Randomized Skip Lists-Based Private Authentication for Large-Scale RFID Systems

Kazuya Sakai[1] Min-Te Sun[2] Wei-Shinn Ku[3] Ten H. Lai[1] Athanasios V. Vasilakos[4]
[1]Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio 43210
[2]Department of Computer Science and Information Engineering, National Central University, Taoyuan 320, Taiwan
[3]Department of Computer Science and Software Engineering, Auburn University, Auburn, Alabama 36849
[4]Department of Computer and Telecommunications Engineering, University of Western Macedonia,Greece
sakai.16@buckeyemail.osu.edu, msun@csie.ncu.edu.tw,
weishinn@auburn.edu, lai@cse.ohio-state.edu, vasilako@ath.forthnet.gr

ABSTRACT

The performance of key authentication and the degree of privacy in large-scale RFID systems are considered by many researchers as tradeoffs. Based on how keys are managed in the system, the privacy preserving tag authentications proposed in the past can be categorized into tree-based and group-based approaches. While a tree-based approach achieves high performance in key authentication, it suffers from the issue of low privacy should a fraction of tags be compromised. On the contrary, while group-based key authentication is relatively invulnerable to compromise attacks, it is not scalable to the large number of tags. In this paper, we propose a new private tag authentication protocol based on skip lists, named Randomized Skip Lists-based Authentication. Without sacrificing the authentication performance, our scheme provides a strong privacy preserving mechanism. Both theoretical and simulation results demonstrate that the proposed scheme meets its design goals and outperforms existing solutions.

1. INTRODUCTION

Radio Frequency Identification (RFID) is widely used to smooth the way of various applications, such as library managements [12], transportation payment, natural habitat monitoring, indoor localization [7, 15], and so on. In these systems, the administrator manages and monitors a large number of objects by reading passive RF tags attached to the objects with an RF reader. To protect the tag's content, low-cost cryptographic operations [6] are conducted during singulation process. Hence, on receiving the tag's reply, the reader must scan all keys to find the corresponding key in order to decrypt the content. When it comes to a large-scale RFID system, the authentication process can take a long time.

To accommodate this issue, a number of private tag authentication protocols with structured key management have been proposed. In these approaches, a unique key and a set of group keys are assigned to each tag. The group keys are shared among several tags and used to confine the search space of the unique key corresponding to a tag's reply. Based on how group keys are managed, they are categorized into two types: tree-based [3,8,10–12,17] and group-based protocols [1,4]. In a tree-based protocol, tags are mapped to leaf nodes in the tree and keys are assigned to internal nodes. Each tag has its unique key and a set of shared keys associated with the nodes from the leaf to the root. By traveling the tree, the reader can securely singulate tags. This results in high authentication efficiency, but discloses a large amount of information once tags in the system are compromised. On the contrary, in a group-based protocol, each tag has two kinds of keys: a unique key and a group key. With this approach, even if one of the group members is compromised, tags in other groups are intact. However, the authentication efficiency of this approach is low.

Therefore, for large-scale RFID systems, the performance and privacy/security of key authentication are commonly seen as tradeoffs. In this research, we propose a scheme that provides both good performance and a high level of privacy/security for a large-scale RFID system. Since both tree-based and group-based structures have pros and cons, we take a different approach based on *skip lists* [13], a data structure with which operations are performed in a logarithmic order like a balanced tree. In our proposed scheme, an interrogator authenticates a tag by traveling skip lists from top to bottom with a random rotation at each level. The analysis and simulation results prove that the proposed scheme is both efficient on authentication complexity and resistant against compromise attacks. In summary, the contributions of this paper are as follows.

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

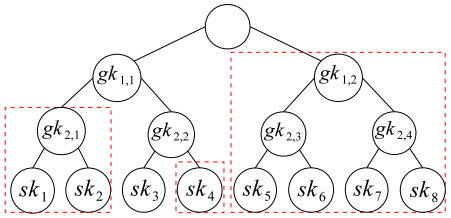

Figure 1: Tree-based.

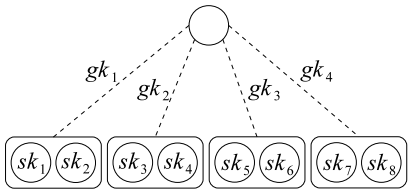

Figure 2: Group-based.

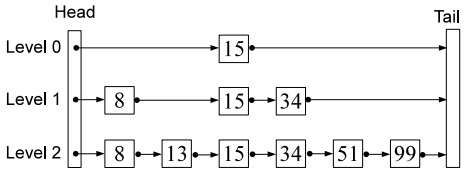

Figure 3: An Example of skip lists.

nance mechanisms for RSLA to adapt to dynamic environments, where existing tags update their keys, and new tags join or leave the system.

