On The Performance Bound of Structured Key-Based RFID Authentication

Kazuya Sakai*, Min-Te Sun†, Wei-Shinn Ku‡, and Ten H. Lai§
*Dept. of Electrical Eng. and Comput. Sci., Tokyo Metropolitan University, Asahigaoka 6-6, Hino, Tokyo 191-0065, Japan.
†Department of Computer Science and Information Engineering, National Central University, Taoyuan 320, Taiwan.
‡Department of Computer Science and Software Engineering, Auburn University, Auburn, Alabama 36849–5347.
§Department of Computer Science and Engineering, The Ohio State University, Columbus, 42312 Ohio, USA.
ksakai@tmu.ac.jp, msun@csie.ncu.edu.tw, weishinn@auburn.edu, lai.1@osu.edu

***Abstract*—Designing fast and secure RFID private authentication with structured key management is one of the most essential components for RFID-enabled large-scale object management. Since group keys are shared by some tags in structured key-based authentication, physical tampering of tags, so called the compromise attack, may enable the adversary to obtain group keys stored in the compromised tags, which in turn can be used to distinguish other tags. All existing structured key-based protocols try to reduce the common group key effect to preserve high privacy. However, the theoretical bound of weak privacy achievable by structured key-based authentication remains unknown. In this paper, we investigate weak privacy in RFID authentication. To this end, we first formulate a mathematical model which identifies the probability of two tags being linked with respect to the number of group keys. Our model shows that the existing solutions are far from the ultimate goal in weak privacy. Then, we propose a k-neighbor graph-based RFID authentication (KNGA) protocol, where random walk over a k-neighbor graph is performed. In addition, we show that KNGA achieves the performance bound, and we then quantify the degree of privacy by anonymity. Finally, the extensive simulations demonstrate that the proposed protocol successfully achieves its design goals.**

***Index Terms*—Radio Frequency Identification, RFID, privacy and security, private authentication**

I. Introduction

Radio Frequency Identification (RFID) is enabling tagging technologies for fast and secure object identification in applications such as supply chain managements [1], transportation payments, and smart cards [2]. The security and privacy concerns are of significance for its wide adoption. To preserve the trustworthiness of large-scale RFID systems, private authentication protocols with light-weight cryptographic operations are widely studied.

Two notions of RFID privacy are defined in [3]. One is *indistinguishability-based* privacy (ind-privacy) which guarantees that the replies from two tags are not computationally distinguishable without knowing internal states; the other is *unpredictability-based* privacy (unp-privacy) where no adversary can predict the output from both a tag and a reader engaging in an interrogation protocol. Thus, in the unp-privacy model, two tags, each of which belongs to two different RFID systems, are not computationally distinguishable as long as the outputs of these protocols are of the same length. The necessary and sufficient condition to achieve the ind-privacy and/or ump-privacy is that the computational power of tags can be used to construct a pseudo random function family (PRF). Unp-privacy implies ind-privacy, but not vice versa.

For each notion of privacy, *strong* and *weak* levels of privacy are defined in [3]. Strong privacy indicates that ind-privacy/unp-privacy holds even when some tags in the system are compromised and the keys associated with them are disclosed to adversaries. In the weak privacy model, ind-privacy/unp-privacy is guaranteed as long as no tag is compromised. Any private authentication protocol with a structured key management is unfortunately not able to achieve strong privacy should some tags in the system be compromised. This is because group keys are shared by several tags and an adversary can distinguish tag's replies from compromised keys in the past interrogations.

To achieve high degree of weak privacy, all existing structured key-based authentication protocols known to us try to reduce the probability of two tags being linked by the adversary who holds the keys from the compromised tags [4]–[11]. However, the performance bound of weak privacy in terms of the correlation probability remains unknown. Achieving the performance bound has not only theoretical significance, but also practical importance to understand the fundamental security and privacy issues in monitoring and managing a large number of tagged items. Therefore, in this paper, we seek to understand the performance bound of achievable weak privacy in private authentication. The contributions of this paper are as follows.

- First, we derive the performance bound of weak privacy that any structured private RFID authentication protocol can achieve. To formulate the limitation of all protocols, we quantify the possible correlation probability of two tags with respect to the number of group keys. By doing this, we demonstrate that the correlation probability of the existing solutions is much larger than the performance bound. This gap indicates that there is plenty of room for improving the existing private authentication protocols.
- Second, we propose a k-neighbor graph-based authentication scheme (KNGA), which uses a k-neighbor graph as the key structure for a given balancing factor k. Given the number of group keys, the proposed scheme reaches

978-1-5386-9148-9/19/$31.00 ©2019 IEEE

the performance bound of weak privacy and has the same logarithmic authentication speed and the same communications cost as the existing solutions.
- Third, we build an extended anonymity as the privacy metric under the compromise attack. Unlike the existing anonymity definition provided in [9], [10], our anonymity model clarifies the strong and weak privacy, where uncompromised tags' anonymity remains 1 as long as they are not linked with compromised tags.
- Finally, we have conducted extensive simulations to demonstrate that our KNGA outperforms the existing solutions in terms of system anonymity in keeping with reasonable key storage and computation cost.

The rest of this paper is organized as follows. Related works are reviewed in Section II. Section III provides the preliminaries. The performance bound of structured key-based authentication is presented in IV. In Section V, we propose the KNGA protocol. Both the quantitative security/privacy analyses of our KNGA are provided in Sections VI. The performance of the proposed scheme is evaluated by simulations in Section VII. Section VIII concludes this paper.

II. Related Work

The security issue considered in this paper is private authentication protocols for large-scale RFID systems, where secret keys are assigned to tags before deployment. Due to the computational weakness of passive tags, the light-weight operations, e.g., the XOR, concatenation, 16-bit pseudorandom generator, and collision resistant hash function, can be used as building blocks [12]. While there are many security notions for different contexts, in this paper, we follow the notion of RFID privacy provided in [3]. We elaborate on the private authentication protocols in the subsequent subsections.

A. Unstructured-Based Authentication

Hash-lock [13] is the baseline of private authentication protocols, where each tag replies a hashed ID with its unique key concatenating nonce. Then, the reader scans all the keys in the back-end-server to identify the tag. As a result, any protocol in this category causes slow authentication, i.e., have the time complexity of $O(n)$, where n is the number of tags in the system. This motivates many researchers to design private authentication protocols with structured key management.

B. Structured-Based Authentication

In structured-based authentication protocols, each tag stores its unique key and a set of group keys, and each group key is shared among several tags. A tag's reply consists of a set of hashed values generated using its unique key and each of its group keys. In the tree-based protocols [4], the unique keys are located at the leaf nodes of a balanced tree and group keys are mapped to non-leaf nodes. Each tag is assigned to one of the leaf nodes and obtains the unique key at its leaf node as well as the set of group keys on the non-leaf nodes along the path to the root. The reader singulates a tag by traveling the tree from the root to the corresponding leaf node. Many tree-based variants [5]–[7] are proposed to improve the tree-based protocol. Dynamic key updating in [5] securely updates the keys in the key structure to counteract the compromise attack, where some tags are physically tampered and the group keys associated with compromised tags are leaked. ACTION [6] uses a sparse tree to reduce the correlation of group keys shared among tags, and ECNP [7] reduces the length of the authentication path in a tree by cryptographic encoding. Including the key search and hash computation, all the tree-based protocols require $O\left(\log_k n\right)$ overhead, where k is the balancing factor.

