The following publication X. Liu, S. Zhang, B. Xiao and K. Bu, "Flexible and Time-Efficient Tag Scanning with Handheld Readers," in IEEE Transactions on Mobile Computing, vol. 15, no. 4, pp. 840-852, 1 April 2016 is available at https://doi.org/10.1109/TMC.2014.2381252.

Flexible and Time-Efficient Tag Scanning with Handheld Readers

Xuan Liu, Shigeng Zhang, *Member, IEEE,* Bin Xiao, *Senior Member, IEEE,* and Kai Bu, *Member, IEEE*

Abstract—Tag scanning is an important issue to dynamically manage tag IDs in Radio Frequency IDentification (RFID) systems. Different from tag identification that collects IDs of all the tags, tag scanning first verifies whether or not a responding tag has already been identified and retrieves its ID when the answer is yes, and collects the tag's ID only when it is unidentified. In this paper, we present the first study on *spot scanning* with a handheld reader, which aims to scan tags in the reader's interrogation range at an arbitrarily specified position in the system. Existing studies mainly focus on *continuous scanning*, and they are highly time inefficient in performing spot scanning. The inefficiency stems from the small overlap between tag populations in different spot scanning operations, in which case existing solutions cannot efficiently recognize unidentified tags. We develop a novel technique called LOCK to efficiently recognize unidentified tags even when the overlapped tags are few. LOCK does not simply use a tag's reply slot index but also compact short responses from tags to efficiently distinguish unidentified tags from identified ones. The valuable compact short responses are firstly investigated, which are the keys for efficient tag identification in the paper. Based on LOCK, three tag scanning protocols are proposed to solve the spot scanning problem. Simulation results show that, for spot scanning, our best protocol reduces per tag scanning time by up to 70% when compared with the state-of-the-art solution. Moreover, the proposed protocols can also be employed to perform continuous scanning with better time efficiency than the best existing solutions.

Index Terms—RFID; spot scanning; handheld reader; time-efficiency; tag scanning

1 Introduction

The Radio Frequency IDentification (RFID) technology brings revolutionary changes to many industry fields and improves their management efficiency, e.g., warehouse management, inventory control, and logistics [1], [2]. In practical applications, RFID tags are attached to products to store the products' information and trace their state changing history. The information stored in tags (e.g., tag IDs) are retrieved using RFID readers in a wireless manner, without needing to move each individual object close to the reader as in traditional barcode systems. The process of collecting tag IDs is usually called tag identification [3], [4]. After the tag IDs are collected, they are stored in a back end server for further processing such as profit statistics and data mining. Tag identification is one of the most frequently executed tasks in RFID systems, and thus it should promise high time efficiency [5]–[9].

Different from tag identification that collects IDs of all the tags, *tag scanning* [10], [11] first verifies whether or not a responding tag has been identified and recorded in the system, and collects its ID only when the tag is unidentified. Tag scanning is helpful to efficient and timely inventory management and tag population monitoring. For example, consider periodical tag inventory in a dynamic RFID system where tags may enter and leave the system frequently. As only a small part of tags enter or leave the system during two consecutive inventory operations, it is highly time inefficient to collect IDs of all the tags in every inventory operation. Only IDs of unidentified tags need to be collected. For the majority tags that have been identified in previous inventory operations, we only need to verify whether or not they are still present in the system.

Existing studies mainly focus on *continuous scanning* [6], [7], [10], [11], which uses a series of scanning operations cooperatively performed at different positions to scan all the tags in the system. Different from existing studies, we present the first investigation on time-efficient *spot scanning* with a handheld reader, i.e., scanning tags in the reader's interrogation range at an arbitrarily specified position in a large RFID system. Spot scanning provides a flexible way to make customized tag inventory for different purposes. For example, consider a salesman who is in charge of managing products of a specified category. As products in the same category are usually placed at adjacent positions, the salesman can make inventory of the products by executing a spot scanning at the position where the products are placed. There is no need to scan other tags in the system as done in the continuous scanning. The difference between continuous scanning and spot scanning is that there are many overlapped tags between adjacent scanning operations in continuous scanning, but there are only few or even none overlapped tags between different spot scanning operations, as spot scanning can be executed at arbitrary positions in the system. This greatly degrades the performance of existing continuous scanning protocols when they are applied to spot scanning.

- *Xuan Liu and Bin Xiao are with the Department of Computing, the Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. E-mail {csxuanliu,csbxiao}@comp.polyu.edu.hk.*
- *Shigeng Zhang is with the School of Information Science and Engineering, Central South University, Changsha, China. E-mail: sgzhang@csu.edu.cn.*
- *Kai Bu is with College of Computer Science and Technology, Zhejiang University, China. E-mail: kaibu@zju.edu.cn.*

Existing solutions to continuous scanning are time inefficient in solving the spot scanning problem. These solutions commonly first distinguish unidentified tags from identified ones, and collect IDs of only unidentified tags to improve time efficiency. Distinguishing unidentified tags and identified tags incurs additional time. In case of continuous scanning in which there are many overlapped tags, this additional time can be offset by the time saved in not recollecting IDs of overlapped tags. However, in case of spot scanning, there are only few or even none overlapped tags, and thus the time spent in recognizing unidentified tags will overwhelm the time saved in not recollecting IDs of overlapped tags [11]. Existing solutions to continuous scanning [10]–[14] thus perform poorly in solving the spot scanning problem. Moreover, they might perform even worse than the baseline approach that directly collects IDs of all the tags in the reader's interrogation range.

