
    2. We address the private tag grouping at the protocol-level, where for a given tag an adversary cannot identify the tags group.
    3. We design four private grouping protocols.

1.3 OrganizationofThisDissertation

The rest of the dissertation is organized as follows. In Chapter 2, we propose a new encryption-based private authentication, called Randomized Skip Graph-Based Authentication (RSGA). In Chapter 3, we study the secure grouping protocols and design private grouping protocols, namely PrivTPG, PrivEPG, PrivBFG, and PrivCFG. We conclude this dissertation in Chapter 5.

Chapter 2: Encryption-Based Private Authentication

2.1 Related Works

2.1.1 Security and Privacy in RFID Applications

In all the RFID systems, an RF reader identifies individual tags by an Aloha-based or tree-waking-based query-and-response [25]. The signal from the reader to tags is called the forward channel; the signal from the tags to the reader is called the backward channel. Defending the forward channel is relatively easy, since the queries form a reader can be easily randomized [37]. On the other hand, designing backward channel protection mechanisms is non trivial work due to the computational weakness of RF tags.

Different RFID applications require different types of the backward channel protection mechanisms. For example, Czeskis et al. [8] develop a motion signature to counteract a man-in-the-middle variant, so-called the ghost-and-leech. Saxena et al. [34] introduce tag activation, where a tag would not reply to a reader's query until it is unlocked by the smartphone of the tag's owner. Sakai et al. [30, 32] design jamming-based backward channel protection mechanism with bit encoding schemes. One of the advantages of their schemes is that no shared secret is required before interrogations. This can be achieved by the jamming-based technique, called *privacy*

masking [6]. For example, when a tag sends a 4-bit string, say 1001, the reader (or a trusted masking device) simultaneously sends a mask, say 1011. Under the assumption of the additive channel, the reader and/or an eavesdropper receives 10X1, where X denotes a corrupted bit. From the received bits 10X1, the legitimate reader with the mask information can recover the original bit string from the information about the mask bits. On the other hand, an eavesdropper without the mask information cannot recover the original bit string. By doing this, a tag can securely transmit a bit string to the reader as long as at least one bit is corrupted. Based on this idea, randomization and encoding schemes are incorporated in [30]. The protocol in [32] also relies on jamming-based environment, but works on weaker physical layer assumptions in the sense that the jamming model relies on the existing physical layer technologies. While these security mechanisms aim to protect RF tag embedded smart cards or devices, they cannot be applied to large-scale RFID systems. This is because bit-by-bit operations as well as additional devices for privacy masking are required, which are too expensive to manage a number of objects with tags. In fact, to securely identify a tag, each bit must be encoded into a codeword with length being longer than or equal to 4-bit, i.e., the communication cost increases by at least four times. Then, the each codeword must be masked by privacy masking or jamming. In addition, all the aforementioned authentication schemes assume that no tag is physically tampered.

Secure grouping protocols are proposed in [17], which ensure the indistinguishablies among tag's IDs and groups. In [5], a generic framework for detecting various anomalies and attacks, such as missing tags due to theft, cloned tags, targeted attacks, and so on. As a privacy preserving RFID protocol, a key-tag tracking algorithm to estimate the number of a small set of key tags among all the tags in the system without

disclosing tag's IDs in [20]. However, none of [17, 5, 20] provides any authentication mechanism. The most related works to this paper is the private tag authentication protocol design for the object management in large-scale RFID systems, which is elaborated on the next subsection.

2.1.2 Private Authentications Protocols

To protect the backward channel, or tag's replies, a number of authentication protocols with light-weight cryptographic operations have been proposed in the past. Weis proposes Hash-lock [36], where a tag computes a hashed ID using its unique key, and then, replies the hash to a reader. The reader communicates back-end server for searching the pair of the tag's ID and unique key corresponding to the tag's reply. The reader must search all the pairs of a tag ID and a unique key, which causes authentication to take a long time. This motivates private authentication to have structured key management.

Private authentication protocols with structured key management [14, 29, 24, 21, 38, 31, 35] use one unique key and a set of group keys. A tag's reply contains a set of hash values each of which is computed using the unique key and one of group keys. In the authentication phase, A reader first scans the group keys to confine the search space of the unique key. As shown in Figure 2.1, the tree-based authentication schemes [24, 21] use a balanced tree for the key management. The tags in the system are located at leaf nodes in the tree. Each tag obtains the unique key, denoted by sk, from its leaf node and the set of group keys, denoted by gk, at the non-leaf nodes on the path to the root. Starting from the root, the reader identifies a tag by traveling

the tree toward the corresponding leaf node. Hence, authentication speed of the tree-based protocols is $O(\log_k N)$, where N is the number of tags and k is the balancing factor of the tree. However, should some tags be compromised and group keys be correlated, the system anonymity significantly decreases.

Improving anonymity is equivalent to making the correlation probability as small as possible. To this end, Lu et al. [22] use a sparse tree, where the number of non-leaf nodes are much larger than the number of tags. However, this approach increases the height of a tree causing to unacceptable storage cost to tags. Sun et al. [35] incorporates the idea of random walking over a skip list, which is a probabilistic tree-like structure consisted of a set of lists. By taking random shift at each level of skip lists and incorporating the dependency among levels, two tags are never linked unless they have exactly the same set of group keys.

There exists a faster authentication protocol, e.g., ETAP [3] and its extended version [4], which runs in $O(1)$ by mapping hashed IDs and real IDs using a hash index. Their claim relies on that the random access of a hash index provides the $O(1)$ access. However, this is the *average* performance, and hash-based protocols may take $O(n)$ in the worst case. Hence, this direction is out of our scope.

Recently, some mutual authentication protocols for application-specific context have been proposed. In [23], Luo et al. propose Succinct and Lightweight Authentication Protocol (SLAP) as an *ultralight* RFID protocol, which does not require for tags to implement any pseudorandom function generator. While such a class of protocols is desirable for the RFID systems with low-cost tags, according to the EPC Global standard [10], tags may implement a random or pseudorandom number generator. Thus, we may still rely on the assumption of the availability of pseudorandom

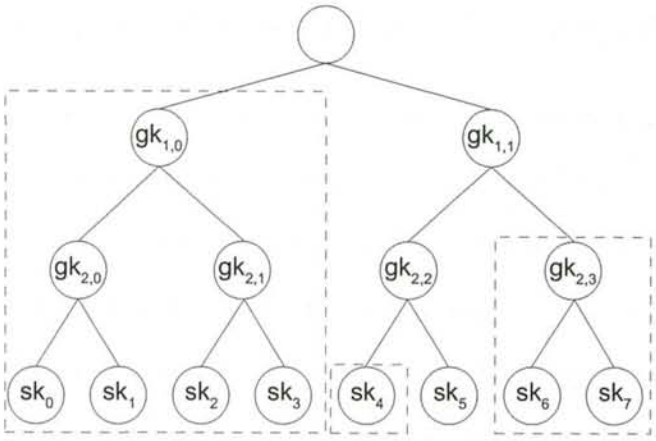


Figure 2.1: An example of tree-based protocols.

generators for protocol designs. The protocol proposed in [12] is primarily designed for healthcare applications. Farash et al. [12] improve the existing solution [40] in terms of the forward security and untraceability by applying Ecliptic curves to the RFID mutual authentication. However, computing the public key operations incurs heavy burden for tags. In [13], Gope et al. study a lightweight authentication protocol for distributed IoT applications in smart city. Their protocol not only protects the forward security, anonymity, and tag access traces, but also the location of RFID-embedded objects/devices. The works [12, 40, 13] ensure the security and privacy in application-specific environments such as medical systems and IoT applications. In contrast, our study focuses on the RFID-based large-scale object management, e.g., inventory management, RFID library, etc., using data structures for key management.

2.1.3 Cryptographic Operations for RFID systems

Since RF tags are computationally weak devices, only lightweight cryptographic operations are feasible at tags. According to the EPC standard [10], the 16-bit pseudo-random number generator shall be implemented in all passive RF tags. To this end, Poschmann et al. developes Data Encryption Standard Lightweight (DESL) [27], which requires only 1,848 gates. Using a DES type s-box and the XOR operations, a lightweight short hash function is proposed by Jappinen et al. in [15]. A pseudorandom function can be implemented with the aforementioned low-cost operations by an 8-bit shift register with an XOR feedback loop [39].

2.1.4 Physical Layer Issues in RFID Systems

EPC Global Gen 2 standard operates in the 860 MHz - 960 MHz UHF range, and the ISO18000-4 standard operates in the 2.4GHz ISM band. The chennel conditions by an RFID system significantly affect the performance of RFID protocols, which can be modeled by the product of distance-dependent average path loss law, variation in the local mean power (shadow fading), and small-scale fading. In [33], an additive model for shadow fading is proposed. Path loss and multipath propagation characteristics are studied in the UHF band in [26]. In [7], the characterization of large-scale fading is investigated in an obstacle-dense environment, a shadow depth calculation method is invented.

Since the focus of this paper is on the protocol-level security in RFID systems, a reader and tags are assumed to be able to communicate each other without frame-level errors. Therefore, the impact of channel conditions is out of our scope.

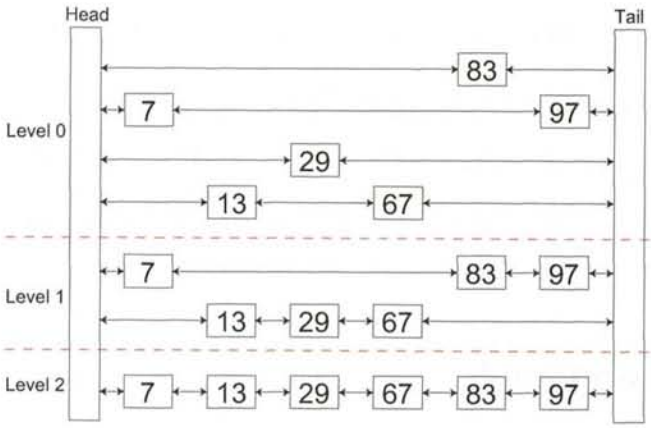


Figure 2.2: An example of skip graphs.

2.2 Preliminary

2.2.1 Anonymity

In many RFID applications, anonymity [35] is widely used as a privacy metric which quantifies the state of not being identifiable within an anonymous set. That is, the replies of the tags in the same anonymous set are indistinguishable from each other.

Consider that an RFID system with 8 tags are maintained by a binary tree structure as shown in Figure 2.1. As long as an authentication protocol is secure, the replies form any tags in the system are not distinguishable from each other. In other words, an adversary can identify a tag with probability of no greater than random guessing, i.e., 1/8, by eavesdropping the tag's reply. However, should one of the tags be compromised, anonymity of the other tags will decrease. For example, assume that tag 5 with unique key sk_5 and group keys $GK_5 = \{gk_{1,1}, gk_{2,2}\}$ is compromised.

Then, the replies from the other tags will be partially disclosed from the keys associated with tag 5. As a result, an adversary can divide the uncompromised tags into 3 disjoint sets, i.e., $\{0, 1, 2, 3\}$, $\{4\}$, and $\{6, 7\}$. These tags are anonymous within one of three sets, and thus, the adversary can identify them with probability of either 1, 0.5, or 0.25.

2.2.2 Skip Graph

A skip graph [1] is a probabilistic data structure, which has the full functionality of a balanced tree and consists of a set of ordered doubly linked lists as shown in Figure 2.2. In other words, a skip graph can be seen as a set of trees. Each level, except the lowest level (labeled by level 2), has one or more lists, and the lowest level has one list containing all the nodes. Every node participates to one of the lists at each level. A node in the list at level i for $i > 0$ appears at level $i-1$ with probability of $1/k$, where k is the balancing factor. Given the number of inputs N, the number of levels, denoted by η, is defined as $\eta = \lceil \log_k N \rceil$, Thus, there are $k^{\eta-j}$ lists at level j on average. The operations of search, insert, and delete are performed in $O(\log_k N)$. The space complexity is $O(N \log_k N)$.

2.3 Skip Graph-Based Authentication

2.3.1 Motivations and Basic Idea

To the best of our knowledge, the use of random walking over a data structure, e.g., RSLA [35], achieves the highest degree of privacy among the existing works. In this approach, the correlation probability depends on the number of levels and the number of internal nodes at each level. In the tree-based and skip lists-based protocols, the number of levels is defined as $\eta = \lceil \log_k N \rceil$. The number of nodes

at level i is defined as k^i. Hence, the correlation probability can be obtained by $\prod_{i=1}^{\lceil \log_k N \rceil - 1} \frac{1}{k^i}$.

One naive approach to decrease the correlation probability is the use of sparse structures [22], i.e., the internal nodes in a data structure is much larger than the number of tags. However, introducing redundant internal nodes is undesirable. The number of group keys that each tag stores increases in proportion to the number of levels of a data structure. In general, the passive tag has 512-bit memory, and the length of a tag's ID is 96 bits. Assuming that the length of a unique key and a group key is 32 bits, the number of levels can be at most 13, (which can be derived by $(512 - 96)/32 = 13$. The number of tags which can be supported by this approach is much smaller than 2^{13} when the structure is sparse. Therefore, increasing the number of levels of a key structure is not practical for large-scale RFID systems.