The group-based protocol [8] and AnonPri [9] divide the tags in the system into disjoint groups, and each tag maintains its unique key and the group key of the group to which it belongs. The reader first scans all the group keys to confine key search space, and then, tries the unique keys associated with the corresponding group. However, group-based protocols still need $O(n)$ for authentication.

C. Private Authentication with $O(1)$ Overhead

There exist authentication protocols with $O(1)$ authentication overhead, such as [14]–[17]. RWP [14] explicitly assigns each tag with no common keys, where an internal node in a tree is used as an anchor referring to the corresponding leaf node. Although the authentication complexity is claimed as $O(1)$, RWP fails to consider the search cost of the hashed components in a tag's reply from the back-end server. Similarly, LAST [15] insists that their protocol incurs only $O(1)$ overhead, but the key searching is again not considered as a part of authentication. Recently, ETAP [16], which is a token-based protocol and very different from any of the existing works, was proposed. In ETAP, a set of tokens is used at each tag to simulate the pseudo randomness of a hash function, and then, the reader maps the received token to the hash index of the corresponding tag entry. The authors claim that the hash index provides random access in $O(1)$. However, we argue that this is the average performance, and thus, ETAP incurs $O(n)$ in the worst case. The protocol in [17] achieves $O(1)$ access with the database containing pre-computed truncated hashed values of possible tags' replies. This approach adds too much storage cost. million tags. Therefore, the protocols in this category cannot be fairly compared with the structured-key authentication protocols.

D. Randomized Structured-Based Authentication

The protocols in this category are mostly related to this paper. In structured-based authentication, two tags can be linked from the group keys shared in common. To remedy this issue, randomized structured-based approaches [10], [11], [18] were proposed, where random walking over a data structure is incorporated in keeping with the worst case authentication speed being $O\left(\log_k n\right)$. RSLA [10] uses skip lists as a balanced tree alternative. The difference from tree-base protocols is that random shifting is taken at each level of the structure. With this randomization, two tags are never

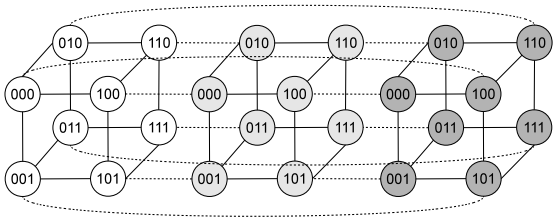

Fig. 1. An example of 4-neighbor graphs.

correlated unless they have exactly the same set of group keys under the compromise attack. RSGA [11] further reduces the probability that two tags are correlated from group keys by using advanced data structure, skip graphs. The existing solutions try to alleviate the common keys effect among tags. Nevertheless, none of them have reached the performance bound.

III. Preliminaries

A k-D hypercube consists of 2^k vertices, and two vertices are connected if and only if the Hamming distance between the IDs of two vertices equals one. A k-D hypercube network (HCN) is a graph in which a set of k-D hypercubes are connected together. For example, Figure 1 illustrates a 3-D HCN, where a circle represents a node, a solid line represents the link between two nodes in the same hypercube, and a dotted line represents the link between two nodes in different hypercubes. In this graph, there are three 3-D hypercubes colored white, light gray, and dark gray, respectively.

A graph is said to be a *k-neighbor graph* if and only if every node has exactly k neighbors. The graph in Figure 1 is a 4-neighbor graph, since every vertex in the HCN has exactly 4 neighbors. Note that the k-neighbor graph is different from the k-connected graph, where each node has at least k neighbors.

IV. The Performance Bound of Weak Privacy

A. Notations and Definitions

An RFID system consists of one RF reader and a set of n RF tags, represented by $RFID(\mathcal{R}, \mathcal{T})$, where $\mathcal{T} = \{t_1, t_2, ..., t_n\}$. Nonce is denoted by r_r and r_t, each of which is randomly selected by a reader and a tag. We denote a collision-resistant hash function by $H(.)$, a pseudorandom function family (PRF) by $F(.)$, an encryption function by $E(.)$, and a decryption function by $D(.)$, respectively.

Each tag, say tag t, stores a set of m group keys $GK_t = \{gk_1, gk_2, .., gk_m\}$ and one unique key sk_t. In a key structure, there are n_g group keys. Both m and n_g are determined by the number of tags n in the system and the balancing factor k of the data structure. The notations used in this paper are listed in Table I.

B. The Correlation Probability

In this section, we seek to understand the performance bound of the weak privacy model. Consider an RFID system with n tags, one of which is compromised. If an authentication protocol achieves strong privacy, then the $n-1$ uncompromised tags are anonymous. That is, no adversary can identify the other tags with probability greater than $1/(n-1)$. In this case, the anonymous set is of size $n-1$. However, in an authentication protocol with structured key management, the anonymous set size significantly decreases, as some tags share the same group keys. With a randomized key structure, the anonymous set size remains $n-1$, as long as a tag does not have exactly the same set of group keys as the compromised tag. We refer to the probability that two tags have the same set of group keys as the *correlation probability*, which is formally defined as follows.

TABLE I
Definition of notation.

Symbols	Definition
$\mathcal{R}$	The reader in the RFID system
$\mathcal{T}$	The set of tags in the RFID system
t_i	Tag i
n	The number of tags in a system, $\lvert\mathcal{T}\rvert$
n_g	The number of group keys in the key structure
m	The number of group keys assigned to each tag
sk_i	Tag i's unique secret key
GK_t	A set of group keys of tag t, $\{gk_1, gk_2, ..., gk_m\}$
RN_t	A set of rand. numbers of tag t, $\{rn_1, rn_2, ..., rn_m\}$
r_t, r_r	Nonces from a tag and a reader
k	The balancing factor of data structure
α, β, γ	A tag's reply
π	The verification code from a reader

Definition 1 (The correlation probability) *Suppose two tags t and t' are each assigned a random set of group keys, say GK_t and $GK_{t'}$, respectively, with $|GK_t| = |GK_{t'}|$. Then, their correlation probability is defined as* $\Pr[GK_t = GK_{t'}]$.

Note that in order to achieve unp-privacy, any two tags must have the same number of group keys, or their number of replies would be different and render them distinguishable. Given a fixed size of group keys, the lower bound of the correlation probability that two tags will be linked can be computed. In other words, the performance bound of weak privacy can be defined by the correlation probability with respect to the number of group keys in the system.