We develop a novel technique called LOCation Key matching, namely LOCK, to efficiently distinguish unidentified tags from identified ones in spot scanning. LOCK combines a tag's reply slot index (*location*) and a compact short response (*key*) to quickly verify whether or not a responding tag has already been identified. With LOCK, the reader can quickly confirm the existence of identified tags and collect IDs of only unidentified tags. Based on LOCK, three protocols are proposed to solve the spot scanning problem. These protocols achieve high time efficiency even when there are few overlapped tags between different scanning operations, and thus are suitable to solve the spot scanning problem.

To summarize, we make the following main contributions in this paper:

- A novel technique called LOCK that can quickly distinguish unidentified tags from identified ones in spot scanning scenarios. LOCK helps build efficient spot scanning protocols due to its high time efficiency even when there are only a few overlapped tags between different scanning operations.
- Three LOCK-based protocols to solve the spot scanning problem. These protocols can be used to perform spot scanning with much higher time efficiency than existing solutions. They can also improve time efficiency in performing continuous scanning.
- Extensive simulation experiments to evaluate the performance of the proposed protocols. Simulation results demonstrate that, compared with state-of-the-art solutions, our best protocol saves up to 70% per tag scanning time in spot scanning, and up to 38% total scanning time in continuous scanning.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 gives system model and problem statement. The detailed design of LOCK is described in Section 4. Based on LOCK, three tag scanning protocols are proposed in Section 5. Simulation results are presented in Section 6. Finally, Section 7 gives some conclusion remarks.

2 Related work

Tag identification: Tag identification protocols can be classified into two categories [3]: Aloha-based protocols [2], [5], [15]–[19] and tree-based protocols [8], [20], [21]. In Aloha-based protocols, the optimal frame size setting was investigated to enhance the time efficiency [15], [16] and energy efficiency [5] in tag identification. Different from them, the DDC protocol [17] used a novel random number (RN) pattern to arbitrate channel access contentions when multiple tags transmit simultaneously. With the new RN pattern, DDC can separate colliding RN signals and collect more than one tag IDs in a slot, and thus greatly improves identification throughput. The FACT protocol proposed in [18] exploited analog network coding to separate tag IDs from mixed signals received in collision slots, and thus improved tag identification throughput. Recently, compress sensing was also used to improve time efficiency in tag identification [2].

Tree-based tag identification protocols mainly focus on reducing collisions by designing smart tree traversing approaches. In [20] the authors proposed AQS and ABS, which respectively assign reply slots for tags adaptively according to information obtained in previous identification process in query tree protocol and binary tree protocol. The *tree hopping* (TH) protocol proposed in [8] first estimates the optimal level to start a query, which effectively reduces collision slots and thus improves time efficiency in tag identification. The QQTT protocol [21] estimates the number of colliding tags and uses the first colliding bit in IDs to separate colliding tags, and consequently improves identification efficiency.

Continuous scanning: Continuous scanning aims to scan all the tags in the system by performing a series of scanning operations cooperatively at different positions. In [10] the authors proposed a two phase solution called CU to perform continuous scanning. CU first distinguishes unidentified tags from identified tags, and then collects IDs of only unidentified tags. However, the performance of CU degrades when the overlapped tags between two adjacent scanning operations are few, as pointed out in [11]. The ACOS protocol proposed in [11] thus first estimates the number of overlapped tags between adjacent scanning operations and then chooses the most efficient way to scan tags, e.g., either use CU or directly collect IDs of all the tags in the reader's interrogation range. In [6], [7] the authors investigated how to move a handheld/mobile reader to implement continuous scanning with high time and energy efficiency. They mainly focused on planning the optimal trajectory for the handheld reader, but paid little attention to avoiding ID recollection of overlapped tags.

Several *unknown tag identification* protocols [12]–[14] were proposed in recent years. These protocols aim to collect IDs of unidentified tags and prohibit replies from identified tags to improve time efficiency. The FUTI protocol [14] exploited the high efficiency of the bit vector

TABLE 1
Main notations used in this paper.

Notation	Meaning
N	Total number of recorded tags
M	Number of the reader's local tags
t_{id}	Time to transmit a tag's ID
t_1	Time to transmit a one bit response
t_k	Time to transmit a key
t_b	Time of a search opeartion
t_l	Time to transmit a key plus the offset in segment
α	Collection probability of unidentified tags
η	The ratio of unidentified tags to all tags
l	Length of a tag's key
KS_j	The key set associated with the j-th slot
$k_{max}(\alpha, l)$	Maximum number of allowed KEYs in one slot to guarantee α when key length is l
$f_{min}(N, \alpha, l)$	Minimum frame length to guarantee α

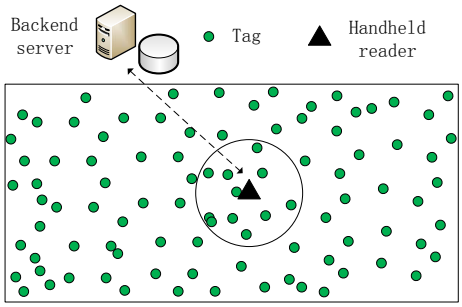

Fig. 1. Illustration of spot scanning in a large RFID system. The reader needs to scan tags in its interrogation range, but it needs not to scan other tags in the system.