- We conduct performance and security analyses to demonstrate that RSLA achieves its design goals: a high level of security/privacy and good performance.
- We evaluate the proposed RSLA by simulations, and validate that RSLA outperforms the existing solutions.

The rest of this paper is organized as follow. Section 2 reviews the existing works. In Section 3, we propose RSLA. Section 4 provides the analyses of the proposed scheme, and Section 5 demonstrates the simulation results. Section 6 concludes this paper.

2. RELATED WORKS

2.1 Private Authentication

Private RFID authentication protocols are classified into two categories: non-encryption-based and encryption-based. Non-encryption-based authentication [2, 9, 14] can be applied to some particular contexts, but not to large-scale RFID systems such as RFID library and super markets. In these applications, it is natural that the provider of the RFID system issues keys to tags before the tag deployment. Therefore, in this paper we focus on encryption-based private authentication. The encryption-based authentication protocols can be further divided into unstructured, tree-based and group-based, which will be elaborated on in the following subsections.

2.1.1 Unstructured Authentication

Due to the computational power constraints of passive tags, traditional cryptographic operations are not practical. Many studies put forth low-cost encryption [6], which relies on only simple functions such as hash, concatenation, and XOR. Weis et al. proposed Hash-lock [16] which uses a hash value to identify tags. However, such an approach requires an RF reader to try all keys in the database to decrypt or compute hash values to validate a tag's reply, which leads to a slow authentication speed proportional to the number of tags in the system. This motivates private authentication to have a structured key management.

2.1.2 Tree-Based Authentication

In tree-based authentication schemes [3, 12], unique keys are mapped to the leaf nodes of a balanced tree, and group keys are mapped to non-leaf nodes. In addition, each tag in the system is associated with a leaf node. A tag obtains its keys on the path from the associated leaf to the root. Thus, each tag has one unique key and a set of group keys, denoted as sk and $GK = \{gk_1, gk_2, ...\}$, respectively. A tag computes a set of hash values with g_i and nonce at each level i of the tree. Starting from the root, an RF reader tries all group keys associated with the children of each non-leaf node. When the reader reaches the bottom of the tree, it applies sk corresponding to the leaf node. Thus, a tree-based protocol runs in $O(log_k N)$, where k is the balancing factor of the tree and N is the number of tags.

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

Tree-based protocols have good performance but result in low anonymity should some tags be compromised. This motivates a number of studies to improve the privacy protection mechanism of tree-based authentications. To alleviate compromise attacks, Lu et al. [10] proposed SAP that augments the tree-based authentication with a dynamic key-updating in which shared keys in the tree are periodically updated. In [8], Li et al. successfully reduce the communication overhead based on cryptographic encodings by having tags reply partial bits of the path indicator.

Other studies utilize a tree in different ways. Lu et al. [11] used a sparse tree to reduce the dependency of shared keys. A path indicator is assigned to each tag for fast authentication. However, as pointed out

by [8], the possible space of path indicators is small and therefore their protocol is vulnerable to the brute force attack against hash values of the path indicator in a tag's reply. In Yao et al. [17], tags do not share any key with other tags, and non-leaf nodes are used as anchors to the corresponding leaf node. By finding the anchor using the random tree walk, a reader will find a valid tag's key. Although Yao et al. claimed that their approach reduces the authentication complexity to $O(1)$, tags are required to perform randomized hash functions, which is not suitable for passive tags with low computational power.

2.1.3 Group-Based Authentication

In group-based authentication schemes [1,4], tags are divided into disjoint groups. Each tag has two keys, a unique key sk and a group key gk. A tag's reply consists of two components encrypted by gk and sk, respectively. A reader first scans all group keys to decrypt the first component, and then applies the unique keys associated with the group to the second component.

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

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

To improve the privacy protection with a group-based scheme, Hoque et al. proposed AnonPri [4] where each tag obtains a set of pseudo IDs from the key issuer. Then, a tag replies with one of the pseudo IDs that it has. A reader first scans all group keys to obtain a pseudo ID in tag's reply, and then tries all unique keys associated with the pseudo ID. To guarantee that AnonPri works, each tag must share every pseudo ID it has with at least two tags in the same group. By doing so, AnonPri slightly improves the privacy preserving mechanism. However, the low authentication efficiency of the group-based approach has not been addressed.

2.2 Skip Lists

Skip lists [13] are a probabilistic data structure that consists of a set of ordered lists as shown in Figure 3. At the lowest level (labeled by Level 2), the list contains all nodes in increasing order of their keys. A node in the list at a level $i > 0$ appears at level $i - 1$ with probability p. Each node in a list has two pointers to the node in the left and right directions. The number of lists in skip lists is $log_{1/p}N$, where N is the input size, and the number of nodes scanned at each level is $\frac{1}{p}$ on average. Thus, search, insert, and delete operations are performed in $O(log_{1/p}N)$ and the space complexity is $O(N)$. Theoretically, skip lists can be considered as an alternative to a balanced tree.