To tackle this issue, we propose Randomized Skip Graph-based Authentication (RSGA), which runs in $O(\log_k N)$. The overview of the proposed RSGA is as follows. First, a skip graph with $\eta + 1$ levels is deterministically constructed in which unique keys are located at the nodes in the list at level η and the group keys are located at the nodes in the list at level j $(1 \leq j \leq \eta - 1)$. Then, each tag is associated with a node of the lowest level list. Starting from the bottom, a tag obtains the unique key and a set of group keys along the path toward the top level list. At each level, random shifting is performed (hence, the protocal is *randomized*). In the singulation process, the reader will find the node corespoinding to a tag's reply in the lowest list by trveling from the top. The proposed RSGA differs from RSLA [35] in the initialization, key issuing, private authentication phases, and key updating. For the

Table 2.1: Definition of notations.

Symbols	Definition
k	The balancing factor of a skip graph
N	The number of tags in the system
η	The number of levels of a skip graph, $\lceil \log_k N \rceil$
$L_{j,l}$	The l-th list at level j in a skip graph ($0 \leq j \leq \eta$)
v_i, V	Node i in a list, and a set of nodes
sk_i	Tag i's unique secret key
GK_i	A set of group keys of tag i, $\{gk_1, gk_2, ..., gk_{\eta-1}\}$
R_i	A set of shift numbers of tag i, $\{r_1, r_2, ..., r_{\eta-1}\}$
ptr	The index of the list at level 1
n_t, n_r	Nonces from a tag and a reader
β, γ	A tag's reply, $\{\beta_1, \beta_2, ..., \beta_{\eta-1}\}$ and γ
$H(.)$, $E(.), D(.)$	The hash, encryption, and decryption functions

system maintenance, the similar approaches in [21, 35] can be applied to RSGA. Each phase is elaborated on in the subsequent sections.

2.3.2 Definitions and Assumptions

We assume that an RFID system consists of N tags and one reader, which is connected to the back-end server. In addition, the reader and back-end server are assumed to be connected via a secure channel. Hence, the reader is the final destination of all the tags.

The nonces are randomly selected by the reader and a tag, which are denoted by n_r and n_t, respectively. The hash function $H(.)$ is assumed to be collision resistant, and an encryption function $E(.)$ is implemented by low-cost cryptographic operations [36]. In addition, the pseudo random family (PRF) is defined as $F(.)$ which returns a 96

bits value. We assume that RF reader has enough computational power to run a decryption function $D(.)$. The symbols used in this paper is listed in Table 2.1.

2.3.3 The Proposed Authentication Protocol

Construction of A Skip Graph

Given the number of tags N and the balancing factor k, a skip graph with $\eta + 1$ levels is generated, where η is defined as $\lceil \log_k N \rceil$. There exits one list in the lowest level, which contains all the nodes, so that every tag can be allocated. At level j $(1 \leq j \leq \eta - 1)$, there are $k^{\eta - j}$ lists and each of them contains $\frac{N}{k^{\eta-j}}$ nodes. To form a list, node v_i has pointers to the right and left nodes in the same list at level j, which are denoted by $v_i.right[j]$ and $v_i.left[j]$. The left pointer of the head node and the right pointer of the tail in the list are null. Let $L_{j,l}$ be the l-th list at the j-th level from the top. For all the level, node v_i belongs to list $L_{j,l}$, where l is computed by $i \ mod \ k^{\eta-j}$. The level 0 has one list which contains node v_0, and this is used as the entry point for key searching.

Each node has a key for each level. Let $v_i.key[j]$ be the variable to store a key at node v_i. To be specific, $v_i.key[\eta]$ contains unique key sk_i and $v_i.key[j]$ $(1 \leq j \leq \eta - 1)$ contains group key $gk_{i,j}$. No key is assigned to the node in level 0 list. Thus, $v_0.key[0]$ is empty.

Since the construction of a skip graph is deterministic, our skip graph with the balancing factor k works in the same fashion to a set of k-balanced trees with the nodes in the lower levels belonging to more than one tree. For example, Figure 2.3 shows a skip graph with $N = 8$ and $k = 2$. The corresponding set of binary trees are shown in Figure 2.4.

Algorithm 1 KeyIssue()

1: /* Key Issuer does following */
2: Issuer locates all tags t to node v_i
3: /* For each tag t, Key Issuer does following */
4: **for** each tag t in the system **do**
5: KeyIssue(t, v_i)
6: /* The function to assign keys to tag t */
7: **KeyIssue**(t, v_i)
8: /* v_i is the current node */
9: $R_t = \phi$ /* Initialize the random shift numbers list */
10: $GK_t = \phi$ /* Initialize the group keys list */
11: / * At the lowest level list $L_{\eta,0}$ */
12: $sk_t \leftarrow v_i.key[\eta]$
13: /* A parent is randomly chosen among $k-1$ nodes */
14: $n \xleftarrow{uniform} [0, k-1]$
15: $v_i \leftarrow v_p$ where $p = (i - nk^{\eta-j})\ mod\ N$
16: **for** (j from $\eta - 1$ to 1) **do**
17: /* Random shifting by r and add a group key */
18: $r \xleftarrow{uniform} [0, |L_{j,l}| - 1]$
19: Add r to R_t
20: $v_i \leftarrow$ shift to the left by r
21: Add $v_i.key[j]$ to GK_t
22: /* Move to upper level */
23: $n \xleftarrow{uniform} [0, k-1]$
24: $v_i \leftarrow v_p$ where $p = (i - nk^{\eta-j})\ mod\ N$
25: $j \leftarrow j - 1$
26: $ptr \leftarrow v_i$

Key Issuing

In RSGA, every tag has four variables, the unique key sk_t, a set of group keys GK_t, a set of random shift numbers R_t, and list index ptr. Tag t is located at one of the nodes, say v_i, in the list at level η, and the unique key at $v_i.key[\eta]$ is assigned to tag t. A set of group keys are assigned to tag t by traveling with random shifting from v_i at level η to the top of the skip graph. At level j, node v_i has a set of parents,

denoted by $V_{i,j}$, since there are more than one lists at level j $(1 \leq j \leq \eta - 1)$. Here, $V_{i,j}$ includes k nodes, v_p for $p = (i - nk^{\eta-j})\ mod\ N$ $(0 \leq n \leq k - 1)$. The key issuer randomly selects one of the node in $V_{i,j}$ and moves to the upper list $L_{j-1,l}$ to which node v_i belongs at level $j - 1$. Then, random number $r_j \in [0, N - 1]$ is generated and the left shift by r_i is taken. If the pointer reaches to the head node in the list, it moves to the tail. The value of r_i is added to R_t for the j-th level. The pointer is now at a node, say $v_{i'}$, at level $j-1$. The group key at $v_{i'}.key[j-1]$ is added to GK_t. This process repeated until the pointer reaches at the top list. Unlike a tree and skip lists, there are more than one node at level 1. Thus, the entry point of the skip graph, i.e., the ID of list $L_{1,l}$ at level 1, is kept in ptr. At the end of this process, tag t has one unique key, $\eta - 1$ group keys, $\eta - 1$ shift number, and ptr. The pseudocode of key issuing is presented in Algorithm 1.

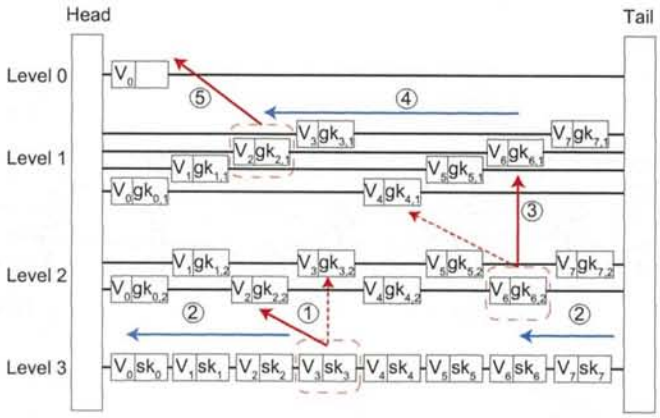
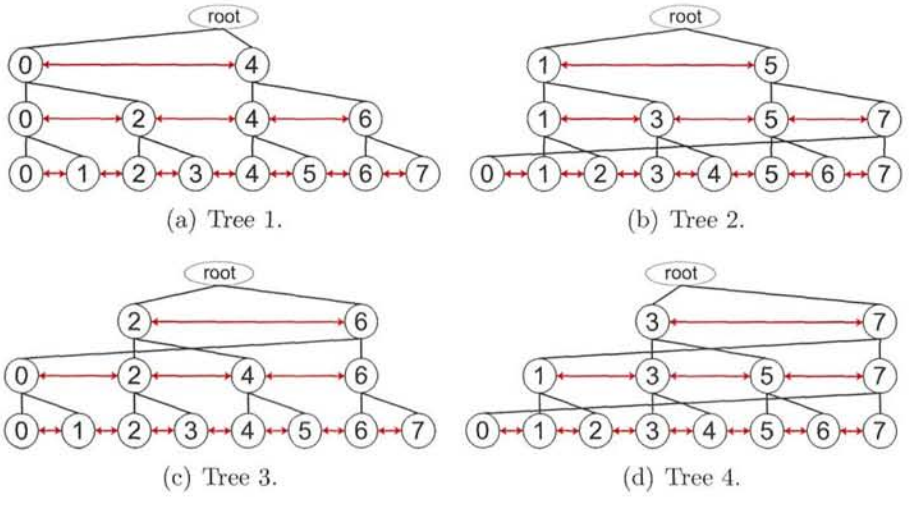


Figure 2.3: An example of key issuing.

Example Figure 2.3 illustrates a skip graph key structure with $k = 2$ and $\eta = 3$ of an RFID system with 8 tags. All tags are located at one of the nodes in the lowest level list $L_{3,1}$. The key issuer assigns group keys and random shift numbers to a tag, say tag 3, by traveling the skip graph starting from v_3 in $L_{3,1}$ to the root as follows. First, unique key sk_3 at $v_3.key[3]$ is assigned to tag 3. According to Figure 2.3, v_3 has two parents nodes v_2 and v_3 at level 2. Assume that the key issuer randomly selects $r_2 = 2$ and the random shift number is added to R_3. The current pointer shifts to the left by 2 and will be at v_6 in L_2. Next, tag 3 obtains group key $gk_{6,2}$ stored in $v_6.key[2]$. This process is repeated until the pointer reaches L_0. Assume that the key issuer selects a parent node v_6 at level 1 and tag 3 selects $r_1 = 1$ at level 1. At last, v_2 (the head node in the list to which v_6 belongs in level 1) is stored at ptr as the entry point to the skip graph. Eventually, tag 3 obtains sk_3, $GK = \{gk_{2,1}, gk_{6,2}\}$, $R_3 = \{1, 2\}$, and $ptr = v_2$.



(a) Tree 1. (b) Tree 2.

(c) Tree 3. (d) Tree 4.

Figure 2.4: The corresponding set of trees.

Mutual Authentication

After issuing keys, the reader can securely communicate with tags. In the RSGA authentication protocol, the reader first sends a query with nonce n_r, then a tag generates a reply with nonce n_t and sends the reply. The reader receives and decrypts the tag's reply.

The replying process at the tag's side is as follow. Assume tag t has the unique key sk_t, a set of group keys $GK = \{gk_1, gk_2, ..., gk_{\eta-1}\}$, a set of random shift numbers $R_t = \{r_1, r_2, ..., r_{\eta-1}\}$, and the pointer ptr. When tag t receives a query with nonce n_r from the reader, tag t generates a reply message with nonce n_t. The reply message is defined by ptr, $\beta = \{\beta_1, \beta_2, ..., \beta_{\eta-1}\}$, and γ. The value of β_j consists of a hash value $\beta_j.hash$ and encrypted shift number, i.e., $\beta_j = (\beta_j.hash, \beta_j.num)$, where $\beta.{}_jhash = H(gk_j||r_{j-1}||n_t||n_r)$ and $\beta_j.num = E(gk_j||r_j)$.

At level 1, $\beta_1.hash$ is computed with the base $r_0 = ptr$. The reason why the shift number is included at the previous level is to enforce dependency among the levels to preserve high anonymity. The random shift number r_j is encrypted by $E(gk_j||r_j)$ and set to $\beta_j.num$. While each component of β is computed using a group key at each level i, and γ is computed using a unique key at level η. The value of γ is obtained by $ID_t \oplus F(0||sk_t||r_{\eta-1}||n_t||n_r)$, where ID_t is the ID of a tag and $F(.)$ is the PRF. As the input of $F(.)$, 0 is concatenated with other parameters for the purpose of the mutual authentication. Finally, tag t sends n_t, β, γ, and ptr to the reader. Note that β contains $\eta - 1$ elements. The pseudocode of the tag's reply is provided in Algorithm 2.