technique [22] to quickly filter out identified tags and label unidentified tags. After being labelled, the IDs of unidentified tags are collected with Aloha-based identification protocols. The MUIP protocol [13] can guarantee collecting IDs of all the unidentified tags. However, these protocols achieve high time efficiency only when all the recorded tags are covered by the reader, while in spot scanning the reader could cover only a very small part of recorded tags. These protocols thus cannot promise high time efficiency in spot scanning. Moreover, they cannot tolerate missing tags and need to detect them with missing tag detection protocols [23]–[25] before starting the protocols. In contrast, the protocols proposed in this paper achieve high time efficiency in spot scanning and tolerate missing tags well.

Tag cardinality estimation: Rather than collecting tag IDs, there were some works on estimating the number of tags in the system [26]–[33]. The UPE algorithm proposed in [27] uses the ratio of empty or collision slots in the frame to estimate the total number of tags. Instead of using the ratio of empty slots, the FNZB algorithm [26] uses the index of the first non-empty slot in the frame to obtain an estimate of tag number, and the ART algorithm [30] uses the number of consecutive non-empty slots to perform estimation. In [28], the authors proposed a tree-based approach to obtaining the tag number estimation. In a recent work [31], the authors established the lower bounds on tag estimation overhead for both single set and multiple set estimation problems. Energy efficiency of tag estimation was also investigated in [29]. These algorithms can be used to obtain accurate estimation of the number of tags in the reader's interrogation range, and can help set the optimal frame size in our protocols.

3 System Model and Problem Statement

In this section, we describe the system model and define the spot scanning problem to be solved. The notations used in this paper are summarized in Table 1.

3.1 System Model

As shown in Fig. 1, we consider an RFID system that contains a large number of tags. A handheld reader is used to scan tags in its interrogation range at a specified position in the system. The reader covers only a small part of tags in the system, and these tags are referred to as the reader's *local tags*. There is a back end server that stores IDs of all the tags that have already been identified up to the latest tag scanning operation. These tags are called *recorded tags*. During two consecutive tag scanning operations, new tags may enter the system and some recorded tags may leave the system. The former tags are called *unidentified tags*, and the latter tags are called *missing tags*. The back end server has no knowledge of these unidentified tags and missing tags. It has to collect IDs of unidentified tags and determine which tags are missing. After each tag scanning operation, the back end server updates the list of recorded tags according to the scanning result.

The handheld reader can obtain the set of all the recorded tags up to the latest scanning operation by querying the back end server, but *it has no priori knowledge about which of these recorded tags are in its interrogation range*, as the reader can be at any position in the system. Note that existing solutions to collecting IDs of unidentified tags [10]–[14] all implicitly assume that the reader knows which recorded tags are in its interrogation range. We assume that the reader knows the rough number of its local tags, which could be easily obtained by using tag cardinality estimation algorithms [27], [30]–[32].

The communications between tags and the reader adopt the frame slotted Aloha protocol [34]. The communications consist of a series of *frames* that are divided into many synchronized *time slots*. At the beginning of each frame, the reader first broadcasts a query command containing the frame size f and a seed number r with which tags determine their reply slot indexes. After receiving the query command, a tag randomly selects a slot s in the range $[0, f-1]$ with a hash function H, i.e., $s = H(ID, r) \bmod f$ where ID is the tag's ID. Only when exactly one tag replies in a slot, the reader can successfully identify that tag's ID. Otherwise, when two or more tags select the same slot, a collision occurs and both tags cannot be correctly identified. When there are collisions, the reader starts a new frame to separate the

colliding tags and try to identify their IDs. This process is repeated until all the tags are identified.

When the tag replies to the reader, it can transmit either its tag ID or a short response. The time duration of a slot depends on how many tags reply in the slot and what information the tags transmit to the reader. A slot in which no tag replies to the reader is called an *empty slot*, and its duration is denoted as t_e. The time to transmit a tag ID is denoted as t_{id}.

3.2 Problem Statement

With the system model given in Section 3.1, we define the spot scanning problem as follows.

Definition 1 (Spot scanning): Given the set of all the recorded tags in the system, the reader executes a spot scanning at an arbitrarily specified position to: 1) determine which of its local tags have been identified and retrieve their IDs from the recorded tags list; and 2) collect IDs of unidentified tags in its interrogation range. The objective is to minimize the time spent in the tag scanning process, and meanwhile guarantee that the probability of unidentified tags being collected is no smaller than a pre-defined threshold α.

For example, imagine an RFID system containing 100,000 recorded tags. Assume that there are 500 local tags in the reader's interrogation range, among which 50 are unidentified tags and the other 450 are recorded tags. The reader can make inventory of these tags by collecting IDs of all the 500 tags. However, as most tags have already been identified, this simple approach is highly time efficient (the IDs of the 450 recorded tags are collected multiple times). On the other hand, the reader has no way to know which 450 of the 100,000 recorded tags are in its interrogation range, because the scanning position is arbitrarily specified. Moreover, it even cannot know the number of recorded tags in its range. It needs to verify identified tags and recognize unidentified tags, but what it has in hand is the only knowledge that *the recorded tags in its range must belong to the set of the 100,000 recorded tags in the system*. Determining a small set of target tags (450) from a very large set of candidate tags (100,000) is the intrinsic difficulty in solving the spot scanning problem.