For example, in Figure 3, to find Key 13, we start from the top level list. The list at Level 0 has only one node with Key 15. Since $13 < 15$, we travel toward the left at Level 1. At Level 1, we reach the node with Key 8 which is greater than 13, and then we travel toward the right at Level 2. Finally, we find node with Key 13.

3. PRIVATE AUTHENTICATION PROTOCOL

3.1 Protocol Overview

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

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

3.2 Definitions and Assumptions

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

n_r and n_t represent nonce randomly selected by the reader and a tag, respectively. For a given key K and an input x, the hash function $H(x)$ is assumed to be collision resistant, and an encryption function $E(K, x)$ is implemented by low-cost cryptographic operations [6]. A reader is assumed to have enough computational power to run a decryption function $D(K, x)$ with a key K and an input x.

3.3 Construction of Skip Lists

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

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

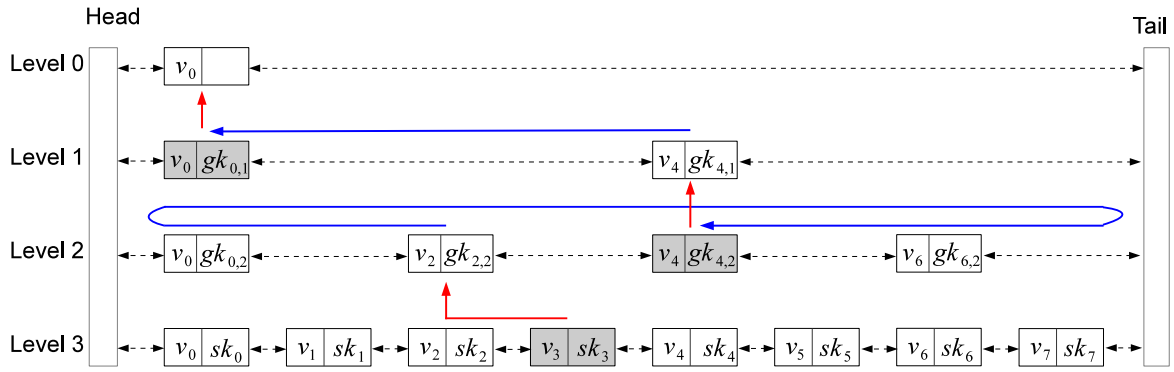

Figure 4: An example of key issuing.

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

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

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

Table 1: Definition of notations.

Symbols	Definition
k	The balancing factor of skip lists
N	The number of tags in the system
η	The height of skip lists, $\lceil log_k N \rceil$
L_i	The list at Level i in skip lists $(0 \leq i \leq \eta)$
v_i	Node i in a list
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 random numbers of Tag i, $\{r_1, r_2, ..., r_{\eta-1}\}$
n_t, n_r	Nonces from a tag and a reader
β	Tag's reply, $\{\beta_1, \beta_2, .., \beta_\eta\}$
N_c	The number of compromised tags in the system
N_g	The number of compromised tags in a group
$E(.)$, $D(.)$	The encryption and decryption functions
$H(.)$	The hash function
A	System anonymity
S_i	Anonymous set that Tag i belongs to

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

3.4 Key Issuing

In RSLA, Tag t has three variables, the unique secret key sk_t, a set of group keys GK_t, and a set of random numbers R_t. Each tag t is randomly assigned to a node, say v_i, in the lowest level list L_η. Starting from v_i, the key issuer traverses to L_0 by shifting to the left for r_j nodes at each L_j $(1 \leq j \leq \eta - 1)$, where r_j is randomly chosen between 0 and $|L_j| - 1$ (i.e., $k^j - 1$). By doing this, the key of the selected node for each level is assigned to a tag.

At v_i in L_η, Tag t obtains the unique key from $v_i.key[\eta]$, and then the pointer moves to $L_{\eta-1}$. When v_i in L_η does not appear in $L_{\eta-1}$, the pointer first moves to node v_j where $j = i - i \ mod \ k$, and then moves to $L_{\eta-1}$. In general, for the current node v_i in L_m, the pointer moves to v_j where $j = i - i \ mod \ k^{\eta-m+1}$, and then goes to L_{m-1}. Thus, this can be seen as a parent and children relation of a k-balanced tree, i.e., v_j in L_{m-1} has k children v_i in L_m $(j \leq i \leq j + k - 1)$.

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

At the end of this process, Tag t has one unique key,

$\eta - 1$ group keys, and $\eta - 1$ random numbers. The pseudo code of the key issuing is given in Algorithm 1.