On receiving tag t's reply, the reader scans k group keys associated to the nodes in list $L_{1,ptr}$. Let v_i be the node whose $key[1]$ contains the corresponding group key gk_1

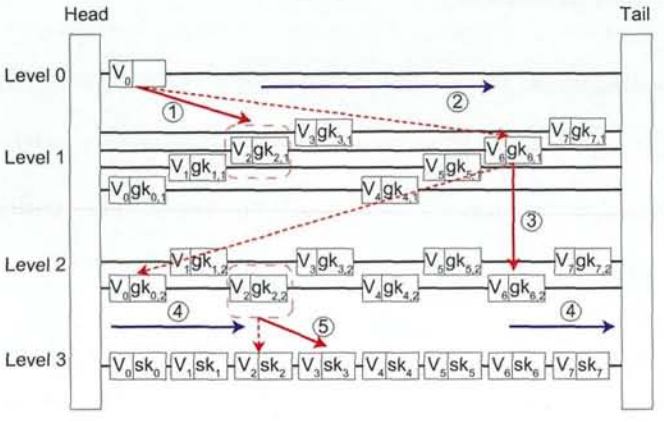


Figure 2.5: An example of authentication.

for $\beta_1.hash$. In addition, $\beta_1.num$ is decrypted by gk_1 to obtain shift number r_1. The pointer moves to v_i in $L_{j,l}$, and then right shift is token by r_1. If the pointer reaches at the tail node in the list, it moves to the head node of the same list. This process is repeated until the pointer arrives in a node in the lowest level list. Since the nodes in the lowest list contains the unique key, the reader can identify the corresponding tag based on the information in the signature γ. The pseudocode of tag authentication is given in Algorithm 3.

After the tag identification, the mutual authentication process is kicked off. Note that the mutual authentication process of the RSGA is basically the same as the exiting ones [24, 35]. At the end of singulation, the reader knows ID_t and sk_t. The reader computes $\pi = ID_t \oplus F(1||sk_t||n_t||n_r)$ and sends it to tag t. On receiving π, tag t computes ID'_t by $ID'_t = \pi \oplus F(1||sk_t||n_t||n_r)$. If ID'_t equals to tag t's ID_t, tag t accepts the reader. By doing this process, tag t also authenticates the reader.

Example Figure 2.5 presents an example of how tag 3 is authenticated by the reader. Assume that tag 3 has sk_3, $GK_3 = \{gk_{2,1}, gk_{6,2}\}$, $R_3 = \{1, 2\}$ and $ptr = v_2$. On receiving a query from the reader, tag 3 will generates nounce n_t and computes its reply, β and γ, as follows:

$$\beta_1 = H(gk_{2,1}||n_t||n_r), E(gk_{2,1}, 1) \quad (2.1)$$

$$\beta_2 = H(gk_{6,2}||1||n_t||n_r), E(gk_{6,2}, 2) \quad (2.2)$$

$$\gamma = ID_3 \oplus F(0||sk_3||2||n_t||n_r) \quad (2.3)$$

When the reader receives the tag's reply n_t, β, γ, and ptr, it tries to search the corresponding entry at the bottom list in a skip graph. As shown in Figure 2.5, there are four lists at the first level. The reader selects the one indicated by ptr. To compare the obtained hash value with $\beta_1.hash$, two nodes v_2 and v_6 in $L_{1,2}$ are scanned, each of which contains $gk_{2,1}$ and $gk_{6,1}$, respectively. Since the key $gk_{2,1}$ is the valid key for $\beta_1.hash$, the reader executes $D(gk_{2,1}, \beta_1.num)$ and obtains $r_1 = 1$. The reader moves the pointer to v_6 by taking the right shift by 1. In the level 2, two group keys at $v_0.key[2]$ and $v_6.key[2]$ are scanned to identify the valid key for $\beta_2.hash$. The reader will figure out that $gk_{6,2}$ works for $\beta_2.hash$ and obtains $r_2 = 2$ by decrypting $\beta_2.num$. This process is repeated and at the end the reader reaches the lowest level list $L_{3,1}$. At this level, the unique keys stored at $v_2.key[3]$ and $v_3.key[3]$ are scanned. The value obtained with sk_3 yields the same value as γ. The reader computes $ID' = \gamma \oplus F(0||sk_3||r_{\eta-1}||n_t||n_r)$ and validates that ID' equals to ID_3. As v_3 is the corresponding node at which tag 3 is located and ID' is tag 3's ID, the reader finally concludes that the reply comes from tag 3.

For the clarification, the pointers to the right and left nodes in the same level is used for random shifting, but nothing to to do with key scanning. When the reader moves the current pointer toward the bottom list in the authentication process, the corresponding child nodes of the current node are scanned. In this example, the current pointer is located at v_6 in level 1, and its children v_0 and v_6 in level 2. It will be clearer by seeing both Figures 2.4 and 2.5. As discussed, the skip graph in Figure 2.5 can be seen a set of 4 binary trees as shown in Figures 2.4 (a), (b), (c), and (d). Node v_6 at level 1 belongs to list $L_{1,2}$ and has two children v_0 and v_6 at level 2, which corresponds to Figure 2.4 (c). This is why the reader scans $v_0.key[2]$ and $v_6.key[2]$ in level 1. A similar argument holds for key scanning at $v_2.key[3]$ and $v_3.key[3]$ of $L_{3,1}$ in level 2.

After identifying tag 3, the mutual authentication phase is kicked off. The reader computes the proof π by $ID_3 \oplus F(1||sk_3||n_t||n_r)$ and sends π to tag 3. This time, 1 is concatenated with the input. On receiving the response from the reader, tag 3 performs $\pi \oplus F(1||sk_3||n_t||n_r)$ and resulted value will be ID_3'. Since the legitimate reader has the correct ID_3 and sk_3, the value of ID_3' computed by tag 3 shall be the same as ID_3. At the end of this process, tag 3 also authenticates the reader.

2.3.4 Key Updating Algorithm

To alleviate the compromise attack, the RFID systems should periodically update the unique and group keys. The unique keys are never shared by different tags, and so the updating of unique key is trivial. Upon accessing a tag, the reader and tag update the unique key in its key structure and the tag's memory. The challenge of designing a key updating mechanism is how to update group keys. On one hand,

Algorithm 2 ReplyToReader(n_r)

/* Assume tag t has sk_t, GK_t, R_t, and ptr */
/* where $GK_t = \{gk_1, gk_2, ..., gk_{\eta-1}\}$ and $R_t = \{r_1, r_2, ..., r_{\eta-1}\}$ */
Generate nonce n_t
for i from 1 to $\eta - 1$ **do**
  $\beta_i.hash \leftarrow H(gk_i||r_{i-1}||n_t||n_r)$ /* Note that $r_0 = ptr$ */
  $\beta_i.num \leftarrow E(gk_i, r_i)$
  Add β_i to β
$\gamma = ID_t \oplus F(0||sk_t||r_{\eta-1}||n_t||n_r)$
reply n_t, β, γ, and ptr.

Algorithm 3 Authentication(n_r, n_t, β, γ, ptr)

/* $\beta = \{\beta_1, \beta_2, ..., \beta_{\eta-1}\}$ and γ*/
$v_i \leftarrow v_0$ /* Here, v_i is the current node */
for j from 1 to $\eta - 1$ **do**
  if $1 \leq j \leq \eta - 2$ **then**
    for m from 1 to k **do**
      $v_i \leftarrow$ the m-th node in $L_{1,ptr}$.
      if $H(v_i.key[j+1]||r_{j-1}||n_t||n_r) = \beta_j.hash$ **then**
        $r \leftarrow D(v_i.key[j+1], \beta_j.num)$
        $v_i \leftarrow$ shift to the right by r
      $m \leftarrow m + 1$
  else
    for m from 1 to k **do**
      $v_i \leftarrow$ the m-th node in $L_{1,l}$
      /* Computing the signature γ */
      if $ID_t = \gamma \oplus F(0||v_i.key[\eta]||r_{\eta-1}||n_t||n_r)$ **then**
        Identify tag t by the ID_t and the unique key $v_i.key[\eta]$
        return tag t
      $m \leftarrow m + 1$
  $j \leftarrow j + 1$
  if The key is not found **then**
    return $FAIL$

SPA [21] first updates the group keys at tags, and then, the corresponding group keys at the back-end server are updated. On the other hand, RSLA [35] first updates all

the group keys in skip lists, and then, the group keys of individual tags are updated when they are accessed.

The key updating algorithm we proposed in this paper is based on one in the RSLA. First, the key issuer randomly generates nonce r and updates all the keys in the skip graph by scanning every node of each list at each level. Here, nonce r is kept in secret by the key issuer. At each node, the key issuer performs $H(r, v.key[j])$ $(1 \leq j \leq \eta)$ to generate a new key. Hence, it is computationally hard for an adversary to obtain a new key, because the new key is created by a one-way hash function. For the reader to interrogate the tags with old keys, the old keys must be kept until all the tags associated with the old keys obtain the new keys. To this end, the old key at a node, say $v.key[j]$ $(1 \leq j \leq \eta)$, is stored at $v.old\_key[j]$. As shown in Algorithm 1, when the reader accesses a tag, the tag's unique key, group keys, random shift numbers, and the list pointer are updated at the tag's side. Tags will keep only the latest set of keys and random numbers, and thus, the old ones will be discarded when they are updated. Accordingly, the proposed key updating algorithm can successfully renew the keys in the system while the reader can still access tags with the old keys.

2.3.5 Path Pruning

To quickly singulate a tag, we propose a path pruning algorithm based on the observation in which the reader does not always have to scan all the group keys. Let $GK_t = \{gk_1, gk_2, ..., gk_{\eta-1}\}$ be the set of group keys of tag t. For all the other tags, say t', if there is level j such that $\{gk_1, gk_2, ..., gk_j\} \neq \{gk'_1, gk'_2, ..., gk'_j\}$, then $\{gk_1, gk_2, ..., gk_j\}$ is the unique subset of group keys with respect to all the other tags.

This implies that the reader can singulate tag 1 without scanning gk_{j+1}, gk_{j+2}, ..., and $gk_{\eta-1}$. Let v and v' be the corresponding node at level j to group key gk_j and level η to unique key sk_η, respectively. A shortcut from v to v' is stored at $v.pp[j]$. By doing this, the reader can quickly identify tag t than the original protocol. Note that this algorithm never scarifies the degree of privacy against the compromise attack, since the shortcut information is never disclosed to adversaries unless the back-end server is compromised. The pseudo code of the proposed path pruning algorithm is given in Algorithm 4.

Note that the existing solutions, e.g., the tree-based and skip lists-based protocols, cannot benefit from the path pruning. In both of these approaches, there are k^j internal nodes at level j. The unique subset of group keys is rarely seen for $j > \eta - 1$. Therefore, one of the advantages of RSGA over the existing solutions is compatibility with the path pruning.

Algorithm 4 InitPathPruning()

```
1: For each tag t, call PathPruning(t, GK_t, v_t)
2: /* Here, v_t is the node at level η to which tag t is assigned. */
3: PathPruning(t, GK_t, v_t):
4: m ← 1
5: for each t' (t' ≠ t) do
6:    count ← 1
7:    for each j from 0 to η − 1 do
8:       if gk_j = gk'_j then
9:          count ← count + 1
10:      if m < count then
11:         m ← count
12:   if m < η − 1 then
13:      v ← the node with gk_m.
14:      v.pp[m] ← v_t.
```

Example Assume that there are two tags in the system, tags 1 and 2, and each of them has the group keys $GK_1 = \{gk_{2,1}, gk_{6,2}, gk_{4,3}\}$ and $GK_2 = \{gk_{2,1}, gk_{1,2}, gk_{5,3}\}$, respectively. Also, tag 1's secret key is assumed to be stored at v_3. While the group keys of GK_1 and GK_2 at level 1 are the same, the second group keys are different from each other, i.e., $gk_{6,2} \neq gk_{4,3}$. Thus, the subset of tag 1's group keys $\{gk_{2,1}, gk_{6,2}\}$ is unique with respect to the others. Node v_6 stores $gk_{6,2}$ at level 2, and tag 1 is mapped to v_3. Therefore, the shortcut from level 2 to level 4 can be stored by $v_6.pp[2] \leftarrow v_3$.