C. The Performance Bound

Let n_g be the number of group keys in a key structure, which is a function of the number of tags in the system. We will discuss how to compute n_g for each different key structure later. Let m be the number of group keys each tag obtains. Consider that t and t' have a set of group keys, $GK_t = \{gk_1, gk_2, ..., gk_m\}$ and $GK_{t'} = \{gk'_1, gk'_2, ..., gk'_m\}$, respectively. Each group key is assumed to be randomly assigned by the uniform distribution. Note that keys must be selected by the uniform distribution to achieve the highest degree of privacy [19]. In general, for the i-th group key, $\Pr[gk_i = gk'_i] = (m-i+1)/(n_g-i+1)$ for $1 \le i \le m$. Therefore, given two parameters, n_g and m, the lower bound of the correlation probability of two tags is obtained by

$$\Pr[GK_t = GK_{t'} \mid n_g, m] \ge \prod_{i=1}^{m} \frac{m-i+1}{n_g-i+1}. \qquad (1)$$

Note that the set of group keys assigned to each tag is selected by a permutation of all the group keys, and each

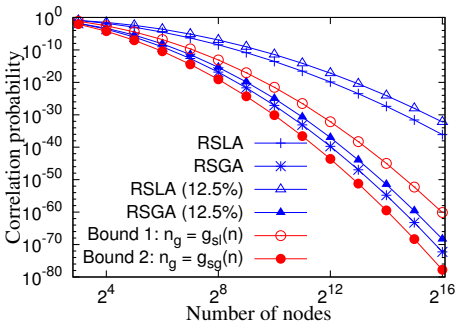

Fig. 2. The bound of weak privacy.

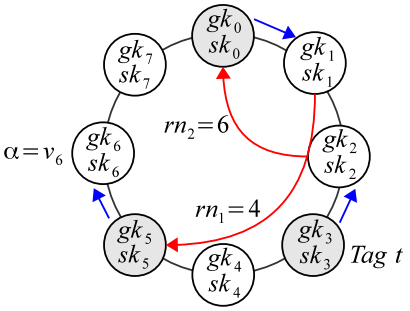

Fig. 3. Key Issuing.

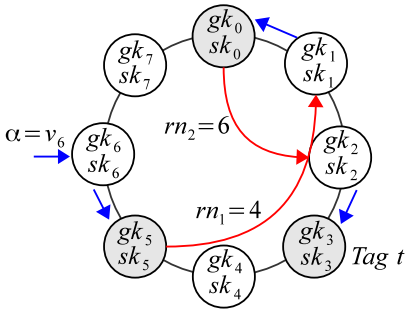

Fig. 4. Authentication.

group key can be used for any level. On the other hand, both skip lists [11] and skip graphs [10] are hierarchical, and the keys at the i-th level are never used in another level. Therefore, the order of group keys in RSLA and RSGA matters.

D. The Analysis

In this subsection, we will show the gap between the correlation probability achievable by the existing solutions and the performance bound.

In skip lists [10] and a skip graph [11], there is a constant number of nodes in a key structure. Let $I(.)$ be a monotonically increasing step function, and we define $I(k,n) = k^{\lceil \log_k n \rceil}$. For both RSLA and RSGA, the number of group keys assigned to each tag is computed by $m = \lceil \log_k n \rceil - 1$. The correlation probability of RSLA [10] and RSGA [11] are obtained by Equations 2 and 3, respectively.

$$\Pr[GK_t = GK_{t'}|RSLA] = \prod_{i=1}^{m} \frac{1}{k^i} \quad (2)$$

$$\Pr[GK_t = GK_{t'}|RSGA] = \left(\frac{1}{I(k,n)}\right)^m \quad (3)$$

RSGA achieves a smaller correlation probability than RSLA. This is because the number of group keys in a skip graph is greater than that in skip lists, when the input size n is the same. We derive the gap between the performance of the existing solutions and the bound as follows. The number of group keys in a key structure is a function of the number of tags in the system, denoted by $n_g := g(n)$. In skip lists and a skip graph, the number of internal nodes, each of which corresponds to the same group key, are determined by the number of tags. Thus, the number of group keys in RSLA is obtained by $g_{sl}(n) = \sum_{i=1}^{\lceil \log_k n \rceil - 1} k^i$, and that in RSGA is computed by $g_{sg}(n) = I(k,n)(\lceil \log_k n \rceil - 1)$. The number of group keys that each tag stores is computed by $m = \lceil \log_k n \rceil - 1$, since the number of levels in skip lists and a skip graph is $\lceil \log_k n \rceil$ and the leaf nodes do not store group keys. For the given parameters $g_{sl}(n)$, $g_{sg}(n)$, and m, the performance bound of RSLA and RSGA are obtained from Equation 1.

Figure 2 shows the correlation probability of the numerical results and their performance bound for RSLA and RSGA. In this figure, Bound 1 is defined as $\Pr[GK_t = GK_{t'}|g_{sl}(n), \lceil \log_k n \rceil - 1]$, and Bound 2 is defined as $\Pr[GK_t = GK_{t'}|g_{sl}(n), \lceil \log_k n \rceil - 1]$. Naturally, the correlation probability decreases as the number of tags increases. This is because the number of group keys m that each tag stores increases, and as a result, the correlation probability decreases. The gaps between the numerical results and their bound can be observed in Figure 2, i.e., the gap between RSLA and Bound 1 is 10^{24}, and that between RSGA and Bound 2 is 10^5, respectively. In addition, the expected correlation probabilities of RSLA and RSGA are plotted, when one eighth (12.5%) of the tags are compromised, which are labeled by RSLA (12.5%) and RSGA (12.5%), respectively. It can be observed that the gap is further enlarged when some portion of tags is compromised. Therefore, there is still a significant space to improve the private authentication with structured key management, and this is the motivation of this research.

V. k-Neighbor Graph-Based Private Authentication

A. The Basic Idea

The data structures used in the existing solutions, i.e., trees, skip lists, skip graphs, and their variants, have two issues. One issue is the property of hierarchy, where there exist successor and descendant relations among the internal nodes in a tree, skip lists, and a skip graph. Such a property prevents a protocol from fully utilizing the random walks over a key structure. As a result, the existing solution cannot reach the lower bound of the correlation probability. Thus, the relation among the internal nodes in a key structure should be flat.

The other issue is that the number of keys and random numbers stored in a tag's memory increases in proportion to the total number of tags. Since the tag's memory is of limited size, e.g., a 512-bit, each tag cannot store too many keys. In other words, for a given storage constraint of tags, the number of tags supported by an RFID system is bounded. Therefore, the number of group keys and random numbers should be independent from the number of tags to support scalable RFID systems.

Based on the aforementioned design goals, we apply the k-neighbor graph as a key structure from a randomly selected entry node, where a pointer moves over the graph by scanning k neighboring nodes and taking a random jump several times.