Algorithm 1 Key Issue

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

Example Consider an RFID system with 8 tags that uses skip lists with $k = 2$ and $\eta = 3$ for key assignment as shown in Figure 4. Tags are mapped to the t-th node in L_3. We illustrate how the key issuer assigns group keys and random numbers to a tag, for instance Tag 3. Starting from v_3, the key issuer traverses to the top level list. First, Tag 3 obtains sk_3 stored at $v_3.key[3]$, and the pointer moves to Level 2 via v_2. Because v_3 does not appear in L_2, the pointer goes to v_2 ($3-3\ mod\ k = 2$), and then moves to Level 2. Assume the key issuer randomly selects $r_2 = 3$ and the pointer shifts to the left by 3. At the same time, 2 is added to R_3. The current pointer is now at v_4 in L_2. The issuer assigns $gk_{4,2}$ stored in $v_4.key[2]$ to Tag 3. This process continues until the issuer reaches L_0. Assume Tag 3 selects $r_1 = 1$ at Level 1. It obtains sk_3, $GK_3 = \{gk_{0,1}, gk_{4,2}\}$, and $R_3 = \{1, 3\}$.

3.5 Authentication

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

Assume Tag t has one unique key sk_t, a set of group keys $GK_t = \{gk_1, gk_2, ..., gk_{\eta-1}\}$, and a set of random numbers $R_t = \{r_1, r_2, ..., r_{\eta-1}\}$. On receiving a query with nonce n_r from the reader, Tag t generates a reply message with nonce n_t. Let $\beta = \{\beta_1, \beta_2, ..., \beta_{\eta-1}\}$ be the reply message. Here, β_i ($i \leq 1 \leq \eta$) consists of a hash value $\beta_i.hash$ and encrypted number $\beta_i.num$ at each level i. The hash value $\beta_i.hash$ is obtained by $H(gk_i||r_{i-1}||n_t||n_r)$ with the base $r_0 = empty$. In other words, $\beta_1.hash = H(gk_1||n_t||n_r)$ because there is no rotation at L_0. The reason that we include the number at the previous level, i.e., r_{i-1} for $\beta_i.hash$, is to enforce dependency between the levels to keep high anonymity. The random number $\beta_i.num$ is encrypted by $E(gk_i, r_i)$. For the last element β_η, the hash value $\beta_\eta.hash$ is defined by $H(sk_t||r_{\eta-1}||n_t||n_r)$ where the unique key is used, and $\beta_\eta.num$ is empty. Finally, the tag sends n_t and β to the reader. Note that β contains η elements. One is computed by sk; the other $\eta - 1$ are computed by gk. The pseudo code of the replying process is illustrated in Algorithm 2.

Algorithm 2 ReplyToReader(n_r)

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

On receiving Tag i's reply, the reader scans group keys associated to nodes from the top level list. At the beginning, the pointer is at node v_0 in L_0. In L_1, there are k nodes, and one of them has the group key $v_i.key[1]$ ($v_i \in L_1$) that matches the group key used for $\beta_1.hash$. After finding the corresponding key used for $\beta_1.hash$, the reader decrypts $\beta_1.num$ with the key. Then, we first move the pointer to L_1 form L_0, and shift the pointer to the right by $\beta_1.num$. If the pointer reaches the tail during shifting, it moves to the head of the same list. Note that the left shift was taken for key assignment by traveling from the lowest level, and on the contrary, the authentication process takes the right shift since the reader travels skip lists from the top. Assume v_i is the current node after shifting right by r_1. The list L_2 has k^2 nodes, but only k nodes v_j ($i \leq j \leq i + k$) need to be scanned. This is because one of the k nodes has the group key for β_2. This process continues until the reader reaches the bottom. Since the key at L_η is unique for a tag, the reader singulates the tag from β. The reader scans no more than k keys at each level $1 \leq i \leq \eta$, hence our skip lists imitate the search operation of a k-balanced tree. During this process, should the reader be unable to find a group key at any level, the tag's reply is invalid and the reader returns a $FAIL$ message. The pseudo code of the authentication process is provided in Algorithm 3.

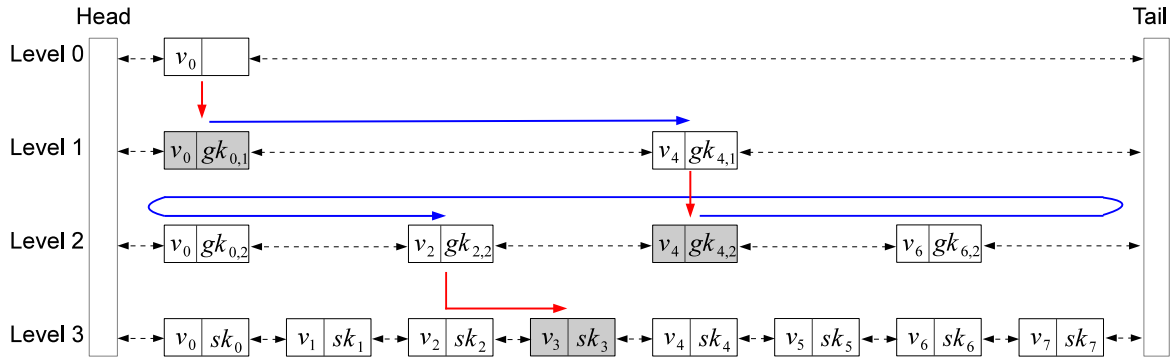

Figure 5: An example of authentication.