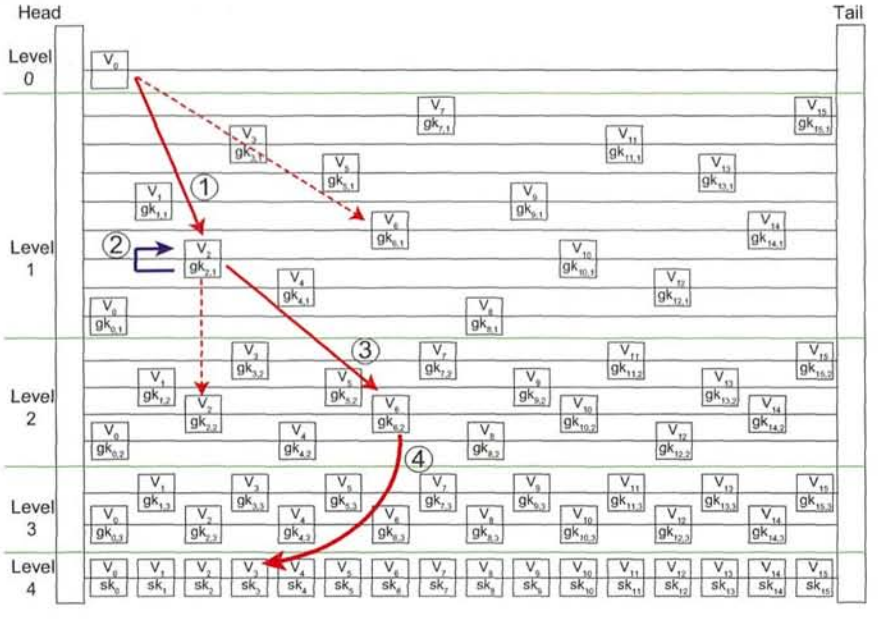


Figure 2.6: An example of the path pruning.

2.4 Analyses

In this section, security and performance analyses are provided in terms of the system anonymity, the key storage cost at the back-end sever and tags, and the number of gates required at tags.

2.4.1 Anonymity Analysis

In RSGA, two tags cannot be correlated unless they have the same group keys at all the levels in a skip graph. The correlation probability of RSGA can be deduced by Theorem 1.

Theorem 1 *Given the number of tags N and the balancing factor k, the correlation probability of RSGA is $(\frac{1}{N})^{\lceil \log_k N \rceil - 1}$.*

Proof A skip graph has $\eta = \lceil \log_k N \rceil$ levels excluding level 0 with each containing N nodes. Every tag obtains $\eta - 1$ group keys from level j $(1 \leq j \leq \eta - 1)$, and two tags will have the same group key with probability $1/N$ from level 1 to $\eta - 1$. Thus, the two tags are correlated with probability $(\frac{1}{N})^{\lceil \log_k N \rceil - 1}$. This concludes the proof.

The anonymity of the system is defined by Equation 2.4, where S_i is the anonymous set consisting of one or more tags.

$$\frac{1}{N} \sum_i |S_i| \cdot \frac{|S_i|}{N} \tag{2.4}$$

If no tag is compromised, there exists only one anonymous set and $|S_1| = N$ and the anonymity yields 1. Should some tags be compromised, the tags in the system are divided into disjoint set and each tag is anonymous within the set, and the resulted anonymity will be between 0 and 1.

The definition of anonymity shown in Equation 2.4 is based on the following observation. When the tags in the system are divided into small disjoint sets, the degree of privacy of an anonymous set, say S_i, with respect to the total number of tags is computed by $\frac{|S_i|}{N}$. There are $|S_i|$ number of tags which belong to S_i. Thus, $\frac{|S_i|}{N}$ is weighted by multiplying with $|S_i|$. After taking the summation of them for each anonymous set, the average is computed by dividing $\frac{1}{N}$.

In RSGA, an uncompromised tag belongs to the anonymous set with size k or $k-1$ only when it has the same group keys as any of the compromised nodes. Such a probability is formulated in Theorem 1. Otherwise, the tag remains anonymous within the set with size $N - N_c$, where N_c is the number of compromised tags.

2.4.2 The Comparisons with The Existing Solution

In this subsection, we demonstrate the proposed RSGA achieves the lower correlation probability than one of the existing solutions, RSLA. We first derive the correlation probability of RSLA by Theorem 2.

Theorem 2 *Given the number of tags N and the balancing factor k, the correlation probability of RSLA is $\prod_{i=1}^{\lceil \log_k N \rceil - 1} \frac{1}{k^i}$.*

Proof Given the number of tags and the balancing factor, skip lists have $\eta = \lceil \log_k N \rceil$ levels, and each level j except 0 has one list that contains k^j nodes with a group key. Hence, the probability of two tags having the exactly the same set of group keys is $\prod_{i=1}^{\lceil \log_k N \rceil - 1} \frac{1}{k^i}$. This completes the proof.

From Theorems 1 and 2, the correlation probability of RSGA is smaller than that of RSLA. This can be proven by showing $\prod_{i=1}^{\lceil \log_k N \rceil - 1} \frac{1}{k^i} - (\frac{1}{N})^{\lceil \log_k N \rceil - 1} > 0$.

Example Assume that the number of tags $N = 2^{10} = 1024$, the balancing factor $k = 2$, the height of tree $\eta = 10$. From Theorem 1, the correlation probability in RSGA is approximately 8.078×10^{-28}. On the other hand, the correlation probability of RSLA is approximately 2.842×10^{-14} by Theorem 2. Therefore, RSGA significantly improves the degree of privacy in term of the correlation probability by 10^{-14}.

2.4.3 The Storage Analysis

The key storage cost at the back-end server is obtained by Theorem 3.

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

Proof A skip graph has $\eta = \lceil \log_k N \rceil$ levels excluding level 0, and there are N nodes from level 1 to η. Note that the list at level 0 does not contain any key, and it is excluded from the consideration. Thus, the total number of nodes containing a key in the skip graph is $N \log_k N$. Therefore, the key storage cost is $O(N \log_k N)$. This completes the proof.

While RSGA requires larger storage cost than the existing solutions, which require $O(N)$ key storage cost, the back-end server has enough storage capacity. In addition, RSGA never sacrifices the authentication speed compared with the tree-based and skip lists-based approaches. Thus, we stress that RSGA provides higher a privacy preserving mechanism with reasonable key storage cost.

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 In a skip graph with η levels, one unique key, $\eta - 1$ group keys, $\eta - 1$ random shift numbers, and list pointer ptr are assigned to each tag. Since $\eta \leq log_k N + 1$, the storage cost which each tag incurs is $O(\log_k N)$. This completes the proof.

2.4.4 Weight Analysis

Based on the weight analysis presented in [35], we can estimate the number of gates required to implement our RSGA in a tag as follows. Compared with RSLA, the proposed RSGA introduces additional $\log_k N$ bits to store ptr, which indicates the entry point of a skip graph. As 1-bit memory of a D flip-flop is implemented with 5 gates, keeping $\log_k N$-bit information introduces additional $5 \log_k N$ gates. With the same condition as [35], the number of gates to implement RSGA at a tag is formulated by $3576 + 80 \times (\eta - 1) + 5 \log_k N$. For example, to maintain 2^{16} tags in a skip graph with the balancing factor $k = 2$, the number of additional gates for each tag equals to 4,856. Since 1,000 gates costs 1 cent [36], the RSGA implementation increases approximately 5 cents for each tag. However, we claim that additional 5 cents would not be a significant issue in the RFID systems, where each tag has a relatively long life-cycle, such as library RFID systems.

2.5 Performance Evaluation

For the performance evaluation by computer simulations using a self-developed simulator in Java, the proposed RSGA is implemented along with the existing solutions, including the tree-based [24], SPA [21], AnonPri [29], and RSLA [35] protocols. Note that SPA is a tree-based protocol with a key updating mechanism.

Table 2.2: The simulation parameters.

Parameters	Values
The number of tags	2^8 to 2^{14}
The balancing factor	2, 4, 8,or 16
The compromised tags	1% to 90%
Num. of pseudo ID pools (AnonPri)	1000
Num. of pseudo IDs (AnonPri)	10

2.5.1 Simulation Configuration

The way we conduct simulations is basically the same as [31, 35, 16] and similar to [14, 21, 29]. That is, the focus of our performance evaluation is on the protocol-level security, and thus, the physical functions of tags are not simulated. A simulation experiment is set up by locating one RF reader and 2^8 to 2^{14} RF tags, which is sufficiently large to accomodate a large-scale system, such as inventory management and an RFID library. The construction of a key structure and key initialization is performed by the key issuing process of each private authentication protocol. Then, to emulate the compromise attack, randomly selected N_c tags are marked as being compromised, where N_c ranges 1% to 90% or is set to be either 64, 128, 256 or 512. In the tree-based, SPA, RSLA, and RSGA, the balancing factor k is set to be either 2, 4, 8, or 16. For AnonPri, the number of pseudo ID pools and the number of pseudo IDs that each tag has are set to be 1000 and 10, respectively. In order to make our performance evaluation consistent with the related works, these protocol-specific parameters are set as the same as the performance evaluation conducted in the existing works [31, 35, 14, 21, 29]. For each system realization, 1000 experiments are performed.

We consider both static and dynamic systems. In a static system, after initializing an RFID system, a set of N_c tags are randomly chosen and marked as being compromised. Then, the system anonymity is computed. In a dynamic system, randomly selected N_c tags are marked as being compromised. Then, another set of N_c tags run the key updating algorithm to update their unique key, group keys, shift numbers, and the list pointer. This process is repeated to emulate a dynamic system. For every iteration, the system anonymity is computed just after a set of N_c tags are compromised. The simulation parameters are listed in Table 2.2.

Under those scenarios, umcompromised tags are interrogated by the reader using a private authentication protocol. To compare the performance of each protocol in various aspects, the system anonymity, authentication speed, and key storage cost are employed as metrics, each of which are computed by exactly the same way as [35]. That is, the system anonymity is computed by taking the average of the anonymity of each tag as formulated in Equation 2.4; the authentication speed is obtained by the summation of the number of light-weight cryoptographic operations including key scanning and decryption of shift numbers; the total number of group and unique keys in the system is considered as the key storage cost.

2.5.2 Simulation Results of Static Systems

Figure 2.7 shows the system anonymity with the respect to the percentage of compromised tags in the case of $k = 2$. Clearly, RSLA and RSGA, which use a random walking over a data structure, outperform the other protocols. The anonymity of RSGA is higher than RSLA by $5\% \sim 10\%$ when the percentage of compromised tags is between 20% and 60%. This is because the correlation probability of RSGA is much

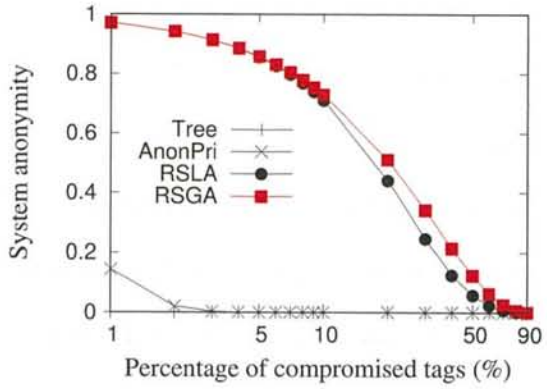


Figure 2.7: System anonymity.

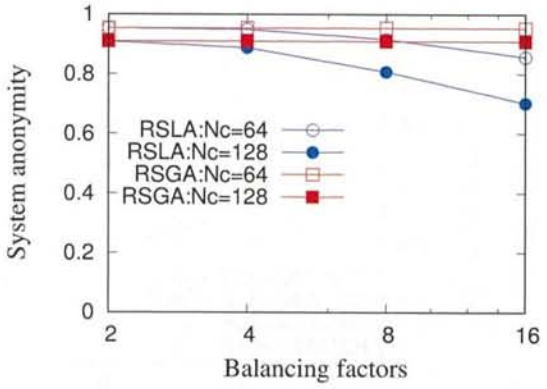


Figure 2.8: System anonymity for different k.

smaller than that of RSLA. Hence, RSGA provides the strongest privacy protection mechanism against the compromised attack. When the percentage of compromised tags is less than 20%, significant difference between RSGA and RSLA is not observed when $k = 2$. However, as we will show later, RSGA outperforms RSLA for $k \geq 4$.

Figures 2.8 and 2.9 demonstrate the system anonymity of RSLA and RSGA with respect to the balancing factor. Since tags have small memory to store keys, and thus, the number of levels of a tree/skip lists/a skip graph is limited. Hence, the balancing factor must be set to be large to support more tags. However, the larger balancing factor causes lower system anonymity as presented in [35]. In fact, as shown in 2.8 and 2.9, the system anonymity of RSLA decreases as the value of k increases. On the other hand, the proposed RSGA still preserves higher anonymity even when $k = 16$. Therefore, RSGA can scale an RFID system in keeping with higher system anonymity, which is one of the most significant advantages of RSGA over the existing solutions.

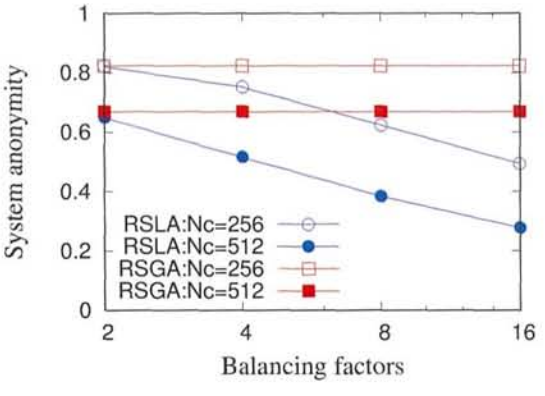


Figure 2.9: System anonymity for different k.