B. Overview

Similar to the other private authentication protocols, the proposed k-neighbor graph-based authentication (KNGA) protocol has four components: the key initialization, authentication,

Algorithm 1 KeyIssue($\mathcal{T}$, $\{v_0, v_i, ..., v_{n_g}\}$)

```
/* The key issuer does following. */
Each tag t is assigned to node v_i.
/* For each tag t, the key issuer does following. */
for each tag t in the system do
    RN_t = {}, GK_t = {}
    sk_t ← v_i.sk /* v_i is the current node. */
    v_i ←_rand N(v_i).
    for j from m to 1 do
        rn_j ←_rand [0, n_g − 1]
        u ← (i + rn_j) mod n_g
        v_i ← v_u /* The pointer moves. */
        Add rn_j to RN_t and v_i.gk to GK_t.
        j ← j − 1
    α ←_rand N(v_i)
    Tag t stores sk_t, GK_t, RN_t, and α.
```

key updating, and system maintenance. Each phase, except the system maintenance, is very different from existing solutions.

The input parameters are the balancing factor k, the number of tags n, the number of group keys n_g, and the number of group keys m that each tag stores. In the key initialization, a set of $(k-1)$-D hypercubes are generated and connected each other to form a k-neighbor graph. Each node of the k-neighbor graph contains a unique private key and a group key. Each tag is randomly located at one of the nodes and obtains a unique key, a set of group keys, a set of random numbers, and the pointer to its entry node, by randomly walking over the k-neighbor graph. In the authentication process, the RF reader in the system can securely singulate individual tags. On receiving a query from the reader, a tag computes its reply using the set of parameters provided in the key initialization phase. Then, the reader identifies the tag by traveling the k-neighbor graph starting from the entry node of the tag. Finally, the reader replies with a verification code to the tag for mutual authentication. In the key updating, each tag updates its secret and group keys for every interrogation. At the same time, the keys maintained by the reader are also updated. In the system maintenance, the k-neighbor graph may grow and shrink to handle dynamic environments where new tags join to and old tags leave from the system. Elaboration of each phase follows in subsequent subsections.

C. Construction of A k-Neighbor Graph

The input parameters are the balancing factor k ($k \geq 2$), the number of tags n, the number of group keys n_g, and the number of group keys m that each tag stores. Note that $2^{k-1} \leq n$ must hold for generating valid key structure. Given these parameters, a set of $(k-1)$-D hypercubes with each having 2^{k-1} nodes are generated. The number of hypercubes, denoted by h, can be arbitrary, as long as $h \geq n/2^{k-1}$ to support all the n tags. At this moment, each node has $(k-1)$ direct neighbors, and a k-neighbor graph is constructed by connecting these h hypercubes. That is, each node is connected with another node in a different hypercube. One way to do this is to connect the hybercubes in a ring structure. Let $V^{(i)}$ be a set of the nodes in the i-th hybercube, $\{v_0^{(i)}, v_1^{(i)}, ..., v_{2^{k-1}-1}^{(i)}\}$. For each i ($1 \leq i \leq h-1$), each of the second half nodes in $V^{(i)}$ is connected to one of the first half nodes in $V^{(i+1)}$. For the first and last hybercube, each of the first half nodes in $V^{(1)}$ is connected to one of the second half nodes in $V^{(h)}$. By doing so, the set of hypercubes are connected to each other, and each node in the graph has exactly k neighbors. Note that we employ a hypercube network to construct a k neighbor graph, and henceafter the nodes in the graph have a flat relationship, i.e., we distinguish neither to which hypercube each node belongs nor if a link connects to the different hypercubes. A unique identifier is given to each node, e.g., v_u for node $v_j^{(i)}$ where $u = i \cdot 2^{k-1} + j$ ($1 \leq i \leq h$ and $0 \leq j \leq 2^{k-1} - 1$).

Algorithm 2 ReplyToReader(r_r)

```
/* On receiving a query, tag t does following. */
Generates r_t.
for j from 1 to m do
    /* Note that the base rn_0 is empty. */
    β_j.hash ← H(gk_j||rn_{j−1}||r_t||r_r)
    β_j.num ← E(gk_j, rn_j)
    Add β_j to β.
γ ← ID_t ⊕ F(0||sk_t||rn_m||r_t||r_r)
Tag t replies with α, β, γ, and r_t.
```

The proposed structure achieves our design goals. Since there is no hierarchy in the structure, the relation among nodes is flat. The number of group keys m that each tag stores is independent from the number of tags n in the system. The value of m can be flexibly tuned as long as $m \leq n_g$.

Example of k-neighbor graph Figure 1 shows a 4-neighbor graph, which is created by connecting three 3-D hypercubes with each being colored by white, light gray, and dark gray, denoted by $V^{(1)}$, $V^{(2)}$, $V^{(3)}$, respectively. The first half of the nodes in a 3-D hypercube includes 000, 001, 010, and 011; the second half nodes are 100, 101, 110, and 111. Thus, each of the first half nodes in $V^{(j)}$ ($j = \{1, 2, 3\}$) and one of the second half nodes in $V^{(i+1)}$ ($i = \{2, 3, 1\}$) are connected to each other. As a result, each node has exactly 4 neighbors.

D. Key Issuing

A pair of a unique key and a group key are randomly generated and assigned to each node in the graph. Node v_i in the graph has two variables to store these keys, denoted by $v_i.sk$ and $v_i.gk$, respectively. In addition, v_i stores the ID of the tag t as $v_i.id$. Each tag t is randomly assigned to one node in the graph. Note that no more than one tag is assigned to the same node. Let v_i be the current node at which the pointer is located. At the beginning, v_i indicates to which tag t is assigned. By a random walk over the graph, tag t obtains a unique key sk, a set of group keys GK_t, a set of random numbers RN_t, and the entry pointer α to the graph. First, tag t obtains the secret key stored at $v_i.sk$. Let $N(v_i)$ be an open neighbor set of node v_i. The pointer randomly moves to one of the k nodes in $N(v_i)$.

For each level j from m to 1, the key issuer repeats the following. A random number $rn_j \in [0, n_g - 1]$ for the j-th level is generated. The pointer jumps to the corresponding node, say v_u, such that $u = (i + rn_j) \mod n_g$, where i is the current node ID. Tag t obtains group key gk_j from $v_u.gk$.

Each of the rn_j and gk_j is then added to the head of RN_t and GK_t, respectively. Assume that the pointer is at v_i after this process. One of the nodes in $N(v_i)$ is randomly selected, and the node ID is saved to α. The value of α is used as the entry point of the graph in the authentication phase.

At the end of this process, tag t obtains one unique key sk_t, m group keys GK_t, a set of m random numbers RN_t, and the entry point α. The pseudo code of the key issuing is provided in Algorithm 1.