Algorithm 3 Authentication(n_r, n_t, β)

```
1: /* β = {β_1, β_2, ..., β_η}*/
2: v_0 ← head /* the pointer to the current node */
3: for j from 1 to η do
4:    /* Scan v.key[j] for k nodes from v_i */
5:    for m from 1 to k do
6:       /* Note that the base r_0 = empty */
7:       if H(v_i.key[j]||r_{j-1}||n_r||n_t) = β_j.hash then
8:          if j == η then
9:             Identify Tag t by the unique key v_i.key[j]
10:         else
11:            r ← D(v_i.key[j], β_j.num)
12:            v_i ← shift to the right by r
13:            j ← j + 1
14:         end if
15:      end if
16:      m ← m + 1
17:   end for
18:   if The key is not found for L_j then
19:      return FAIL
20:   end if
21: end for
22: return t
```

Example We provide an example to demonstrate how the reader authenticates Tag 3 as shown in Figure 5. Tag 3's parameters are sk_3, $GK_3 = \{gk_{0,1}, gk_{4,2}\}$, and $R = \{1, 3\}$. Thus, the reply of Tag 3 β as follows:

$$\beta_1 = H(gk_{0,1}||n_t||n_r), E(gk_{0,1}, 1) \quad (1)$$

$$\beta_2 = H(gk_{4,2}||1||n_t||n_r), E(gk_{4,2}, 3) \quad (2)$$

$$\beta_3 = H(sk_3||3||n_t||n_r),\ empty \quad (3)$$

On receiving the tag's reply n_t and β, the reader travels skip lists as shown in Figure 5. First, the reader scans $v_1.key[1]$ and $v_4.key[1]$ in L_1, i.e., $gk_{0,1}$ and $gk_{4,1}$, to compare the obtained hash value with $\beta_1.hash$. As the key $gk_{0,1}$ works, the reader applies $D(gk_{0,1}, \beta_1.num)$ and obtains $r_1 = 1$. The reader takes the right shift by 1, and moves to v_4. For Level 2, the reader scans two nodes as $k = 2$ in this example, i.e., $v_4.key[2]$ and $v_6.key[2]$. The reader will validate that $gk_{4,2}$ works for $\beta_2.hash$ and obtains $r_2 = 3$ from $\beta_2.num$. This process continues until the reader reaches the lowest level list L_3. At Level 3, the reader scans the unique keys stored at $v_2.key[3]$ and $v_3.key[4]$. The hash value obtained with sk_3 returns the same value with $\beta_3.hash$. Since v_3 corresponds to Tag 3, the reader finally concludes the reply comes from Tag 3.

3.6 Key Update

Secure RFID systems should periodically update shared keys to avoid tag compromise attacks. In SPA [10], the reader first updates keys at a tag when it accesses the tag, and then updates the corresponding keys in the key tree.

However, our RSLA updates keys in the opposite order to simplify the key updating mechanism. First, the reader updates all the keys in all of the skip lists, and updates tags' side upon accessing tags. The new key is obtained by $H(r, v.key[i])$ $(0 \le i \le \eta)$ where r is a random number. The old key at a node, for instance $v.key[i]$ $(1 \le i \le \eta)$, is kept as $n.old\_key[i]$, so that tags with old keys can be singulated. For tags' side, the reader updates a tag's unique key, group keys, and random numbers by Algorithm 1 when it accesses a tag. Therefore, our key-updating mechanism successfully renews the keys in the system while the reader can still access tags with old keys. The pseudo code is given in Algorithm 4.

Algorithm 4 KeyUpdate

```
1: Generate a random number r
2: for for each i from 0 to η do
3:    v ← L_i.head
4:    while v.right ≠ null do
5:       v.old_key[i] ← v.key[i]
6:       v.key[i] ← H(r, v.key[i])
7:       v ← v.right
8:    end while
9: end for
```

3.7 System Maintenance

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

When a new tag joins the system, the system first

tries to find a node in the lowest level list L_η such that no tag is assigned to. If found, the key issuer assigns a unique key, group keys, and random numbers, by Algorithm 1. If there is no such node, a new set of skip lists with the same size as the original one will be created and assign keys to the new tag. Thus, the reader needs to scan $2k$ nodes at the beginning. There are k keys in the original skip lists; the other k keys are in the new skip lists. After this, the reader narrows the search space to either skip lists. This does not affect authentication efficiency, since only k more keys need to be scanned by the reader only for the first element of the reply, i.e., β_1.