It is meaningless to detect missing tags with the result of a single spot scanning due to two reasons. First, the reader does not know which recorded tags *should* be present in its interrogation range as the scanning position is arbitrarily specified. Second, the reader has no way to differentiate between a missing tag and a tag out of the reader's range but still in the system, because they act the same in the perspective of the reader. However, solutions to spot scanning should guarantee that the collection probability of unidentified tags is higher than the required threshold even when there are tags *missing from the system*. Moreover, solutions to spot scanning should be able to be leveraged to perform continuous scanning by move the handheld reader along a well planned trajectory [6], [7]. The protocols proposed in this paper can meet these requirements.

4 LOCK: The LOCation-Key Matching Mechanism

In this section, we first describe the design of LOCK in Section 4.1, then discuss how to set optimal parameters in Section 4.2, and finally discuss its limitations and improvement directions in Section 4.3.

4.1 Description of LOCK

LOCK consists of two phases. In the first phase, LOCK predicts the reply slots of the recorded tags and divides them into different subsets according to their reply slot indexes. Then for every slot, LOCK calculates its *associated key set* according to the recorded tags selecting this slot. In the second phase, LOCK starts a query frame and collects responses from the reader's *local tags*. According to the received response in a slot and the key set associated to that slot, LOCK differentiates unidentified tags from recorded tags. Recoded tags are simply confirmed; only unidentified tags' IDs are collected. We give the details of the two phases below.

In the first phase, LOCK calculates the expected reply slot for every recorded tag according to the tag's ID. Denote by TR the set of all the recorded tags in the system, where $TR = \{t_1, t_2, \ldots, t_N\}$. Given the frame size f and the random seed r, the reply slot index of t_i ($1 \leq i \leq N$) is calculated as $H(ID(t_i)||r) \bmod f$, where $ID(t_i)$ represents t_i's ID. Note that because the reader knows all the recorded tags, it can predict the reply slot index for every recorded tag.

The reader then divides the recorded tags into different subsets according to their reply slot indexes, and calculates the associated key set for every slot. A tag's key is an l-bits random number generated based on that tag's ID. For example, t_i's key is generated as $Key(t_i) = H_{key}(ID(t_i)) \bmod L$, where $L = 2^l$ is the total number of possible keys, and H_{key} is a uniform hash function known by both the reader and tags. All the recorded tags that select the j-th slot are grouped into the same subset TR_j, i.e., $TR_j = \{t_i | H(ID(t_i)||r) \bmod f = j, 1 \leq i \leq N\}$. For every tag in TR_j, the reader calculates its key and constructs the key set associated to the j-th slot. Denoting the key set associated with to the j-th slot as KS_j, we have $KS_j = \{Key(t) | t \in TR_j\}$.

Fig. 2 shows how the associated key set for different slots in the frame are calculated. The reader predicts that t_1 and t_2 will reply in the first slot, and concludes that $TR_1 = \{t_1, t_2\}$. It then calculates the keys for tags in TR_1 and constructs the key set associated to the first slot. In this example, the keys of t_1 and t_2 are $KEY1$ and $KEY3$, respectively. Thus the key set associated to the first slot is $KS_1 = \{KEY1, KEY3\}$. Note that because the number of recorded tags is usually much larger than the total number of possible keys, different tags may have the

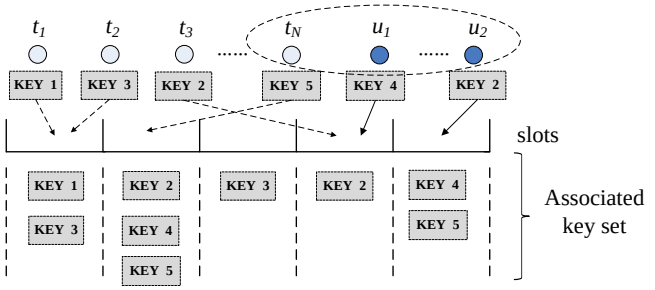

Fig. 2. Illustration of LOCK. The bottom part shows the associated key set of each slot calculated according to TR. Tags in the ellipse are covered by the reader.

same key. However, the number of tags selecting a given slot is far smaller than the total number of possible keys, and thus we can assume that tags mapped to the same slot have distinct keys. (How to guarantee this will be discussed in Section 4.2.) The bottom part of Fig. 2 shows the associated key set for every slot after all the recorded tags are considered.

In the second phase, the reader broadcasts the frame size f, the random seed r, and the key length l to its local tags to start a new frame. These parameters are the same as in the first phase. After receiving these parameters, every local tag first determines its reply slot index, then generates its key and transmits the key to the reader in its reply slot. After receiving a key in the j-th slot, the reader acts as following:

- If the key belongs to KS_j, the tag is recognized as a recorded tag. As all the recorded tags mapped to the j-th slot have distinct keys, the reader can determine which recorded tag the responding tag is. It then retrieves the tag's ID from the recorded tag list and replies an acknowledgement to make the tag silent.
- If the key is not in KS_j, the tag is recognized as an unidentified tag. The reader then sends an ID query command to collect its ID.