Example of Key Issuing Figure 3 shows an example of the key issuing phase of KNGA, where $k = 2$, $m = 2$, and $n_g = 8$. Tag t is located at v_3 and let $sk_t \leftarrow v_3.sk$ (i.e., sk_3). Then, v_2 is randomly selected out of $N(v_3) = \{v_2, v_4\}$, and the pointer moves to v_2. A jump number $rn_2 = 6$ is randomly generated, and the pointer moves to v_0 as $(2 + rn_2) \mod n_g = 0$, where 2 is the v_2's ID. Tag t obtains $v_0.gk$ (i.e., gk_0) as one of the group keys. Random number 6 and group key gk_0 are added to $RN_t = \{6\}$ and $GK_t = \{gk_0\}$, respectively. Similarly, the pointer moves to either v_1 or v_7, both of which are in $N(v_0)$. Assume that v_1 is selected and the pointer moves to v_1. Again, a random number is generated, say $rn_1 = 4$. The pointer moves to v_5, since $(1 + rn_1) \mod n_g = 5$. Random number 4 and group key gk_5 are added to the head of the lists, i.e., $RN_t = \{4, 6\}$ and $GK_t = \{gk_5, gk_0\}$. Assume that v_6 is selected from $N(v_5)$. Finally, the pointer randomly moves to node $v_6 \in N(v_5)$, and $\alpha = v_6$ is used as the entry point of tag t to this key structure.

E. Mutual Authentication

After key initialization, the reader can securely communicate with tags, and they can mutually authenticate each other. In the proposed protocol, the reader first sends a query with nonce r_r, and then, a tag computes an encrypted reply using randomly generated nonce r_t. On receiving the tag's reply, the reader searches the tag by a random walk over the key structure. Finally, the reader sends back a verification code so that the tag also authenticates the reader.

Assume that tag t has the unique key sk_t, a set of group keys $GK_t = \{gk_1, gk_2, ..., gk_m\}$, a set of random numbers $RN_t = \{rn_1, rn_2, .., rn_m\}$, and the entry point α. On receiving a query with r_r from the reader, tag t randomly generates nonce r_t. Let $\beta = \{\beta_1, \beta_2, ..., \beta_m\}$, where each β_j consists of $\beta_j.hash$ and $\beta_j.num$. Each β_j is computed using group key gk_i and random number rn_i for $1 \leq i \leq m$ as follows. The value of $\beta_j.hash$ is computed by $H(gk_j||rn_{j-1}||r_t||r_r)$, where the base rn_0 is empty. On the other hand, $\beta_j.num$ is computed by $E(gk_j, rn_j)$. The value of γ is obtained using the secret key sk_t by $ID_t \oplus F(0||sk_t||rn_m||r_t||r_r)$, where ID_t is the tag t's ID. Note that 0 is concatenated at the beginning of the input for the mutual authentication. Then, the tag replies with α, β, γ, and r_t to the reader. The pseudo code of how a tag generates a reply is given in Algorithm 2.

On receiving the tag's reply, the reader moves the pointer to the node indicated by α, say v_i. For each β_j, the reader repeats the following. Since the balancing factor is k, each node has exactly k neighbors, and one of them has the group keys gk_j corresponding to $\beta_j.hashs$. The reader scans all the nodes in $N(v_i)$ and finds the corresponding node, say v_u, by validating if $H(v_u.gk||rn_{j-1}||r_t||r_r)$ equals to $\beta_j.hash$. Also, the reader decrypts $\beta_j.num$ using $D(v_u.gk, \beta_j.num)$ to obtain rn_j. The pointer moves to the node v_i where $i = (u - rn_j) \mod n_g$. Assume the pointer is located at v_i when the reader finds the group keys of β_j for all $1 \leq j \leq m$. One of the nodes in $N(v_i)$ has the corresponding unique key to γ, and thus, the reader searches v_u such that $\gamma \oplus F(0||v_u.sk||r_t||r_r)$ equals to $v_u.id$. By identifying the corresponding node v_u, the reader authenticates tag t. The pseudo code of the tag singulation process is presented in Algorithm 3.

Algorithm 3 Authentication(α, β, γ, r_t)

```
v_i ← α.
for j from 1 to m do
    for each v_u ∈ N(v_i) do
        /* Note that the base rn_0 is empty. */
        if H(v_u.gk||rn_{j-1}||r_t||r_r) = β_j.hash then
            rn_j ← D(v_u.gk, β_j.num)
            i ← (u − rn_j) mod n_g /* The pointer moves */
            j ← j + 1
    if The corresponding group key is not found then
        Reader R rejects tag t.
for each v_u ∈ N(v_i) do
    if v_i.id = γ ⊕ F(0||v_i.sk||rn_m||r_t||r_r) then
        Tag t is identified by v_i.id and v_i.sk.
        Reader R accepts tag t and runs Algorithm 4.
    if The corresponding unique key is not found then
        Reader R rejects tag t.
```

Algorithm 4 MutualAuthentication($\mathcal{R}$,t)

```
/* Reader R does following at the end of Algorithm 3. */
/* v_i is the node that tag t is located. */
π ← v_i.id ⊕ F(1||v_i.sk||r_t||r_r)
R sends π to tag t.
/* On receiving π, tag t does following */
ID'_t ← π ⊕ F(1||sk_t||r_t||r_r)
if ID'_t = ID_t then Tag t accepts reader R.
else Tag t rejects reader R.
```

In KNGA, the tag can also authenticate the reader. Assume that the reader identifies tag t located at v_i in the authentication phase. The reader generates a verification code, denoted by π, by computing $\pi = v_i.id \oplus F(1||v_i.sk||r_t||r_r)$, and then forwards it to tag t. Note that 1 is concatenated in the input. When tag t receives π from the reader, it obtains ID' by taking $\pi \oplus F(1||sk_t||r_t||r_r)$. If the reader has valid ID_t and sk_t, the resulting value ID' should equal ID_t. Therefore, tag t also authenticates the reader at the end of this process. The pseudo code of the tag singulation process is provided in Algorithm 4.

Example of Authentication An example of the mutual authentication is shown in Figure 4. Assume that the key issuing is complete as shown in Figure 3. The reply from tag consists of

$$\alpha = v_6 \quad (4)$$

$$\beta_1 = H(gk_5||r_t||r_r), E(gk_5, 4) \quad (5)$$

$$\beta_2 = H(gk_0||r_t||r_r), E(gk_0, 6) \quad (6)$$

$$\gamma = ID_t \oplus F(0||sk_t||3||r_t||r_r). \quad (7)$$

Algorithm 5 KeyUpdate($\mathcal{T}$, $\{v_0, v_i, ..., v_{n_g}\}$)

1: /* Key update at the reader's side */
2: **for** i from 1 to n_g **do**
3: $v_i.old\_gk \leftarrow v_i.gk$
4: $v_i.old\_gk \leftarrow H(r, v_i.gk)$, where r is a random number.
5: /* Upon singulation of tag t */
6: Allocate t to v_i associated with no other tag.
7: $v_i.sk \leftarrow H(r, v_i.sk)$, where r is a random number.
8: Run Algorithm 1 (Lines from 5 to 15).