The tag removing process is simple. The system removes the tag from the corresponding node in L_η.

3.8 Implementation Issue

One of the implementation issues is the small domain of the random numbers $R = \{r_1, r_2, ...\}$, as each element r_i is randomly selected between 0 and $k^i - 1$. To avoid the brute force attack, the domain should be large enough. Thus, we can select r_i between 0 and 2^{32} (r_i will be 32 bits for all i), and then take $r_i \ mod \ k^i$ shift at Level i. This virtually protects a tag's replies from the brute force attack.

4. ANALYSIS

4.1 Performance Analysis

Theorems 1 and 2 show the upper bound and the average time complexity of authentication in RSLA, respectively. The authentication running time is determined by the computations of hash and decryption functions by a reader.

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

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

Theorem 2 *Given N and k, the average running time of RSLA is $\frac{k+3}{2} log_k N$.*

Proof: Let X_i be the random variable which represents the number of times a reader computes the hash values at Level i. Since the reader scans at most k nodes in L_i where $1 \leq i \leq \eta$, we have $1 \leq X \leq k$. Thus, the expected value $E[X]$ is obtained by $\sum_{i=1}^{i=k} \frac{i}{k} = \frac{k+1}{2}$.

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

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

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

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

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

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

4.2 Unlinkability

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

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

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

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

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

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

4.3 Analysis for Compromise Attacks

A privacy protection mechanism against the compromise attack can be measured by anonymity. Anonymity is a state of not being identifiable within an anonymous set. Let S_i be the anonymous set that Tag i belongs to. According to [4], the system anonymity, denoted as A, can be formulated as Equation 4.

$$A = \frac{1}{N^2} \sum_i |S_i|^2 \tag{4}$$

When no tag is compromised, $|S_i|$ for any tag i equals N and therefore A equals 1. The system anonymity decreases as the number of compromised tags increases.

When a tag is compromised, the adversary will have the unique key, group keys, and random numbers. Since RSLA takes a random shift at each level of skip lists, any pair of two tags cannot be linked unless they have all their group keys in common. Let T_c be the set of compromised tags, and N_c ($1 \leq N_c \leq N - 1$) be $|T_c|$. A compromised tag t in T_c always has $|S_t| = 1$. On the other hand, we can obtain $|S_{t'}|$ which an uncompromised tag t' ($t' \notin T_c$) belongs to as follows. By the definition of search over skip lists, the nodes in the lowest level are divided into a number of groups with each having k nodes (which is similar to branches in a balanced tree). If there is at least one compromised tag t in T_c that has all group keys in common as Tag t', the adversary knows the group that t' belongs to. Let N_g ($0 \leq N_g \leq k - 1$) be the number of compromised tags in the same group as t'. We can derive two cases as follows.

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

Note that $P[\exists t \in T_c\ s.t.\ GK_t = GK_{t'}]$ is $1-(1-\frac{1}{k^{\eta-1}})^{N_c}$ and thus is very small. By computing an anonymous set size for each tag, we can obtain the system anonymity by Equation 4.

Example Assume tag 3 is compromised in Figure 5, where $N = 8$ and $k = 2$. Tag 2's anonymous set size will be 1 in Case 1, and 7 in Case 2. The anonymous set size of other tags (Tag 0, 1, and 4 to 7) will be 2 in Case 1, and 7 in Case 2.

Next, we formulate the average anonymous set size of a tag when N_c tags are compromised. Let us say $\sigma = (1 - \frac{1}{k^{\eta-1}})^{N_c}$. The expected anonymous set size of an uncompromised tag $E[|S|]$ is computed by Equation 5.

$$E[|S|] = (1 - \sigma)(k - E[N_g]) + \sigma(N - N_c) \tag{5}$$

Here, $E[N_g]$ can be obtained by

$$E[N_g] = \sum_{i=0}^{k-1} (k - i) \binom{N_c}{i} (\frac{k}{N})^i (\frac{N - k}{N})^{k-1-i}. \tag{6}$$

4.4 Qualitative Security Analysis

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

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

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

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

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

5. PERFORMANCE EVALUATION

To evaluate the performance of the proposed RSLA, simulations are conducted with existing solutions, including static tree [12], SPA [10], group-based [1], and AnonPri [4].

5.1 Simulation Configuration

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

The parameters for each protocol is set to be as follows. Unless specified, the balancing factor k in RSLA is set to be 2, i.e., the skip lists behave like a balanced binary tree. For fair comparison, the static tree and SPA is implemented with a balanced binary tree. In group-based protocols, the size of each group is 64, which is the same setting as [4]. For AnonPri, the size of pseudo IDs pool in the system and the number of pseudo IDs that each tag has are set to be 1000 and 10, respectively. In addition, we assume that AnonPri always succeeds, i.e., we initialize key issuing to guarantee that a tag shares its pseudo ID with at least two members in the group.