Figure 2.10 presents the authentication speed with respect to the number of tags. AnonPri incurs the slowest authentication speed, since it does not run in a logarithmic order. The tree-based, RSLA, and RSGA result in similar performance. This is

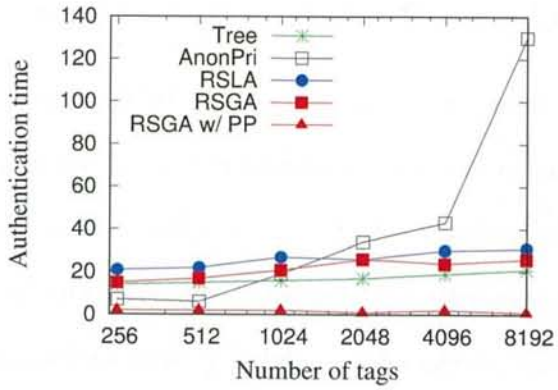


Figure 2.10: Authentication speed.

because all of them run in $O(\log_k N)$. Although RSGA w/ PP runs in a logarithmic order, the average authentication time is much smaller than $\log_k N$. In fact, as shown in the figure, RSGA w/ PP achieves the shortest authentication speed by using shortcuts. Note that the tree-based and RSLA cannot benefit the path pruning algorithm as discussed in Section 2.3.5.

Figure 2.11 illustrates the storage cost in terms of the number of unique keys and group keys in the system. RSGA incurs the most key storage cost among the existing solutions. To be specific, the key storage cost of RSGA is $O(N \log_k N)$, while that of the others is $O(N)$. However, as shown in the figure, the additional storage cost in RSGA is not that significant compared with the others. We claim that unique keys and group keys are maintained in the back-end server which has enough data storage, and therefore, the additional key storage cost never discourages the deployment of the proposed RSGA.

2.5.3 Simulation Results of Dynamic Systems

Figure 2.12 depicts the anonymity of the RFID system in a dynamic scenario for different protocols with respect to the percentage of compromised tags. In this setting, the balancing factor is set to be $k = 2$. Note that AnonPri is excluded because it does not provide a key updating mechanism. Compared with SPA (a tree-based protocol), RSGA and RSLA provides much higher system anonymity. This is because RSGA and RSLA randomized key structure. Unlike to the one in a static environment shown in Figure 2.7, the difference between RSGA and RSLA is not significant when $k = 2$. However, again we claim that RSGA results in much higher system anonymity for $k \geq 4$ in a dynamic environment.

Figures 2.13 and 2.14 illuminate the system anonymity in a dynamic scenario for different protocols with respect to the balancing factor. Similar to the static scenario, the system anonymity of RSLA decreases in proportion to the balancing factor. On the other hand, the proposed RSGA maintains high system anonymity for large value of k. Particularly, when $k = 16$ and $N_c = 512$ in Figure 2.9, RSGA presents significant improvement compared with RSLA. Therefore, our RSGA can scale up the RFID system without sacrificing the degree of privacy.

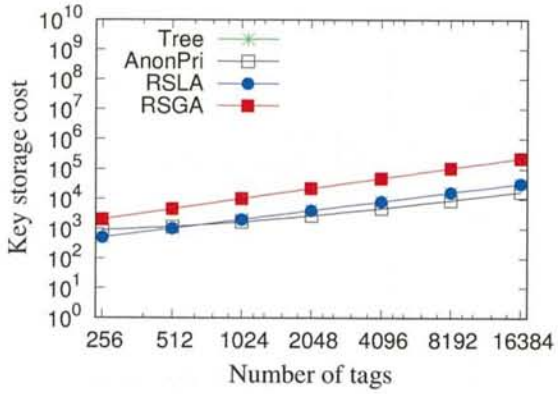


Figure 2.11: Storage cost.

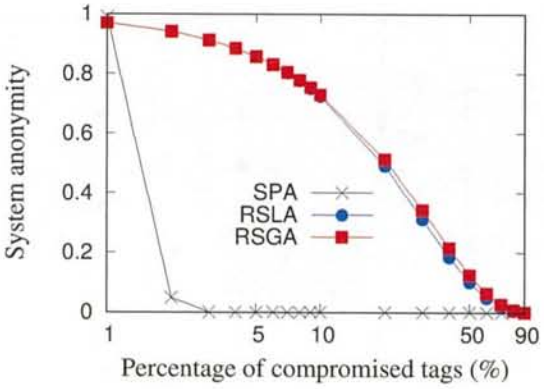


Figure 2.12: System anonymity.

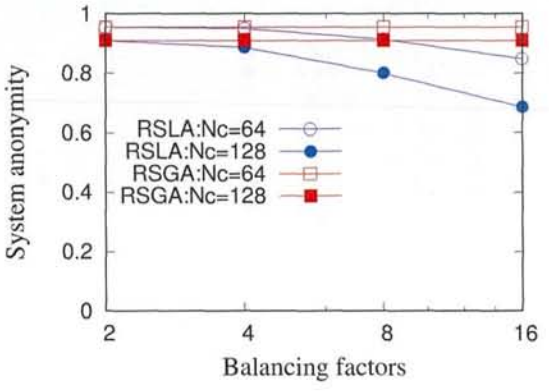


Figure 2.13: System anonymity for different k.

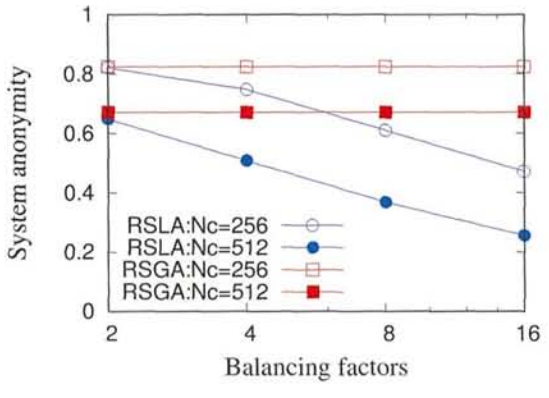


Figure 2.14: System anonymity for different k.

Chapter 3: Grouping Protocols

3.1 Related Works

RFID security and privacy are widely studied for many different RFID-enabled applications. The tag's privacy in terms of indistinguishably and unpredictability is formally defined in [18]. The private tag authentication protocols [24] protect tags' replies from eavesdropping over several interrogation. In order to achieve fast and secure singulation, advanced data structures are employed for key managements, such as skip lists-based [35] and hash index-based [3]. Some researches rely on physical layer supports for secure tag authentication. In [32], Sakai et al. design a novel coding scheme for the secure tag singulation under the privacy masking [30], where some portion of a tag's reply is intentionally corrupted by jamming. The ghost-and-leech attacks, which is the man-in-the-middle like attack, is addressed by motion signature in [8]. To avoid unexpected tag accesses, Saxena et al. [34] propose a locking/unlocking mechanism, in which a tag must unlocked by the smartphone of the tag's owner before authentication.

3.2 Preliminary

3.2.1 The Tag Grouping Problem

An RFID system consists of one reader and n tags. The set of tags in the system is denoted by $\mathcal{T} = \{t_1, t_2, ..., t_n\}$, and they are divided into m groups, denoted by $\mathcal{G} = \{G_1, G_2, ..., G_m\}$. The set of groups are disjoint, and thus, we have $\sum_{i=1}^{m} |G_i| = n$. To uniquely identify each tag and group, we define ID_i as the tag t_i's ID and GID_i as the group P_i's ID. The reader is assumed to know all tag IDs as well as the group ID corresponding to each tag.

The tag grouping problem is defined by the efficient labeling of all the tags in $\mathcal{T}$ according to $\mathcal{G}$. Note that the prefix of a tag's ID serves on representing a static ID. However, these part cannot be changed once tags are manufactured. In the tag grouping problem, some portions of a tag's memory is devoted to store a group ID by user's preference.

3.2.2 Grouping Protocols

A naive solution for grouping tags is the traditional polling grouping (TPG) protocol, which is desinged by extending a polling protocol in [28]. In TPG, each tag has three state, *unlabeled*, *marked*, and *labeled*. At first, a tag is in *unlabeled* state. In the polling phase, it switches its state to *marked*. Then, in the labeling phase, the reader can write a group ID to the tag with *marked* state. After labeling, the tag's state goes to *labeled*. This straightforward approach, unfortunately, takes a long time. To alleviate this issue, Liu et al. propose a set of grouping protocols, e.g., the enhanced polling grouping (EPG) and filtering grouping (FIG) protocols in [19]. In EPG, instead of writing a group ID to individual tags with *marked* state, the reader

broadcasts a group ID towards several tags for simultaneous labeling. By doing this, the transmission costs of group IDs in the labeling phase can be minimized. In FIG, the Bloom filter is used so that a set of tags change its state from *unlabeled* to *marked* on one query. With this approach, the transmission costs in both the polling and labeling phases are reduced.

3.2.3 Bloom Filters

A Bloom filter [2] is a probabilistic data structure to identify whether or not an element is a member of a set. In a Bloom filer, there might be a false positive, but false negatives never occur. In other words, on receiving a query, a tag concludes that it is possibly in the set or definitely not a member. A query consists of ℓ-bit and K different hash functions. At the beginning, all the bits in a filter are set to be 0. To add an element (i.e., a tag's ID in this paper) to the filter, K array positions are computed by applying K hash functions, and the corresponding K bits in the filter is set to be 1. To check whether a tag is included in the filter, K array positions are computed. The tag is possibly included in the filter if all the positions corresponding to the array equal to 1. Otherwise, the tag is definitely not a member.

3.2.4 Cuckoo Filter

A Cuckoo Filter [11] improve on Bloom filters and supports deletion of elements. To insert an element into the cuckoo filter, we get two indexes from the hash and its fingerprint-based elements. After that, as soon as these indexes are obtained, it inserts the fingerprint of the element into one of the two possible buckets corresponding to the obtained index. As the cuckoo hash table begins to fill up, we encounter the situation that there are no more free 2 indexable indexes into which the element

should be inserted. In this case, the elements in the cuckoo hash table at this point are exchanged with other indexes to release the space needed to insert the new element. By implementing inserts in this way you can examine the fingerprints in one of the two indexable indexes and easily remove the elements from the table if that fingerprint is present.

3.3 The Privacy Model

In this paper, we address the private tag grouping at the protocol-level, where for a given tag an adversary cannot identify the tag's group. In our privacy model, an adversary is able to eavesdrop all the observed information on the communication channel between the reader and tags. Note that the internal state of tags, i.e., the secret key, is assumed to be never compromised, since our design aims at achieving the protocol-level privacy. In addition, we assumed that the adversary cannot distinguish tags based on the physical layer characteristics, e.g., an adversary cannot know which tag replies to the reader from the signal strength.

3.3.1 Privacy Model

In our privacy experiment, there are one reader R and a set of tags $\mathcal{T}$. Tags are divided into disjoint groups, denoted by $\mathcal{G}$. For convenience, we define $g : ID \rightarrow GID$ as a mapping function from a tag ID to the corresponding group ID, and $g(t_i)$ denotes the t_i's group. The parameters of a grouping protocol include the number of groups m, the mean of a group size μ, and the standard variance σ. We define $\Pi := (m, \mu, \sigma)$. A set of data (e.g., nonce, hashed IDs, and group IDs) transmitted over the wireless channel during a grouping protocol is denoted by C_b.

We assume $|G_i| \geq 2$ for an indistinguishably-based privacy experiment. According to the EPC global standard, every tag shall be equipped with a 16-bit pseudorandom generator. Thus, tags are assumed to be able to execute a pseudorandom function family (PRF).

We define the following random oracles, in which the inputs are not tractable from the outputs.

- $InitSys(R, \mathcal{T}, \mathcal{G}, \Pi)$ sets up an RFID systems with given parameters.
- $Select(\mathcal{T}, \mathcal{G})$ randomly selects one group G and returns randomly selected two tags, t_0 and t_1, in G.
- $Grouping(t_0, t_1)$ selects one of t_0 and t_1 by the uniform distribution, i.e., let $b \leftarrow_{uniform} \{0, 1\}$ and t_b be the selected tag. Then, the oracle runs a secure grouping protocol to t_b and returns C_b. Here, C_b is a set of values which can be observed in a grouping protocol between R and t_b, e.g., nonce, hashed IDs, and so on.
- $Query(\mathcal{T}')$ returns $g(t_i)$ for given $t_i \in \mathcal{T} - t_0, t_1$. Note that $|\mathcal{T}'|$ is polynomial, i.e., an adversary can queries tags to the oracle polynomial number of times.

Each oracle is denoted by $\mathcal{O}_{InitSys}$, $\mathcal{O}_{Grouping}$, and $\mathcal{O}_{Query}$, respectively.

An experiment is denoted by $\mathrm{Exp}_{\mathcal{A},\Pi}(R, \mathcal{T}, \mathcal{G})$. Adversary $\mathcal{A}$ tries to succeeds the following experiment.