For example, assume that tags u_1, u_2 and t_N are the reader's local tags, as shown in Fig. 2. The reader receives $KEY5$ in the second slot. Because $KEY5$ belongs to K_2 and among all the recorded tags in TR_2 only t_N has key $KEY5$, the reader knows that the reply is from t_N. In contrast, the reader receives $KEY4$ in the fourth slot and $KEY2$ in the fifth slot, both are not in corresponding associated key sets. The reader thus knows that these replies are from unidentified tags and collects their IDs.

It is possible that the unidentified tags might be incorrectly recognized as recorded tags and thus cannot be successfully collected. For example, if u_2 transmits $KEY4$ rather than $KEY2$ in the fifth slot, then LOCK will treat it as a recorded tag and make it silent. In this case, u_2's ID will not be collected. To guarantee that the collection probability of unidentified tags is higher than the threshold α, we should carefully set the frame length f and the key length l. We will discuss how to set these parameters to guarantee α in the next section.

4.2 Parameter Settings

For an unidentified tag u that chooses the j-th slot, it could be correctly recognized and identified only when its key is different from all the keys in KS_j. To guarantee the collection probability of unidentified tags, the frame size f and the key length l should be carefully chosen in order to make the probability that tags selecting the same slot have distinct keys is higher than the desired probability α.

Consider an arbitrary slot j. Assume that there are k tags choosing this slot and the key length is l bits. Let n_{key} be the number of distinct keys generated by these k tags. Then the probability that all the k tags have distinct keys equals the probability that k numbers randomly chosen from $[1, L]$ are different from each other, where $L = 2^l$ is the total number of possible keys. It is easy to conclude that

$$Pr(n_{key} = k) = \prod_{i=1}^{k} \frac{L - i + 1}{L}. \quad (1)$$

There are totally N recorded tags in the system. Without loss of generality, we assume that there are $k = \lceil N/f \rceil$ recorded tags that choose the j-th slot. For an unidentified tag u, the probability that it could be successfully recognized and identified equals the probability that u chooses a key different from all the k keys associated with slot j, which is

$$\begin{aligned} p_d &= Pr(n_{key} = k) \times \frac{L - k}{L} \\ &= \prod_{i=1}^{k+1} \frac{L - i + 1}{L}. \end{aligned} \quad (2)$$

To guarantee that tag u could be identified with probability higher than α, it requires that

$$p_d \geq \alpha. \quad (3)$$

According to Eqs. 1 and 3, we can computer the maximum number of distinct keys that could be associated to each slot when the key length is l. Denote the maximum number of distinct keys that could be associated to each slot by k_{max}. It is obvious that k_{max} is determined by l and α, and it determines the frame size f.

The key length l should be large enough such that the reader could detect collisions when multiple tags transmit their keys in the same slot. As pointed out in previous studies [35], a minimum length of 10 bits is required for the reader to successfully detect the collision. TABLE 2 lists k_{max} for different combinations of collection probability α and key length l. It shows that k_{max} increases when the key length l increases, and decreases when the collection probability α increases.

The minimum frame size f_{min} is the ratio of the total number of recorded tags N to k_{max}, i.e., $f_{min} = N/k_{max}$. TABLE 3 lists the frame size corresponding to k_{max}

TABLE 2
k_{max} for different probability (α) and key length (l).

α	0.9	0.95	0.96	0.97	0.98	0.99
$l = 10$	14	9	8	7	5	4
$l = 12$	28	19	17	15	12	8
$l = 14$	58	40	36	31	25	17
$l = 16$	116	81	72	62	50	35

TABLE 3
Minimum frame size f_{min} when $N = 100,000$.

α	0.9	0.95	0.96	0.97	0.98	0.99
$l = 10$	7143	11112	12500	14286	20000	25000
$l = 12$	3572	5264	5883	6667	8334	12500
$l = 14$	1725	2500	2778	3226	4000	5883
$l = 16$	863	1235	1389	1613	2000	2858

given in TABLE 2. It could be observed that f_{min} is large when α approaches 1, and becomes smaller when longer keys are used. In fact, the key length l affects LOCK's performance from two aspects. On one hand, when l becomes large, the frame size becomes small and thus the time to complete the frame would be shorten. On the other hand, a longer key takes more time to be transmitted, which would increase the time to complete the frame. When setting the frame size, we should consider both of the two factors.

4.3 Improvement Directions

From Table 3 we observe that when N is large, LOCK needs to use a very long frame to guarantee the collection probability of unidentified tags. For example, when there are 100,000 recorded tags in the system and 10 bits keys are used, the frame size needs to be larger than 25,000 to guarantee that the collection probability of unidentified tags is larger than 0.99. In practice, the number of local tags (M) covered by the handheld reader is very limited. Assume that M is 500. In this case, most slots in the frame (more than 98 percent) will be empty. It is waste in time to poll these empty slots because they provide no useful information. In the next section, we propose three protocols that could drastically improve the time efficiency of LOCK by polling only non-empty slots in the frame.