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

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

Dynamic Systems - In the dynamic systems scenario, tags periodically update their keys. First, N_c tags are randomly selected as being compromised. Second, we measure the system anonymity. Third, another set of N_c tags is randomly selected and they update their keys. N_c ranges from 1 to 512. Note that the system anonymity is measured before tags update their keys, since the system is more vulnerable when tags are just compromised. In addition, static tree, group-based, and AnonPri do not have a key updating mechanism, and thus we exclude them from the consideration in this scenario.

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

5.2 Simulation Results of Static Systems

Figure 6 illustrates the system anonymity with respect to the number of compromised tags. Clearly, RSLA achieves much higher anonymity than other protocols, and significant improvement from the existing solutions can be seen. As indicated in [5], the anonymity of AnonPri and the group-based protocol is similar.

Figure 7 demonstrates the authentication speed with respect to the number of tags. Since skip lists and a tree structure run in $log_k N$, both RSLA and the static tree can quickly singulate a tag. In addition to computing a hash function, RSLA is required to decrypt a random number at each level of skip lists, and so it incurs slightly high overhead compared with the static tree. In contrast, AnonPri and the group-based protocol take a much longer time for authentication as the scale of the system increases.

Figure 8 presents the number of unique keys and group keys in the system. RSLA has the same amount of key storage cost as the static tree, since the construction of our skip lists creates the same number of nodes as a balanced tree. Although AnonPri and the group-based protocol do not require that much storage cost compared with ours, the difference is small.

5.3 Simulation Results of Dynamic Systems

Figure 9 shows the anonymity of RSLA and SPA with respect to the number of compromised tags. From the figure, we can see that RSLA improves the anonymity compared with SPA, especially when a large number of tags are compromised. Therefore, we can say that our RSLA is the best alternative for a tree-based authentication protocol.

5.4 The Optimization of Skip Lists

Figure 10 depicts the anonymity of RSLA with different balancing factors with respect to the number of compromised tags. By Theorem 6, two tags t and t' are indistinguishable as long as $GK_t \neq GK_{t'}$. Thus, the balancing factor k should minimize $P[GK_t = GK_{t'}]$. As $\eta = \lceil log_k N \rceil = log_k N + c$ where $0 \leq c \leq 1$, we can derive $P[GK_t = GK_{t'}] = \frac{1}{k^{\eta-1}} = \frac{k^{1-c}}{N}$. Since $k \geq 2$, the optimal value is $k = 2$ for high anonymity, and anonymity decreases as k decreases. This figure validates our analysis.

Figure 11 shows the authentication time and the number of keys in the system required by RSLA with different balancing factors with respect to the number of tags. Although the balancing factor affects authentication speed, the increase of authentication time is small. On the contrary, the value of k is critical to the key storage cost. For example, the number of keys is large when $k = 14$. Recall in this simulation, N is set to be 4096. We have $14^3 < 4096 < 14^4$ and thus skip lists must contain 4 lists. This indicates that the system has 7050 keys (2744 shared keys and 4096 unique keys). On the contrary, When $k = 16$, $16^3 = 4096$ and the number of keys in the system will be 4368. This implies

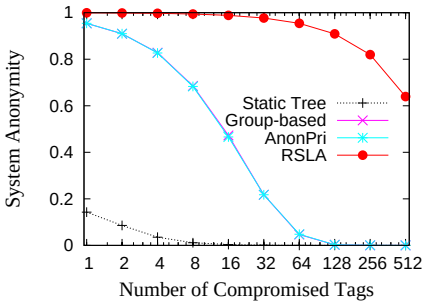

Figure 6: System anonymity.

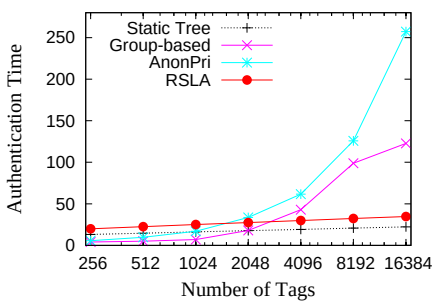

Figure 7: Authentication speed.

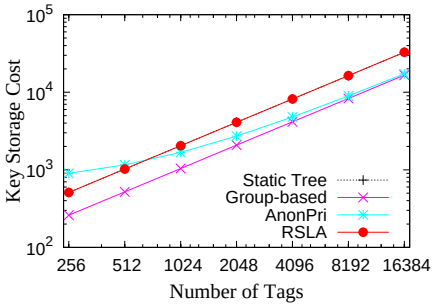

Figure 8: Storage cost.