Experiment 1 $\mathrm{Exp}_{\mathcal{A},\Pi}(R, \mathcal{T}, \mathcal{G})$:

1. An RFID system is initialized by $\mathcal{O}_{InitSys}$.

2. Adversary $\mathcal{A}$ is given two tags, t_0 and t_1, both of which belong to the same group, from the oracle $\mathcal{O}_{Select}$.

3. Adversary $\mathcal{A}$ sends the two tags to $Grouping(.)$ and receives C_b.

4. Adversary $\mathcal{A}$ learns C_i by querying randomly selected tag, t_i, polynomial number of times.

5. Adversary $\mathcal{A}$ guesses b and determines $b' \leftarrow \{0,1\}$.

6. If $b = b'$, the experiment outputs 1 (the experiment succeeds) and 0 otherwise.

With the aforementioned privacy model, the security of a tag groping protocol is formally defined as follows.

Definition 1 (A Private Grouping Protocol) *A tag grouping protocol is said to be private against polynomial adversaries at the protocol-level if Equation 3.1 holds.*

$$\Pr\left[\mathrm{Exp}_{\mathcal{A},\Pi}(R,\mathcal{T},\mathcal{G}) = 1\right] \leq \frac{1}{2} + negl(n). \tag{3.1}$$

Here, $n = |\mathcal{T}|$ and $negl(n)$ is negligible.

3.3.2 The Limitation of Our Model

The proposed privacy model has limitation in its experiment; the two challenge tags are selected by the oracle and these tags belong to the same group. If adversary $\mathcal{A}$ is allowed to select two tags of her choice, her advantage of the experiment will not be negligible. For example, $\mathcal{A}$ randomly selects t_0 and t_1. Let G_i be $g(t_0)$ and G_j be $g(t_1)$. Then, $G_i \neq G_j$ and $|G_i| \neq |G_j|$ most likely hold. In this case, the

$\mathcal{A}$'s advantage is abs $\left(\frac{|G_i|}{n} - \frac{|G_j|}{n}\right) + negl(n)$, which is not negligible. Therefore the challenge tags are selected by the oracle in our privacy model.

In addition, our model does not guarantee the positive and negative membership privacy. That is, given tag t, an adversary can tell whether or not t belong to G_i with a non-negligible probability. Since the size of each group may differ each other, an adversary can tell if t is in G_i or not with non-negligible probability.

3.4 Private Grouping Protocols

3.4.1 Motivations and Basic Idea

Fast grouping of RF tags achieves quick distribution of group IDs for further efficient object monitoring and classifications. However, none of the existing grouping protocols does not consider the privacy of tags, where adversaries can identify the corresponding group of each tag belonging to. Therefore, in this section, we propose four private grouping protocols, namely PrivTPG, PrivEPG, PrivBFG, and PrivCFG.

All of them consists of three phases. First, the initialization phase is to exchange nonce; second, the polling phase changes tags' state from *unmarked* to *marked*, which indicates the tag is ready to be categorized; third, the labeling phase assign a group ID to individual tags and these tags' state switch to *labeled*. Each proposed protocol differs in their polling phase. The basic idea to preserve tags' privacy is the use of a hash function in the polling phase.

3.4.2 System Parameters

We assume that an large-scale RFID system consists of n tags and one reader, which is securely connected to the back-end server. We denote the tag set as $\mathcal{T} = \{t_1, t_2, ..., t_n\}$. A set of groups is defined by $\mathcal{G} = \{G_1, G_2, ..., G_m\}$, where m is the

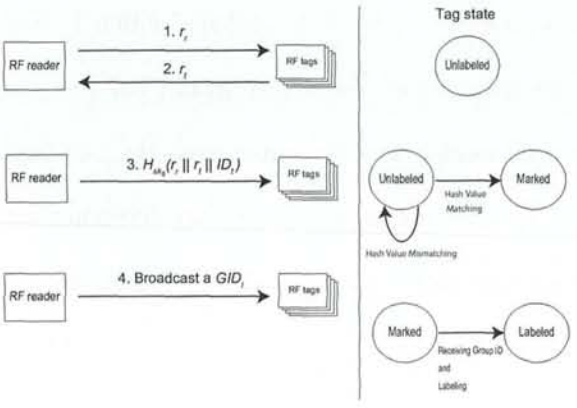


Figure 3.1: The state diagram of PrivTPG.

number of groups. Hence, the total number of tags is the sum of the group sizes, i.e., $\sum_{i=1}^{m} |G_i| = n$. Each group G_i has a unique group ID GID_i $(1 \leq i \leq m)$. On the system deployment, only an RF reader knows the tag partition $\mathcal{G}$. Hence, an RF reader needs to label individual tags based on $\mathcal{G}$.

3.4.3 The Private TPG Protocol

We first design a TPG-based baseline protocol, namely private tag polling grouping (PrivTPG), which state diagram is shown in Figure 3.1. First, the reader and individual tags exchange nonce, r_r and r_t, each of which is generated by the reader and a tag. In the polling phase, the reader sends a hashed ID of tag t, i.e., $H_{sk_k}(r_r||r_t||ID_t)$. Then, t checks if the hashed value corresponding to its ID, and if so, t changes its state to *marked*. In the labeling phase, the reader sends GID_i (where $g(t) = GID_i$) to t and the tag goes to the *labeled* state. This process is continue until the reader labels n tags.

Hence, the execution time of Privacy TPG is $\{\frac{t_{id}}{96} \times (|2r| + |H|) + t_{gid} + t_{in} + t_h\} \times n$, where t_{id} is the length of a time slot that transmits a 96-bit tag ID [9], t_{gid} is the length of a slot for transmitting a group ID, r is the length of a nonce, H is the length of a hash value, t_{in} is the time interval, t_h is the hash calculation time, and n is the number of tags.

3.4.4 The Private EPG Protocol

The performance of PrivTPG can be improved by reducing unnecessary broadcasting in the polling phase. To this end, we propose private enhanced polling (PrivEPG), which state diagram is illustrated in Figure 3.2. PrivEPG differs from PrivTPG in the polling phase. After exchanging nonce, r_r and r_t, in the initialization phase, the protocol enters the polling phase. For each group G_i $(1 \leq i \leq m)$, the reader does followings. The reader computes $H_{sk_k}(r_r||r_{t_j}||ID_{t_j})$ for tag $t_j \in G_i$ and sends it to the tag. Tag t_j switches its state to *marked*, if the hashed query corresponds to the one which is computed by itself. Before sending GID_i, the reader changes the state of all the tags in G_i. Then, GID_i is broadcasted by the reader. In this approach, single transmission of a group ID to label all the tags in the same group is conducted for each group in the labeling phase. Thus, the transmission cost can be reduced.

The Privacy EPG sends each group ID by broadcasting, the execution time of Privacy EPG is $\{\frac{t_{id}}{96} \times (2|r| + |H|) + t_{in} + t_h\} \times n + (t_{gid} + t_{in}) \times m$, where m is the number of groups. Hence, the Privacy EPG can improve the grouping efficiency over the Privacy TPG.

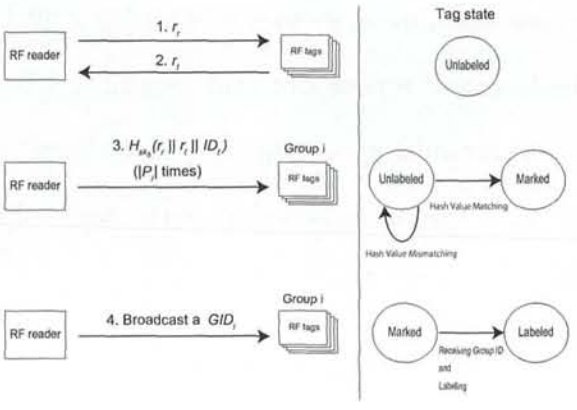


Figure 3.2: The state diagram of PrivEPG.

3.4.5 The Private BFG Protocol

We further propose a private tag grouping protocol with one of the advanced data structures, namely privae Bloom filter-based grouping (Priv.BFG). The state diagram of Priv.BFG is presented in Figure 3.3. In the initialization phase, the reader and tags exchange nonce, r_r and r_t.

In the polling phase, the reader simultaneously changes the state of tags in the same group by applying a Bloom filter. In this phase, for each group, say G_i the reader executes the followings. First, a Bloom filter with length ℓ and K hash functions is initialized. Let H^k be the k-the hash function ($1 \leq k \leq K$). When a tag, say t, is added to the filter, a pseudo ID is used as an entity instead of ID_{t_j}. Let PID_{t_j} be the pseudo ID of tag t, which defined as $PID_t = H_{sk_t}(r_r||r_t||ID_t)$. Then, the reader obtains K array positions of PID_t in the filter by applying K hash functions and sets the corresponding bits to be 1. This process is repeated for each tag in G_i, and the reader broadcasts a query with a Bloom filter to tags. On receiving a query, each tag,

say t', does the followings. First, t' compute $PID_{t'}$ from $H_{sk_{t'}}(r_r||r_t||ID_{t'})$. Then, computes K array positions form K hash functions obtained from the Bloom filter. If all the K array positions in the Bloom filter are 1, then t' concludes it is in the member and switches its state to *marked*. Otherwise, t' ignores the filter.

In a Boom filter, false positive occurs, i.e., tags which are not in G_i might changes their state to *marked*. This can be corrected as follows. The reader knows a set of marked tag, denoted by M_i, and a set of tags in G_i. Thus, the tags in $M_i - G_i$ should changes their state to *unmarked*. To this end, reader computes $PID_t = H_{sk_t}(r_r||r_t||ID_t)$ for each tag t in $M_i - G_i$ and broadcasts the set of PIDs. On receiving query, each tag again computes its PID and goes to the *unmarked* state if it is included in $M_i - G_i$.

In the labeling phase, the reader broadcasts the group ID, GID_i, to the tags in G_i in order to change their state to *labeled*. This process is repeated until the reader labels all the tags.

Filtering Phase

In this phase, an RF reader generates and broadcasts a bloom filter to quickly mark each tag in group P_i. Let L_i be the length of the bloom filter in round i, and k_i be the number of hash functions in round i. An RF reader generates the bloom filter taking the false positive into account. The flase positive rate f_i in round i is defined the following Formula 3.2, where m_i is the number of tags in group P_i.

$$f_i = (1 - (1 - \frac{1}{L_i})^{k_i m_i})^{k_i} \approx (1 - \exp \frac{-k_i m_i}{L_i})^{k_i} \tag{3.2}$$

The minimal value of f_i is $0.6185^{\frac{L_i}{m_i}}$ where $k_i = \ln 2 \times \frac{L_i}{m_i}$. Hence, the optimal L_i and k_i are the following Formula 3.4 and 3.3, where n_i is the number of unlabeled tags

at that point. If L_i is too long, an RF reader can split it into 96-bit segments, and transmit each segment in t_{id}.

$$L_i = \frac{m_i}{(\ln 2)^2} \times \ln(96 \times (\ln 2)^2 \times \frac{n_i - m_i}{m_i}) \tag{3.3}$$

$$k_i = \ln 2 \times \frac{L_i}{m_i} \tag{3.4}$$

Each unlabeled tag hashes own tag ID to k_i bit positions in the bloom filter by using k_i hash functions. If the all k_i bits in the bloom filter are 1, the tag passes the filter and transitions to the marked state from the unlabeled state.

Polling Phase

In this phase, an RF reader broadcasts the false positive tag ID. Since an RF reader knows the set of marked tag M_i and set of tags P_i, can predict the tags that should be unmarked from subset $M_i - P_i$. Hence, an RF reader broadcasts the tags ID in the subset $M_i - P_i$. Each tag that received own tag ID in this phase transitions to the unlabeled state from the marked state.

Labeling Phase

In this final pahse, an RF reader broadcasts the group ID of P_i, and each marked tag transitions to the labeled state from the marked state.

Execution Time

The minimal execution time $T(m_i, n_i)$ in round i is the following Formula 3.5.

$$\begin{aligned} T(m_i, n_i) = k_i \times t_h \times m_i + \frac{(|2r| + L_i)}{96} \times (t_{id} + t_{in}) \\ + (n_i - m_i) \times 0.6185^{\frac{L_i}{m_i}} \times (t_{id} + t_{in}) + t_{gid} \end{aligned} \tag{3.5}$$

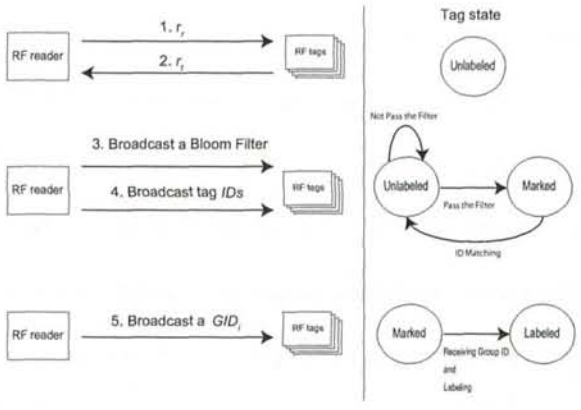


Figure 3.3: The state diagram of PrivBFG.