4.4 Resistance to Missing Tags

LOCK is resistant to missing tags, making it different from previous approaches that require all the recorded tags to be in the system during the scanning operation [10], [12]–[14]. In LOCK, although the associated key sets for slots in the frame are calculated according to all the recorded tags in the system, the recognition of unidentified tags is based on responses from only local tags. When some recorded tags are missing, they will not change the associated key set for slots. Thus LOCK still guarantees the collection probability of unidentified tags. Furthermore, LOCK could also be used to detect missing tags. For example, the reader can roam the whole system along a planned path to perform continuous scanning. After scanning all the tags in the system, the reader can determine which recorded tags are still present in the system and which are not, and the latter ones must be missing tags. Note that missing tags cannot be determined with result of a single spot scanning operation, because the reader cannot differentiate between a missing tag and a tag out of the reader's interrogation range but still in the system.

5 Tag Scanning Protocols Based on LOCK

In this section, we develop three LOCK-based protocols to solve the spot scanning problem.

5.1 Binary LOCK

To avoid wasting time in polling empty slots, we could poll only non-empty slots to collect keys from the local tags. In this section, we propose the Binary LOCK protocol, namely BinLock, which uses binary search to quickly pinpoint non-empty slots in the frame.

5.1.1 Protocol Description

BinLock consists of multiple rounds. In each round, the reader starts a new frame in which local tags transmit their keys to the reader. However, different from the basic LOCK mechanism, in BinLock the reader polls only non-empty slots, i.e., those slots in which there is at least one local tag transmitting. The reader collects keys in non-empty slots, recognizes unidentified tags and collects their IDs. If in some non-empty slots there are more than two local tags transmitting simultaneously, the key could not be correctly received. In this case, the reader will start a new frame to scan those colliding tags.

The details of each round in BinLock are as follows. The reader first determines the frame size f, the key length l, the random seed r, and broadcasts these parameters to local tags. After receiving these parameters, a local tag calculates its key and determines its reply slot as in LOCK. When collecting replies from local tags, the reader pinpoints the first non-empty slot in the remaining frame. To quickly determine the index of the first non-empty slot, the reader broadcasts a query containing a *searching range* $[lb, ub]$. Upon receiving the searching range, a local tag checks whether its reply slot index s is in the range. If $lb \leq s \leq ub$, the tag transmits a one bit short response to the reader immediately. Otherwise, it keeps silent. The reader listens replies from tags. If it receives responses, it divides the searching range into two halves, namely $[lb, \lfloor \frac{lb+ub}{2} \rfloor]$ and $[\lfloor \frac{lb+ub}{2} \rfloor + 1, ub]$, and uses the first half as the new searching range and the second half as the candidate searching range. The reader then broadcasts the new searching range and repeats the searching process until the searching range contains only one slot. If it receives no response from the searching

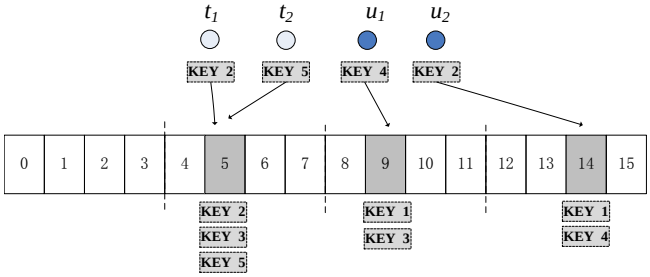

Fig. 3. Pinpointing non-empty slots in BinLock. Non-empty slots are marked with gray boxes.

range, the reader broadcasts the candidate searching range to tags and repeats the searching process until it locates the index of the first non-empty slot in the frame.

When the reader finds the first non-empty slot in the frame, it issues a key query to collect the keys of tags transmitting in this slot. When there is only one local tag selecting this slot, the reader can successfully receive the key. In this case, the reader first determines whether the replying tag is a recorded tag or an unidentified tag by using LOCK. If the tag is an unidentified tag, the reader issues an ID query command to collect its ID. If the tag is a recorded tag, the reader first determines which tag it is by using LOCK and confirms its existence. The reader also sends an acknowledgement to make the tag keep silent in the rest of the current scanning operation. If a collision happens in the slot, in which case the reader cannot successfully receive the keys, the reader sends an "NAK" to keep the replying tags active to participate the following rounds. The reader then locates the next non-empty slot in the frame and repeats the process described above. When the whole frame is scanned and there are still active local tags (when there are collisions in the frame), the reader starts a new frame to scan the remaining local tags. Otherwise, the protocol terminates.

We use Fig. 3 to show how BinLock pinpoints non-empty slots. There are totally 16 slots in the frame, among which only 3 are non-empty. The reader first sets the searching range as $[0, 15]$ and broadcasts the range to tags. All the four tags will reply to the reader, and hence the reader detects responses. It then generates the new searching range $[0, 7]$ and broadcasts it to tags. The reader receives replies from tags t_1 and t_2, thus generates the new searching range $[0, 3]$ and the candidate searching range $[4, 7]$. The reader first broadcasts $[0, 3]$. No tag will respond and the reader will not receive any responses. Thus it broadcasts the candidate searching range $[4, 7]$ to tags. The reader repeats this process until it detects responses in the interval $[5, 5]$. As there is only one slot in this range, the reader finds that the index of the first non-empty slot is 5. It then issues a key query command, upon receiving which t_1 and t_2 transmit their keys to the reader. In this example, the reader detects a collision and cannot receive the keys successfully. It sends a "NAK" to t_1 and t_2 to make them participate the next round. When searching the second non-empty slot, the reader sets the new searching interval as $[6, 15]$ and repeats the searching process.