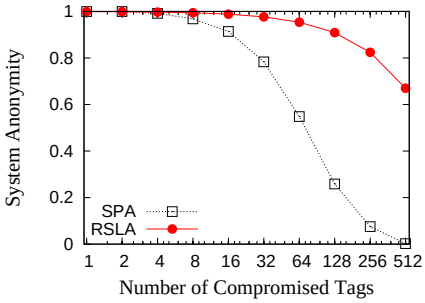

Figure 9: System anonymity.

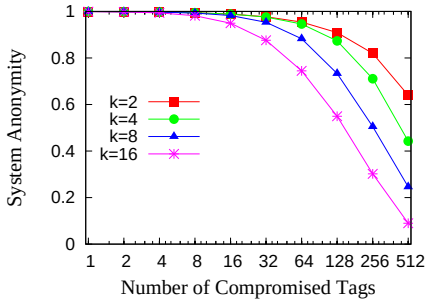

Figure 10: Anonymity with different k values.

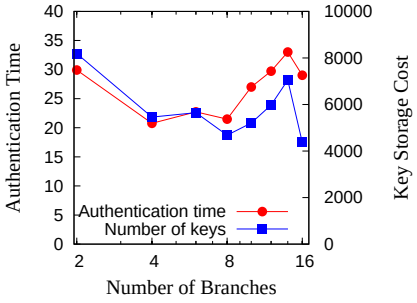

Figure 11: Performance with different k values.

that the balancing factor has a significant impact on the authentication speed and the degree of privacy.

6. CONCLUSION

Large-scale RFID systems always have tradeoffs between performance and security/privacy. The private authentication protocols proposed in the past are either slow or vulnerable to active attacks. In this paper, we propose RSLA which provides both high authentication efficiency and a strong privacy protection mechanism. RSLA relies on skip lists, a different data structure from the existing solutions. In addition, performance and security/privacy analyses are conducted. Our simulations demonstrate that our RSLA outperforms existing solutions in terms of authentication speed and degree of privacy. We believe the proposed skip lists-based approach is the most suitable authentication scheme for the next generation RFID systems.

7. REFERENCES

[1] G. Avoine, L. Buttyan, T. Holczer, and I. Vajda. Group-based Private Authentication. In *WoWMoM*, pages 1–6, 2007.

[2] A. Czeskis, K. Koscher, J. R. Smith, and T. Kohno. RFIDs and Secret Handshakes: Defending against Ghost-and-Leech Attacks and Unauthorized Reads with Context-Aware Communications. In *CCS*, pages 479–490, 2008.

[3] T. Dimitriou. A Secure and Efficient RFID Protocol that cold make Big Brother (partially) Obsolete. In *PerCom*, pages 269–275, 2006.

[4] M. E. Hoque, F. Rahman, and S. I. Ahamed. AnonPri: An Efficient Anonymous Private Authentication Protocol. In *PerCom*, pages 102–110, 2011.

[5] C. Huang and H. Min. A New Method of Synchronization for RFID Digital Receivers. In *ICSICT*, pages 1595–1597, 2006.

[6] A. Juels. Minimalist Cryptography for Low-Cost RFID Tags. In *SCN*, pages 149–164, 2004.

[7] W.-S. Ku, K. Sakai, and M.-T. Sun. The Optimal k-Covering Tag Deployment for RFID-Based Localization. *Special Issues of JNCA on RFID Technology, Systems, and Applications*, 34(3):914–924, 2011.

[8] T. Li, W. Luo, Z. Mo, and S. Chen. Privacy-Preserving RFID Authentication based on Cryptographical Encoding. In *Infocom*, pages 2174–2182, 2012.

[9] T.-L. Lim, T. Li, and S.-L. Yeo. Randomized Bit Encoding for Stronger Backward Channel Protection in RFID Systems. In *PerCom*, pages 40–49, 2008.

[10] L. Lu, J. Han, L. Hu, Y. Liu, and L. M. Ni. Dynamic Key-Updating: Privacy-Preserving Authentication for RFID Systems. In *PerCom*, pages 13–22, 2007.

[11] L. Lu, J. Han, R. Xiao, and Y. Liu. ACTION: Breaking the Privacy Barrier for RFID Systems. In *Infocom*, pages 1951–1961, 2009.

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

[13] W. Pugh. Skip Lists: a Probabilistic Alternative to Balanced Trees. *Comms. of the ACM*, 33(6):668–676, 1990.

[14] 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.

[15] S. Wagner, M. Handte, M. Zuniga, and P. J. Marron. On Optimal Tag Placement for Indoor Localization. In *PerCom*, pages 162–170, 2012.

[16] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *SPC*, pages 201–212, 2003.

[17] Q. Yao, Q. Qi, J. Han, J. Z. X. Li, and Y. Liu. Randomized RFID Private Authentication. In *PerCom*, pages 1–10, 2009.