The first member is the time of calculating hash, and the second member is none exchange and filtering phase that broadcasts the L_i length bloom filter. The third member is polling phase that broadcasts $|M_i - P_i|$ tag IDs, where $0.6185^{\frac{L_i}{m_i}}$ is the minimal false positive rate. The third member is labeling phase that broadcasts the group ID of P_i.

3.4.6 The Private CFG Protocol

Finally, we propose a private tag grouping protocol applying the Cuckoo filter, namely private Cuckoo filter-based grouping(Priv.CFG). The state diagram of Priv.CFG is presented in Figure 3.4. In the initialization phase, the reader and tags exchange nonce, r_r and r_t.

In the polling phase, the reader simultaneously changes the state of tags by applying a Cuckoo filter. First, a L length cuckoo filter consisting of 2 hash functions and 4 buckets is initialized, which is denoted as a $(2, 4)$-cuckoo filter (i.e., each item has two candidate buckets, each bucket has up to four fingerprints and the number of buckets

is L) is initialized. Each bucket contains the $x's fingerprint$ (f bit) and a encrypted group ID ($GID_i \oplus r_t \oplus r_r$). After the initialization phase, the RF reader calculates the storage destination candidate i and j ($0 \leq i, j < L$) of the element x by using the Equation (3.6) and the Equation (3.7). The RF reader randomly selects either i or j buckets with free space, and then stores a $x's fingerprint$ and a encrypted group ID. If bucket i and j have no available space, kick out a previously stored element and store it there. The RF reader calculates a relocation destination of the kicked out element using by Equation (3.8) and the process repeats again.

$$h_1(x) = hash(x||r_t||r_r) \tag{3.6}$$

$$h_2(x) = h_1(x) \oplus hash(x\text{'s } \mathbf{fingerprint}) \tag{3.7}$$

$$j = i \oplus hash(x\text{'s } \mathbf{fingerprint}) \tag{3.8}$$

It is possible to encounter an infinite loop of relocating the elements. Therefore, we set an upper limit on the loop of relocation, and we decided to retransmit the element that could not be placed by the second small cackoo filter.

After receiving a cuckoo filter, each tag calculates i and j by by using the Equation (3.6) and the Equation (3.7). If tag's fingerprint matches the element in the bucket, the tag decrypt the group ID and change own state to labeled.

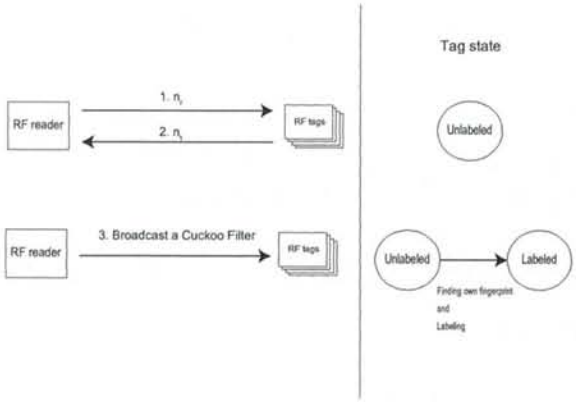


Figure 3.4: The state diagram of PrivCFG.

Execution Time

If the number of entries in the bucket is 4, the execution time of PrivCFG is $t_h \times 2 \times n + \frac{t_{id}+t_{in}}{96} \times \{2|r| + 4L(f + |r|)\}$, where f is length of fingerprint, L is the number of bucket.

False Positive

The false positive rate ϵ is given by the following Equation (3.9) from the probability that the fingerprints match and the probability of using the same two bucket.

$$\epsilon = 1 - \left(1 - \frac{1}{2^f}\right)^{2b} \approx \frac{2b}{2^f} \tag{3.9}$$

Optimal Length of Fingerprint

Consider the construction process of inserting n random items into an empty table with bucket size b and the number of buckets $m = cn$, where c is a constant. Insertion failures occur whenever $q = 2b + 1$ items are mapped to the same two

buckets. Therefore, the expected number of groups of $2b+1$ items colliding during the inserting process is

$$\binom{n}{2b+1} = \binom{n}{2b+1}\left(\frac{2}{2^f \cdot cn}\right)^{2b} = \Omega\left(\frac{n}{4^{bf}}\right) \tag{3.10}$$

In this case, in order to set $\Omega(\frac{n}{4^{bf}})$ to $\Omega(1)$, 4^{bf} must be n. Hence, we set the fingerprint size as $f = \Omega(\frac{\log_2 n}{b})$ bits.

Optimal Number of Buckets

The average number of bits per item C representing the space efficiency is shown by the following Equation (3.11), where α is the load factor.

$$C = \frac{\textbf{table size}}{n} = \frac{f \cdot (\textbf{\# of entries})}{\alpha \cdot (\textbf{\# of entries})} = \frac{f}{\alpha} \textbf{ bits.} \tag{3.11}$$

The table size is expressed by the product of the bucket size b, the number of buckets L and the number of bits of the fingerprint f. Hence, the number of buckets is

$$L = \frac{n}{\alpha \cdot b} \tag{3.12}$$

Optimal Bucket Size

Figure 3.5 demonstrates amortized space cost per item vs. measured false positive rate with different bucket size. As the false positive rate increases, the difference due to the bucket size decreases. Since the bucket size 4 is the most space efficient, we choose $(2, 4)$-cuckoo filter.

3.5 Analyses

The proposed PrivTPG, PrivEPG, and PrivBFG are private in our privacy model defined in Experiment 3.3.1. Due to the space constraint, we only prove the privacy of PrivBFG as follow.

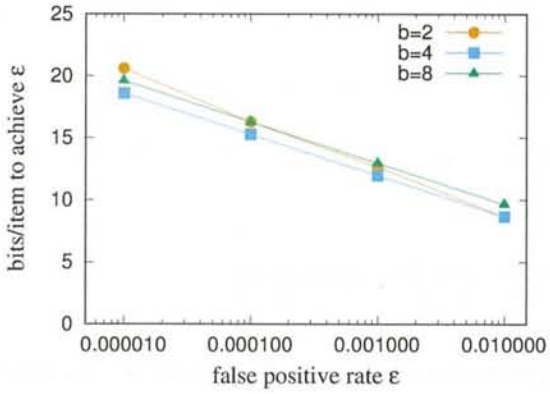


Figure 3.5: Amortized space cost per item vs. measured false positive rate

Theorem 5 *PrivBFG is private against adversaries in Experiment 1.*

Proof: To break the privacy of tags, adversary $\mathcal{A}$ must be able to tell C_b is generated by the interrogation of t_0 or t_1, where the set of observed values is $C_b = \{r_r, r_t, f, g(t_b)\}$ in PrivBFG.

Thank to the one-way property of a collision resistant hash function, $\mathcal{A}$ can deduce neither t_b from f without the valid secret key of t_b. Thus, $\mathcal{A}$ is unable to tell C_b is generated by the interrogation of t_0 or t_1. While $\mathcal{A}$ can query tags except t_0 and t_1 polynomial number of times, the $\mathcal{A}$'s advantage is negligible, which can be proven by the reduction to distinguishing a truly random value and one generated by a PRF. Let $q(n)$ be the polynomial number of queries that $\mathcal{A}$ asks to the oracle $\mathcal{O}_{Query}$. Given a queried tag t', the oracle generates random strings, r'_r and r'_t. Then, it computes a filter containing the hashed value by $H(r'_r||r'_t||ID_{t'}||sk_{t'})$, which is obtained by a random function family. Finally, r'_r, r'_t, f, and $g(t')$. For $\mathcal{A}$ to link t_b and t', the hashed IDs with nonce generated by t_b and t' must be the same. Let *Coll* be the

event that the collision occurs and $\Pr[Coll] \leq negl(n)$. Thus, the probability that $\mathcal{A}$ succeeds the experiment is bounded by $q(n) \times \Pr[Coll] \leq negl(n)$. Therefore, the above claim is true. This concludes the proof. ∎

3.6 Performance Evaluation

For the performance evaluation by computer simulations, three private RFID grouping protocols, PrivTPG, PrivEPG, and PrivBFG are implemented.

3.6.1 Simulation Configuration

In our simulation experiments, an RFID system consists of one reader and a number of tags. The tags are divided into m groups and the number of tags in one group is determined by the normal distribution $N(\mu, \sigma)$, where μ is the mean and σ is the standard variance. Thus, the total number of tags in the RFID system is computed by $n = m \times \mu$. For each system parameters, we set $2^1 \leq m \leq 2^{10}$, $10 \leq \mu \leq 100$, and $100 \leq \sigma \leq 800$, respectively.

As the specification of the EPC global Gen-2 standard [9], the tag's ID is set to be 96-bit, and hence, the transmission of one tag's ID takes $t_{id} = (37.45 \times 96 + 302) = 3897.2\mu s$. Note that one bit transmission costs $37.45\mu s$ with the time interval $302\mu s$. The length of a group ID is defined by $\lceil \log_2 m \rceil$, and thus, the transmission of one group ID costs $t_{gid} = (37.45 \times \lceil \log_2 m \rceil + 302)\mu s$. For each configuration, 1000 independent simulations are performed.

Table 3.1: The simulation parameters.

Parameters	Vlaues
The number of groups	$m = 2^1$ to 2^{10}
The mean of each group size	$\mu = 10$ to 100
The standard variance	$\sigma = 100$ to 800
The number of tags	$n = m \times \mu$
The length of tag ID	96-bits
The length of group ID	$\lceil \log_2 m \rceil$
The transmission time of one bit	$37.45\mu s$
The delay in computing a hash	$28077.92\mu s$
The time interval	$302\mu s$
The load factor in cuckoo filter	95%

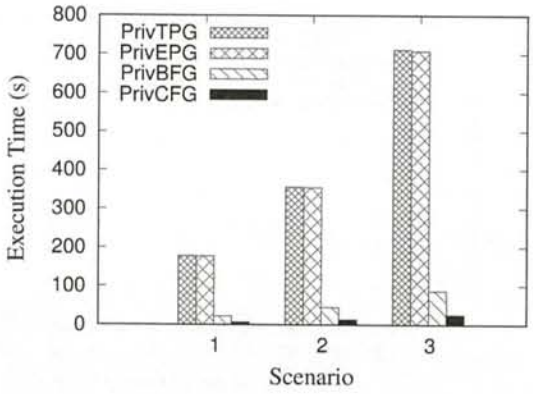


Figure 3.6: The execution time of different protocols.

3.6.2 Simulation Results

Figure 3.6 shows the execution time of different protocols under three set of system parameters. For each scenario, a set of system parameters (m, μ, σ) is set to be Scenario 1 (50, 100, 40), Scenario 2 (100, 100, 100), and Scenario 3 (100, 200, 100),

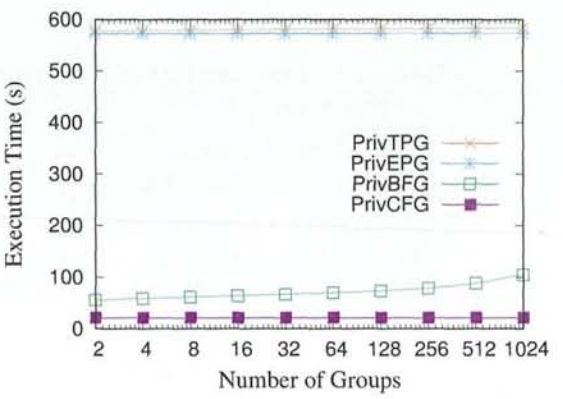


Figure 3.7: The execution time for different number of groups.

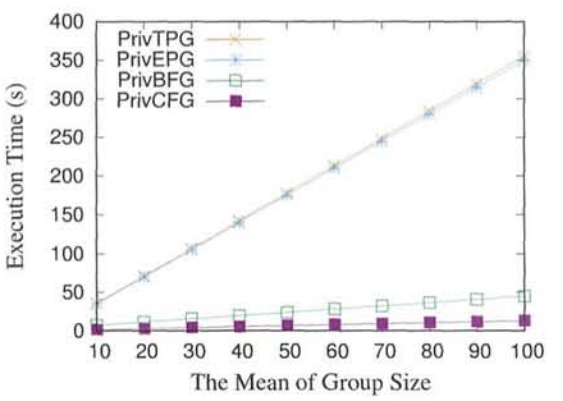


Figure 3.8: The execution time for different means of the group size.

respectively. Since the number of tags is determined by $N = k \times \mu$, the number of tags in each scenario is set to be either 5000, 10000, or 20000. As can be seen from the figure, the execution time increases as the number of tags in the system increases. For all the scenarios, PrivCFG achieves the faster grouping than the others, and PrivEPG

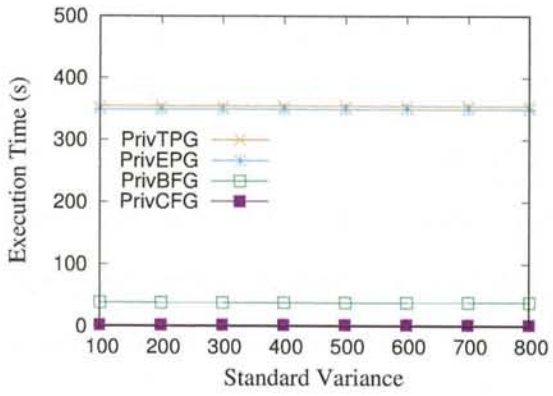


Figure 3.9: The execution time for different standard variance of the group size.

result in slightly faster execution time than PrivTPG does. Moreover, PrivBFG and PrivCFG significantly reduce execution time than other methods. Figure 3.6 implies that the execution time is influenced by various parameters. In the subsequent discussion, we will see how m, μ, and σ affect the performance.