The time efficiency could be further improved by broadcasting only the right endpoint of the searching range rather than the whole range. Assume that the previous non-empty slot is j and the current searching range is $[j+1, ub]$. Note that tags selecting slots before j will not participate in the current round. Thus the reader could broadcast only the right endpoint of the range, i.e., ub, to tags. A local tag replies to the reader when the index of its reply slot is smaller than the received value. For instance, in the example given in Fig. 3, the reader should broadcast the following ranges: $[6, 15]$, $[6, 10]$, $[6, 8]$, $[9, 10]$, and $[9, 9]$ to find the second non-empty slot. Actually, because the reader knows in which interval there are no responses from tags, it could broadcast only the right endpoints of these ranges, namely 15, 10, 8, 10 and 9, to tags. u_1 will always respond in these intervals (note that t_1 and t_2 will not respond because they are acknowledged when searching the first non-empty slot), according to which the reader can correctly find that slot 9 is the second non-empty slot.

5.1.2 Optimal Load Factor

The frame size affects BinLock's performance from two aspects. On one hand, if the frame size is very large, the reader needs many searching operations to locate a non-empty slot. On the other hand, if the frame size is small, there would be a lot of collisions and thus more rounds are needed to scan all the local tags. In this section, we analyze how to set the frame size to minimize the average time needed to find a non-empty slot. Note that the analyses in Section 4.2 give the minimum frame size to guarantee the collection probability α. We should take the larger one between the two obtained frame sizes to guarantee the collection probability, and meanwhile minimize the polling time.

Let the frame size be f in the i-th round. We define the load factor as the ratio of the number of active local tags to the frame size, i.e., $\rho = M_i/f$, where M_i indicates the number of active local tags which decreases after every round. Let N_b be the total number of binary searching operations needed to locate all the non-empty slots in the frame, and let N_s be the number of singleton slots in the frame. Then the amortized cost to successfully locate a singleton slot and receive a tag's key is

$$c = \frac{N_b}{N_s}. \tag{4}$$

Our objective is to minimize c in every round.

To calculate N_b, we first calculate the total number of non-empty slots in the frame, denoted as N_1, when the load factor is ρ, as

$$N_1 = f * (1 - (1 - \frac{1}{f})^{M_i}) \approx \frac{M_i}{\rho}(1 - e^{-\rho}). \tag{5}$$

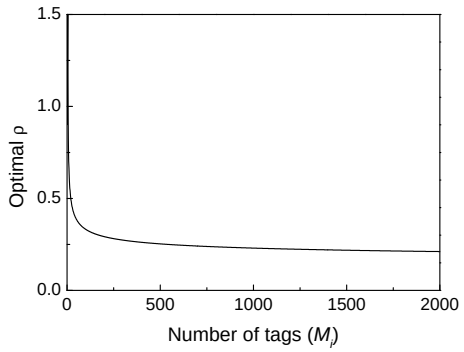

Fig. 4. Optimal load factor (ρ) in BinLock for different M_i.

Each non-empty slot requires approximately $\lg f$ binary searching operations to be located. Thus we have

$$N_b \approx N_1 * \lg f = \frac{M_i}{\rho}(1 - e^{-\rho}) \lg \frac{M_i}{\rho}. \quad (6)$$

The number of singleton slots is approximately

$$N_s = f * \frac{M_i}{f}(1 - \frac{1}{f})^{M_i - 1} \approx M_i e^{-\rho}. \quad (7)$$

Substituting Eq. 6 and Eq. 7 into Eq. 4, we get

$$c = \frac{N_b}{N_s} = \frac{1}{\rho}(e^{\rho} - 1) \lg \frac{M_i}{\rho}. \quad (8)$$

The minimum value of c is obtained when

$$\frac{\partial c}{\partial \rho} = 0. \quad (9)$$

By solving Eq. 9, we get

$$\frac{1}{\rho^2}\left[\rho e^{\rho}(\lg M_i - \lg \rho) - \frac{e^{\rho} - 1}{\ln 2} - (e^{\rho} - 1)(\lg M_i - \lg \rho)\right] = 0. \quad (10)$$

Figure 4 plots the optimal load factor for different active tag number M_i. When the reader sets the frame size, it computes the optimal frame size according to the number of remaining active tags. If the frame size is larger than f_{min} determined in Section 4.2, this frame size is used. Otherwise, f_{min} is used to guarantee the collection probability.

5.2 Segment LOCK

In BinLock, the reader needs $\lg f$ searching operations to pinpoint a non-empty slot. When f is large, it will take very long time to locate all the non-empty slots. In this section, we propose Segment LOCK, which further reduces time needed to find non-empty slots by dividing the whole frame into some short segments and search non-empty slots in only non-empty segments. We name this protocol as SegLock.

5.2.1 Protocol Description

In SegLock, the reader divides the whole frame into S fixed-length segments. Each tag determines the index of the segment it maps to. Before polling the frame, the reader starts a segment polling phase that consists of S slots, one slot for each segment, to determine which segments contain non-empty slots. A tag whose reply slot is in the j-th segment transmits a one bit short response to the reader in the j-th slot of the segment polling phase. The reader polls only non-empty segments to search for non-empty slots. Because most slots in the frame are empty, only a small part of segments would be non-empty, and thus SegLock can avoid searching in empty segments. Furthermore, denoting the segment length as L_s, searching non-empty slots in a segment incurs only $\lg L_s$ binary searching operations, far less than the $\lg f$ binary searching operations in BinLock.