Starting from v_6, the reader finds the corresponding node v_5 to $\beta_1.hash$ and moves the pointer to v_5. Then, the pointer moves toward the left by 4, which information is obtained by decrypting $\beta_1.num$. This process is applied to β_2 and γ. After identifying tag t, the reader sends $\pi = ID_t \oplus F(1||v_3.sk||r_t||r_r)$. On receiving π, tag t computes $\pi \oplus F(1||sk_t||r_t||r_r)$, which results in ID_t. By doing this, tag t also authenticates the reader.

F. Key Updating

Our approach is the mix of the key updating mechanisms in SPA [5] and RSLA [10]. Group keys shared among several tags, and therefore, the reader must keep old group keys until all the tags associated with these old keys update their keys. Thus, the proposed key updating first renews all the group keys in the k-neighbor graph, and the keys at tags' side are updated when they are singulated. On the other hand, unique keys are never shared among tags. This implies that the reader does not need to maintain old unique keys. Therefore, the unique key at a tag can be updated when the reader interrogates the tag, and the old unique key can be immediately discarded after key updating.

The proposed key updating works as follows. A new key at each node v_i in the k-neighbor graph is computed using a one-way hash function, and the old group keys are kept in $v_i.old\_gk$. That is, for $1 \leq i \leq n_g$, $v_i.old\_gk \leftarrow v_i.gk$ and $v_i.gk \leftarrow H(r, v_i.gk)$, where r is a nonce. The value of r is kept in secret, and so adversaries cannot obtain a new key from the current key. When the reader singulates tag t, the group keys are updated. When the old keys become unnecessary, they are removed from the key structure. At the same time, the unique key of tag t is renewed by $v_i.sk \leftarrow H(r, v_i.gk)$, where r is a nonce. Both the old unique key and random number can be immediately discarded upon key updating. With the new keys, a unique key, a set of group keys, a set of random numbers, and the entry point are assigned to tag t as shown from Lines 5 to 15 in Algorithm 1.

G. System Maintenance

An RFID system should adapt to dynamic environments, where new tags join to and some old tags leave from the system. When a tag leaves the system, no modification at the key structure is required. When a new tag joins the system, the tag is randomly assigned to one node, which is not associated with any other tag, in the k-neighbor graph. Then, the keys are issued to the new tag using Algorithm 1. If no such a node is available, a new k-neighbor graph with the same size as the original one is created, and the new tag is randomly assigned to a node in the new graph. In this case, there will be more than one set of k-neighbor graphs for the key structure. This causes the reader to scan $2k$ nodes at the first component of tag's reply, i.e., β_1, in the authentication phase. Since the additional computation cost for the reader in the singulation process is up to k, the authentication efficiency remains the same as the original KNGA in the asymptotic order. In addition, by creating a new k-neighbor graph, the system can accommodate twice the number of tags as the original. We believe the number of tags rarely increases more than twice from the original number of tags during online process, which is a relatively short period. On the other hand, in a long-term period, the number of tags could grow more than twice, but this can be handled by reconstructing the whole key structure off-line.

H. Considerations on Parameter Selection

The system parameters in this paper include the balancing factor k, the number of tags n, the number of group keys n_g, and the number of group keys m that each tag stores. How these parameters are set plays a critical role for both security and performance. As a rule of thumb, the greater values of n_g and m are, the higher degree of privacy can be achieved. In the performance aspect, the values of n and m determine the authentication speed. Here, the number of tags n is provided by an application and is not tunable. As $n \leq n_g = k^{m+1}$ must hold, the values of m and k dominate the system parameters. Note that the value of k does not affect the performance in the asymptotic order. Since our primary goal is to improve the privacy degree, the value of m should be maximized as long as the tag's memory is dedicated to the security mechanism.

While the value of m is limited due to the low cost design of passive tags, we claim that KNGA is still scalable. In general, the key length in passive tags is set to be 32-bit. A random number for shifting as well as an entry point can be encoded into a 32-bit string as long as $n_g \leq 2^{32}$. To be specific, when $m = 7$ and $k = 16$, a memory space of 512 bits is required to store one unique, 7 group keys, 7 random numbers, and one entry point. Therefore, up to 2^{32} tags can be supported in such an RFID system, which is sufficiently large enough for most of the real world RFID applications.

VI. Quantitative Security Analysis

A. The Correlation Probability Analysis

We first prove that the proposed KNGA protocol achieves the performance bound of the weak privacy.

Lemma 1 *Given compromised tag t and uncompromised tag t', they are indistinguishable as long as $GK_t \neq GK_{t'}$, where $GK_t = \{gk_1, gk_2, ..., gk_m\}$ and $GK_{t'} = \{gk'_1, gk'_2, ..., gk'_m\}$.*

Proof: The proof is trivial, and thus is omitted. ∎

Lemma 2 *For given n_g and n_t, The correlation probability of KNGA is $\prod_{i=1}^{m} \frac{m-i+1}{n_g-i+1}$.*

Proof: Let $GK_t = \{gk_1, gk_2, ..., gk_m\}$ and $GK_{t'} = \{gk'_1, gk'_2, ..., gk'_m\}$ be the set of group keys that tag t and t'

store, respectively. According to Lemma 1, t and t' are never linked unless $GK_t = GK_{t'}$. We have $\Pr[\exists gk_i \in GK_{t'} | gk_1 = gk_i] = \frac{m}{n_g}$, $\Pr[\exists gk_i \in \{GK_{t'} - gk_1\} | gk_2 = gk_i] = \frac{m-1}{n_g-1}$, and so on. Thus, the correlation probability of KNGA is obtained by

$$Pr[GK_t = GK_{t'} | KNGA] = \prod_{i=1}^{m} \frac{m-i+1}{n_g-i+1}, \quad (8)$$

This concludes the proof. ∎

We reach the following conclusion.

Theorem 3 *The proposed KNGA achieves the performance bound of the weak privacy.*

Proof: From Lemma 2, KNGA has the same correlation probability as the performance bound of the weak privacy shown in Equation 1. Therefore, the above claim is true. This concludes the proof. ∎

B. Extended Anonymity

When some tags are physically compromised, the degree of privacy of the other tags may decrease. To quantify this, anonymity is widely used as a privacy metric against the compromise attack. In this paper, we extend the definition of anonymity in [9], [10] by excluding the anonymity of compromised tags. Note that the anonymity of compromised tags is always 0, as they are identified with 100% probability. By doing this, our anonymity definition clarifies the difference between the strong and weak privacy models. That is, no matter what portion of tags is compromised, the degree of anonymity, denoted by A, always equals to 1 in the strong privacy model. In the weak privacy model, the degree of anonymity of uncompromised tag ranges $0 \leq A \leq 1$. We define an extended anonymity as follows.

$$A = \frac{1}{n - n_c} \sum_i |S_i| \cdot \frac{|S_i|}{n - n_c} \quad (9)$$

Here, S_i is an anonymous set in which subjects are distinguishable from each other. When no tag is compromised, all the tags belong to the same anonymous set, say S_1. Hence, $|S_1| = n$ and $A = 1$ hold. If some tags are compromised, the tags are divided into disjoint sets with each $|S_i|$ being smaller than $n - n_c$, where n_c is the number of compromised tags

Equation 9 is conceptually defined by the observation as follows. The degree of ambiguity of an anonymous set, say S_i, with respect to the number of uncompromied tags in the system is formulated by $\frac{|S_i|}{n-n_c}$. Since there are $|S_i|$ tags in the set, the degree of ambiguity can be weighted by multiplying $|S_i|$. After computing the summation of the ambiguity of each anonymous set, the average anonymity of each uncompromised tag is computed by multiplying $\frac{1}{n-n_c}$.