Figure 3.7 presents the execution time with respect to the number of groups m. Here, we set $n = 2^{14}$, $\sigma = 0$, and $\mu = \frac{n}{m}$, respectively. Clearly, the execution time of the privacy PrivTPG, PrivEPG and PrivCFG are independent of the number of groups, since the number of times to send a group ID is relatively small compared with broadcasting tag IDs and their hash values. On the other hand, the execution time of PrivBFG slightly increases as the number of groups increases. This is because the retransmissions of group IDs occur due to the false positive filtering of the Bloom filter in PrivBFG.

Figure 3.8 demonstrates the execution time with respect to the mean of the group size μ. In this setting, the number of groups and the standard variance are set to

be a constant, i.e., $k = 100$ and $sigma = 0$. Then, the mean of group size μ ranges from 0 to 100. Note that the number of tags in each groa prup is of the same size, as we set $\sigma = 0$. In addition, the total number of tags N in the system increases when μ increases. Hence, the execution time increases in proportion to the value of μ. PrivBFG and PrivCFG present reduces the execution time by nearly half of those of PrivTPG and PrivEPG. Again, PrivEPG yields slightly faster execution than PrivTPG.

Figure 3.9 illustrates the execution time with respect to the standard variance σ, where we set $m = 10$ and $\mu = 1000$. The value of σ changes from 100 to 800. No matter what the value of σ is, the total number of tags in the system remains the same. Therefore, as the figure indicates, the execution time is independent from the value of σ. Again, the performance of PrivCFG results in the best.

Chapter 4: Conclusion

Security/Privacy and scalability issues are concerns when we deploy RFID systems into realistic systems such as IoT.Therefore, in this dissertation, we address the problem of private tag authentication and grouping. The summary of this dissertation is as follows:

First, we seek to preserve higher anonymity of a large-scale RFID system under various attacks, including eavesdropping and the compromise attack. To this end, we propose Randomized Skip Graph-Based Authentication (RSGA) where one of the advanced data structures, called a skip graph, is employed to maintain unique and group keys. The key idea of RSGA is the randomization at each level and dependency among different levels of the skip graph. In the proposed scheme, the replies from two different tags are never distinguishable unless they have exactly the same group keys at all the levels, and the analysis shows that the correlation probability of our RSGA is much smaller than any of the existing protocols. The proposed RSGA is augmented by a key updating algorithm to adopt to a dynamic environment. In addition, we propose a path pruning algorithm to further facilitate the singulation process by taking a shortcut from an internal node to the corresponding node of a tag in a skip graph. By analysis, we derive the correlation probability of RSGA as well as that of existing solutions. Furthermore, the key storage cost in terms of the number of

keys stored at the back-end server as well as at individual tags is analyzed. Moreover, the number of gates required by tags are quantified. The performance evaluation by computer simulations demonstrates that our RSGA achieves the highest anonymity among the existing tree-based, group-based, and skip lists-based protocols in keeping with reasonable storage cost.

Second, we address the private tag grouping problem. First, we propose an indistinguishably-based privacy model. Then, a set of private grouping protocols, PrivTPG, PrivEPG, and PrivBFG, based on the existing solutions [28, 19], and PrivCFG based on the cuckoo filter. The proposed protocol are proven private by a provable security analysis using random oracles. On the performance side, the simulations are conducted to show that our PrivTPG, PrivEPG, PrivBFG, and PrivCFG can complete grouping tags within reasonable amount of time. In the future, we will further improve private tag grouping protocols as follows. First, as discussed in Section 3.3.2, our privacy model has some limitations. Thus, we will improve the privacy model. Second, while adversaries cannot link tags and their groups, the group IDs themselves are disclosed in the proposed protocols. Therefore, we will design an efficient grouping protocol that simultaneously labels tags in the same group in keeping with the group IDs in secret.

Our work is important both in theory and practical. Our results can be applied to existing RFID applications to increase the degree of security, reliability, and scalability without sacrificing system performance. We believe that the proposed protocol and architecture will be the foundation of the next generation RFID systems.

Bibliography

[1] James Aspnes and Gauri Shah. Skip Graphs. *ACM Trans. Algorithms*, 3(37), 2007.

[2] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Comms. of the ACM*, 13(7):422–426, 1970.

[3] M. Chen and S. Chen. ETAP: Enable Lightweight Anonymous RFID Authentication with O(1) Overhead. In *ICNP*, pages 267–278, 2015.

[4] Min Chen, Shigang Chen, and Yuguang Fang. Lightweight Anonymous Authentication Protocols for RFID Systems. *IEEE/ACM Trans. Netw.*

[5] Min Chen, Jia Liu, Shigang Chen, Yan Qiao, and Yuanqing Zheng. DBF: A General Framework for Anomaly Detection in RFID Systems. In *INFOCOM*, 2017.

[6] Wonjoon Choi, Myungchul Yoon, and Byeong-Hee Roh. Backward Channel Protection Based on Randomized Tree-Walking Algorithm and Its Analysis for Securing RFID Tag Information and Privacy. *IEICE Transactions*, 91-B(1):172–182, 2008.

[7] Theofilos Chrysikos and Stavros Kotsopoulos. Characterization of Large-Scale Fading for the 2.4 GHz Channel in Obstacle-Dense Indoor Propagation Topologies. In *IEEE VTC-Fall*, 2012.

[8] Alexei Czeskis, Karl Koscher, Joshua R. Smith, and Tadayoshi Kohno. RFIDs and Secret Handshakes: Defending against Ghost-and-Leech Attacks and Unauthorized Reads with Context-Aware Communications. In *CCS*, pages 479–490, 2008.

[9] EPCglobal. EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz-960MHz version 1.0.9.

[10] EPCglobal. EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz-960MHz version 2.0.0., 2013.

[11] Bin Fan, Dave G. Andersen, Michael Kaminsky, and Michael D. Mitzenmacher. Cuckoo filter: Practically better than bloom. In *CoNEXT*, pages 75–88, 2014.

[12] Mohammad Sabzinejad Farash, Omer Nawaz, Khalid Mahmood, Shehzad Ashraf Chaudhry, and Muhammad Khurram Khan. A Provably Secure RFID Authentication Protocol Based on Elliptic Curve for Healthcare Environments. *J. Med. Syst.*, 40(7):165, 2016.

[13] Prosanta Gope, Ruhul Amin, SK Hafizul Islam, Neeraj Kumar, and Vinod Kumar Bhalla. Lightweight and Privacy-Preserving RFID Authentication Scheme for Distributed IoT Infrastructure With Secure Localization Services For Smart City Environment. *Future Generation Computer Systems*, 2017. , 2017, in press.

[14] Md. Endadul Hoque, Farzana Rahman, and Sheikh Iqbal Ahamed. AnonPri: An Efficient Anonymous Private Authentication Protocol. In *PerCom*, pages 102–110, 2011.

[15] Pekka Jappinen and Finland Lappeenranta. Probabilistic Authentication with Very Lightweight Short Hash. In *ICS*, pages 539–543, 2012.

[16] Yudai Komori, Kazuya Sakai, and Satoshi Fukumoto. Randomized Skip Graph-Based Authentication for Large-Scale RFID Systems. In *WASA*, volume 9798, pages 1–12, 2016.

[17] Yudai Komori, Kazuya Sakai, and Satoshi Fukumoto. RFID Grouping Protocol Made Private. In *DSN*, pages 105–106, 2017.

[18] Yingjiu Li, Robert H. Deng, Junzuo Lai, and Changshe Ma. On two RFID privacy notions and their relations. *ACM Trans. Inf. Syst. Secur.*, 14(4):Article No. 30, 2011.

[19] Jia Liu, Bin Xiao, Shigang Chen, Feng Zhu, and Lijun Chen. Fast RFID Grouping Protocols. In *INFOCOM*, pages 1948 –1956, 2015.

[20] X. Liu, X. Xie, K. Li, B. Xiao, H. Qi J. Wu, and D. Lu. An Additive Model as a Physical Basis for Shadow Fading. *IEEE/ACM Trans. Netw.*, 25(1):278–291, 2017.

[21] Li Lu, Jinsong Han, Lei Hu, Yunhao Liu, and Lionel M. Ni. Dynamic Key-Updating: Privacy-Preserving Authentication for RFID Systems. In *PerCom*, pages 13–22, 2007.

[22] Li Lu, Jinsong Han, Renyi Xiao, and Yunhao Liu. ACTION: Breaking the Privacy Barrier for RFID Systems. In *Infocom*, pages 1951–1961, 2009.

[23] Hanguang Luo, Guangjun Wen, Jian Su, and Zhong Huang. SLAP: Succinct and Lightweight Authentication Protocol for low-cost RFID system. *Wireless Netw.*, 23:1–10, 2016.

[24] David Molnar and David Wagner. Privacy and Security in Library RFID Issues, Practices, and Architectures. In *CCS*, pages 210–219, 2004.

[25] Jihoon Myung, Wonjun Lee, Jaideep Srivastava, and Timothy K. Shih. Tag-Splitting: Adaptive Collision Arbitration Protocols for RFID Tag Identification. *IEEE Trans. Parallel Distrib. Syst.*, 18(6):763–775, 2007.

[26] Y. Oda, R. Tsuchihashi, K. Tsunekawa, and M. Hata. Measured Path Loss and Multipath Propagation Characteristics in UHF and Microwave Frequency Bands for Urban Mobile Communications. In *IEEE VTC-Spring*, 2001.

[27] Axel Poschmann, Gregor Leander, Kai Schramm, and Christof Paar. New Light-Weight Crypto Algorithms for RFID. In *ISCAS*, pages 1843–1846, 2007.

[28] Yan Qiao, Shigang Chen, Tao Li, and Shiping Chen. Energy-Efficient Polling Protocols in RFID Systems. In *Mobihoc*, 2011.

[29] Farzana Rahman, Md Endadul Hoque, and Sheikh Iqbal Ahamed. Anonpri: A secure anonymous private authentication protocol for RFID systems. *Information Sciences*, 379:195–210, 2017.

[30] K. Sakai, W.-S. Ku, R. Zimmermann, and M.-T. Sun. Dynamic Bit Encoding for Privacy Protection Against Correlation Attacks in RFID Backward Channel. *IEEE Transactions on Computers*, 62(1):112–123, 2013.

[31] K. Sakai, M.-T. Sun, W.-S. Ku, and T. H. Lai. Randomized Skip Lists-Based Private Authentication for Large-Scale RFID Systems. In *Mobihoc*, pages 277–280, 2013.

[32] Kazuya Sakai, Min-Te Sun, Wei-Shinn Ku, and Ten H Lai. A Novel Coding Scheme for Secure Communications in Distributed RFID Systems. *IEEE Trans. Comput.*, 65(2):409–421, 2016.

[33] Jari Salo, Lasse Vuokko, Hassan M. El-Sallabi, and Pertti Vainikainen. An Additive Model as a Physical Basis for Shadow Fading. *IEEE Trans. Veh. Technol.*, 56(1):13–26, 2007.

[34] Nitesh Saxena, Md. Borhan Uddin, Jonathan Voris, and N. Asokan. Vibrate-to-Unlock: Mobile Phone Assisted User Authentication to Multiple Personal RFID Tags. In *PerCom*, pages 181–188, 2011.

[35] M.-T. Sun, K. Sakai, W.-S. Ku, T. H. Lai, and Athanasios V. Vasilakos. Private and Secure Tag Access for Large-Scale RFID Systems. *IEEE Trans. Dependable Secur. Comput.*, 13(6):657–671, 2016.

[36] S. A. Weis. Security and Privacy in Radio-Frequency Identification Devices. *Master Thesis, MIT*, 2003.

[37] Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *SPC*, pages 201–212, 2003.

[38] Qingsong Yao, Qong Qi, Jinsong Han, Jizhong Zhao Xiangyang Li, and Yunhao Liu. Randomized RFID Private Authentication. In *PerCom*, pages 1–10, 2009.

[39] Yan Zhang, Laurence T. Yang, and Jiming Chen. *RFID and Sensor Networks: Architectures, Protocols, Security, and Integrations.* CRC Press, 2009.

[40] Z. Zhang and Q. Qi. An Efficient RFID Authentication Protocol To Enhance Patient Medication Safety Using Elliptic Curve Cryptography. *J. Med. Syst.*, 38(5):47, 2014.