C. Anonymity Analysis

We formulate the expected anonymity when c ($0 \leq n_c \leq n-1$) tags are compromised. Let $\mathcal{T}_c$ (where $|\mathcal{T}_c| = n_c$) be a set of compromised tags, and c be the average number of compromised tags located at neighbors of each node. Note that the average value of c is obtained by $\frac{kn_c}{n}$, because the compromised tags are randomly selected, and such a distribution is known to follow the Binomial distribution. We define $s(t)$ as a mapping function from tag t to its anonymous set S_i, i.e., $s : t \rightarrow \mathcal{P}(\mathcal{T} - \mathcal{T}_c) \cup \{t\}$, where $\mathcal{P}(.)$ is the power set. The anonymous set size of $s(t')$ will be either one of the following three cases.

Case 1: If $\forall t \in \mathcal{T}_c$, $GK_t \neq GK_{t'}$, then $|s(t')| = n - n_c$.

Case 2: If $\exists t \in \mathcal{T}_c$, s.t. $GK_t = GK_{t'}$ and $rn_m = rn'_m$, then $|s(t')| = k - c$.

Case 3: If $\exists t \in \mathcal{T}_c$, s.t. $GK_t = GK_{t'}$ and $rn_m \neq rn'_m$, then $|s(t')| = n - k - n_c + c$.

When an authentication protocol achieves strong privacy, two tags are never linked, i.e., $GK_t \neq GK_{t'}$ always holds. Thus, cases 2 and 3 never occur, and the resulting anonymity equals to 1, no matter what the value of n_c is. In the weak privacy model, the anonymity can be smaller than 1, since compromising some tags results in smaller anonymous set size of the uncompromised tags as indicated by cases 2 and 3. Different protocols have different $\Pr[GK_t = GK_{t'}]$. The proposed KNGA keeps maximal achievable anonymity by minimizing the probability of $GK_t = GK_{t'}$.

We derive the expected average anonymity of uncompromised tags as follows. Let us denote $\rho = \prod_i^m \frac{m-i+1}{n_g-i+1}$ by the lower bound of the correlation probability in Equation 1. The probabilities of cases 1, 2, and 3 occurring are given by

$$p_1 = \Pr[Case\ 1] = (1-\rho)^{n_c} \quad (10)$$

$$p_2 = \Pr[Case\ 2] = 1 - \left(1 - \frac{\rho}{n_g}\right)^{n_c} \quad (11)$$

$$p_3 = \Pr[Case\ 3] = 1 - \left(1 - \frac{\rho(n_g-1)}{n_g}\right)^{n_c}. \quad (12)$$

Therefore, we can derive the expected average anonymity of uncompromised tags by

$$E[A] = p_1(n-n_c) + p_2 k\left(1 - \frac{n_c}{n}\right) + p_3\left(n - k - n_c + k\frac{n_c}{n}\right). \quad (13)$$

Equation 13 can be simplified as follows. Since ρ is a very small value, $\rho/n_g \approx 0$ for a large n_g. This makes $p_2 \approx 0$ in Equation 11. Also, $(n_g - 1)/n_g \approx 1$ for a large n_g, and thus, we can say $p'_3 \approx 1 - (1-\rho)^{n_c}$ in Equation 12. Therefore, the expected anonymity can be approximated by

$$E[A] \approx p_1(n-n_c) + p'_3\left(n - k - n_c + k\frac{n_c}{n}\right). \quad (14)$$

VII. Performance Evaluation

The performance of the proposed KNGA is evaluated using simulations by comparing with the tree-based [4], AnonPri [9], RSLA [10], and RSGA [11].

A. Simulation Configurations

In the simulations, the RFID system is composed of one RF reader and a number of RF tags ranging from 2^8 to 2^{14}. During an interrogation process, randomly selected n_c ($0 \leq n_c \leq 2048$) tags are considered as being compromised.

The parameters for each of the protocols are set to be as follows. For the tree-based, RSLA, RSGA, and KNGA, the balancing factor k is set to be either 2, 4, 8, or 16. For fair

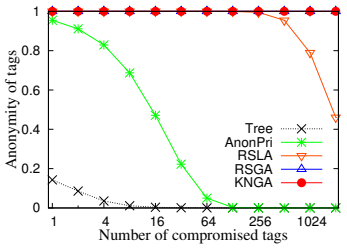

Fig. 5. Anonymity with $k = 2$.

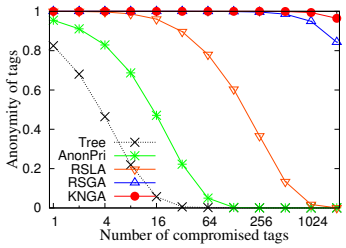

Fig. 6. Anonymity with $k = 16$.

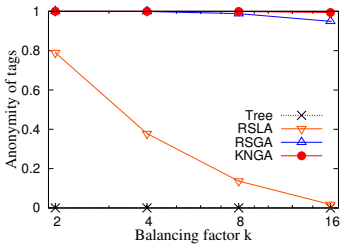

Fig. 7. Anonymity with $n_c = 1024$.

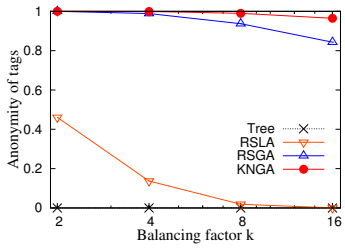

Fig. 8. Anonymity with $n_c = 2048$.

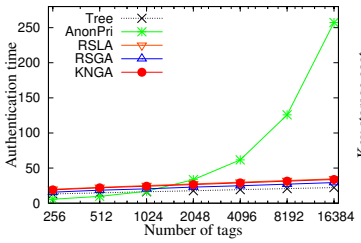

Fig. 9. The authentication time.

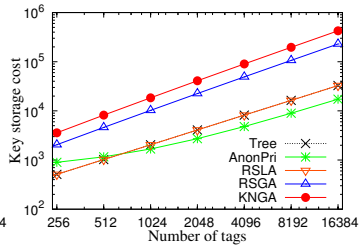

Fig. 10. The key storage cost.

comparison, the number of group keys at each node is set to be $m = \lceil \log_k n \rceil - 1$. In AnonPri, tags are divided into disjoint groups with size of 64. The number of pseudo ID pool and the number of keys that each tag stores are set to be 1000 and 10, respectively.

Anonymity defined in Section VI-B is used as the privacy metric. Randomly selected n_c tags are compromised, and then, the anonymity of the other tags is computed. As the performance metrics, the authentication speed is employed, which is defined as the number of computations of the hash and encryption functions. The number of group and unique keys at the server side are counted as the key storage cost. For each configuration, 1000 simulation experiments are conducted.

B. Simulation Results

Figures 5 and 6 show the anonymity for different protocols with respect to the number of compromised tags, where the number of nodes is 4096 and the balancing factor is $k = 2$ for Figure 5 and $k = 16$ for Figure 6. As can be seen in Figure 5, the introduction of random walks (i.e., RSLA, RSGA, and KNGA) outperforms the other protocols. The anonymity of RSLA significantly drops at $n_c = 1024$ (12.5% of tags are compromised), and that of RSGA significantly drops at $n_c = 2048$ (50% of tags are compromised). On the other hand, the proposed KNGA still keeps high anonymity even in the case that a large portion of tags are compromised. This trend becomes clearer when $k = 16$ as shown in Figure 6. The number of group keys at each tag is $m = \lceil \log_{16} 4096 \rceil - 1 = 2$. Thus, the probability of two tags being correlated is much higher than the case of $k = 2$. Nevertheless, our KNGA maintains high anonymity even when $n_c = 2048$.

Figures 7 and 8 illustrate the anonymity for different protocols with respect to the balancing factor k. In these figures, the number of nodes is set to be 4096 and the number of compromised nodes is $n_c = 1024$ for Figure 7 and $n_c = 2048$ for Figure 8. As a rule of thumb, each tag stores fewer group keys when the balancing factor k increases. This is because the system can support up to k^{m+1} nodes for the given number of group keys m and the balancing factor k. Therefore, the probability of two tags being correlated increases, as the value of k increases. In Figure 7, where $n_c = 1024$, KNGA and RSGA maintain high anonymity for $k = 16$. On the other hand, as shown in Figure 8, where $n_c = 2048$, the anonymity of KNGA is higher than that of RSGA by 10% for $k = 16$.

Figure 9 presents the authentication time for different protocols with respect to the number of tags in the system. The authentication by AnonPri takes a longer time, as the number of tags increases. Since the other protocols run in logarithmic order, the authentication speed is relatively stable. Compared with the original tree-based protocol, RSLA, RSGA, and KNGA slightly incur more computation cost. This is because decryption functions must be applied to a tag's reply for random walks.

Figure 10 provides the key storage cost at the server side for different protocols with respect to the number of tags in the system. For the tree-based, AnonPri, and RSLA, the key storage cost is dominated by the number of tags in the system. The key structure of both RSGA and KNGA have $n(\lceil \log_k n \rceil - 1)$ internal nodes, and thus, they incur more storage cost. In addition, each node in the k-neighbor graph of KNGA must store a dummy unique key even when no tag is assigned to it. This is why KNGA incurs more storage cost. However, we stress that the key storage cost at server side is not a significant issue, and the storage cost at tags' side in KNGA remains the same as the existing solutions.

VIII. Conclusions

Although the existing structured key-based private protocols for quickly and securely interrogating tags in large-scale RFID systems are known to achieve weak privacy, the theoretical bound is still unknown. In this paper, we first derive the lower bound of the correlation probability with respect to the number of group keys in the system. Then, we propose the k-neighbor graph-based authentication (KNGA) protocol, where the reader singulates individual tags by randomly walking over a k-neighbor graph. Our analysis shows that the proposed KNGA achieves the performance bound of structured key-based authentication, and the extended anonymity is quantified as the degree of privacy of uncompromised tags. The simulation results demonstrate that the proposed KNGA outperforms the existing solutions.

Acknowledgments

This research has been funded in part by the JSPS KAKENHI Grants JP17K12675 and by the U.S. National Science Foundation grants IIS-1618669 (III) and ACI-1642133 (CICI).

REFERENCES

[1] S. Qi, Y. Zheng, X. Chen, J. Ma, and Y. Qi, "Double-Edged Sword: Incentivized Verifiable Product Path Query for RFID-Enabled Supply Chain," in *ICDCS*, 2017, pp. 414–424.

[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*, 2008, pp. 479–490.

[3] Y. Li, R. H. Deng, J. Lai, and C. Ma, "On Two RFID Privacy Notions and Their Relations," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 30, 2011.

[4] D. Molnar and D. Wagner, "Privacy and Security in Library RFID Issues, Practices, and Architectures," in *CCS*, 2004, pp. 210–219.

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

[6] L. Lu, J. Han, R. Xiao, and Y. Liu, "ACTION: Breaking the Privacy Barrier for RFID Systems," in *INFOCOM*, 2009, pp. 1951–1961.

[7] T. Li, W. Luo, Z. Mo, and S. Chen, "Privacy-Preserving RFID Authentication based on Cryptographical Encoding," in *INFOCOM*, 2012, pp. 2174–2182.

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

[9] M. E. Hoque, F. Rahman, and S. I. Ahamed, "AnonPri: An Efficient Anonymous Private Authentication Protocol," in *PerCom*, 2011, pp. 102–110.

[10] M.-T. Sun, K. Sakai, W.-S. Ku, T. H. Lai, and A. V. Vasilakos, "Private and Secure Tag Access for Large-Scale RFID Systems," *IEEE Trans. Dependable Secur. Comput.*, vol. 13, no. 6, pp. 657–671, 2016.

[11] Y. Komori, K. Sakai, and S. Fukumoto, "A Fast and Secure Tag Authentication in Large-Scale RFID Systems Using Skip Graphs," *Comput. Commun.*, vol. 116, pp. 77–89, 2018.

[12] A. Juels, "Minimalist Cryptography for Low-Cost RFID Tags," in *SCN*, 2004, pp. 149–164.

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

[14] Q. Yao, Q. Qi, J. Han, J. Z. X. Li, and Y. Liu, "Randomized RFID Private Authentication," in *PerCom*, 2009, pp. 1–10.

[15] L. Lu, Y. Liu, and X.-Y. Li, "Refresh: Weak Privacy Model for RFID Systems," in *INFOCOM*, 2010, pp. 1–9.

[16] M. Chen and S. Chen, "ETAP: Enable Lightweight Anonymous RFID Authentication with O(1) Overhead," in *ICNP*, 2015, pp. 267–278.

[17] B. Alomair, A. Clark, J. Cuellar, and R. Poovendran, "Scalable RFID Systems: A Privacy-Preserving Protocol with Constant-Time Identification," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1536–1550, 2012.

[18] Y. Komori, K. Sakai, and S. Fukumoto, "Randomized Skip Graph-Based Authentication for Large-Scale RFID Systems," in *WASA*, vol. 9798, 2016, pp. 1–12.

[19] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Chapman and Hall/CRC, 2008.