The details of SegLock are as follows. In each round, the reader broadcasts the segment length L_s to tags along with other parameters used in BinLock. After receiving parameters from the reader, a tag calculates its slot index s and its segment index as $h = \lfloor \frac{s}{L_s} \rfloor$. Before polling the frame, the reader first uses a segment polling phase to determine which segments are non-empty. There are totally $S = \lceil \frac{f}{L_s} \rceil$ segments, thus the reader polls S slots in the segment polling phase. The tag whose segment index is h transmits a one bit short response in the h-th slot of the polling phase. If the reader receives replies in the h-th slot of the segment polling phase, it knows that the h-th segment contains non-empty slots. Otherwise, it knows that there would be no non-empty slots in the h-th segment. After the segment polling phase, the reader knows which segments are non-empty and polls only those non-empty segments to find non-empty slots.

After figuring out which segments are non-empty, the reader uses binary searching to pinpoint non-empty slots in only those *non-empty segments*. Because a segment is much shorter than the whole frame, finding a non-empty slot in a segment incurs much less searching operations than finding a non-empty slot in the whole frame.

We take Fig. 5 as an example to illustrate how SegLock works. The segment length is set at 4. After receiving parameters from the reader, t_1 and t_2 determine that they should reply in slot 5 which belongs to the second segment. Thus they will transmit a one bit short response to the reader in the second slot of the segment polling phase. Similarly, u_1 and u_2 know that their reply slot are in the third and fourth segments, and thus transmit in the third and fourth slots of the segment polling phase, respectively. After the segment polling phase, the reader knows that the first segment is empty and thus it searches non-empty slots in only the last three segments. In this example, BinLock uses 12 searching operations to find all the three non-empty slots, while SegLock uses only 6 searching operations plus 4 very short segment polling slots to complete the task.

SegLock reduces the time cost in two aspects. First,

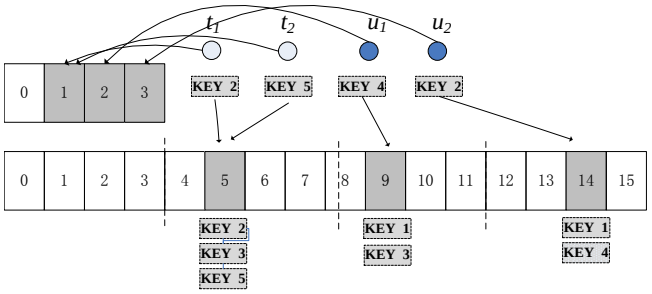

Fig. 5. Polling non-empty slots in SegLock. Before polling the original frame, the reader first polls the four slots related to the four segments. It then polls only the last three non-empty segments to find non-empty slots.

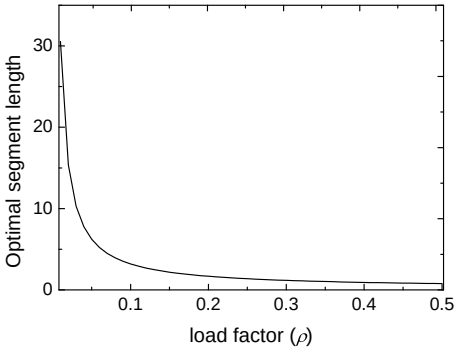

Fig. 6. Optimal segment length (L_s) for different load factor (ρ). According to [34], t_1=0.2ms and t_b=0.46ms.

the reader searches in a segment instead of in the whole frame, which involves much less searching operations to locate a non-empty slot. Second, the reader skips the empty segments, which further shrinks the searching scope. However, the segment polling process induces extra cost. The smaller L_s is, the more slots are needed in the segment polling phase, and the less time are needed to pinpoint non-empty slots in each segment. On the other hand, if we use a large L_s, the time incurred in the segment polling phase will decrease but the time to pinpoint non-empty slots in each segment will increase. We discuss how to set the optimal L_s in the next section.

5.2.2 Optimal Segment Length

Compared with BinLock, the average time to locate a non-empty slot in SegLock changes in two aspects. First, the segment polling phase in SegLock incurs additional time. There are $\lceil \frac{f}{L_s} \rceil$ slots in this phase, in each slot at most a one bit short response is transmitted. Assume that there are totally N_i non-empty slots in the frame. The increased time in the segment polling phase is shared by all the N_i non-empty slots. Second, the searching operation needed to locate a non-empty slot is reduced from $\lg f$ to $\lg L_s$. Compared with BinLock, the saved time to locate a non-empty slot in SegLock is

$$T_{save} = (\lg f - \lg L_s) * t_b - \frac{f}{L_s * N_i} * t_1, \qquad (11)$$

where t_1 is the time that a tag transmits a one bit response to the reader, t_b is the time of a searching operation that includes a query from the reader and a one bit response from tags, and $N_i \approx M_i(1 - e^{-\rho})$ is the number of non-empty slots.

To maximize T_{save}, we let

$$\frac{\partial T_{save}}{\partial L_s} = 0. \qquad (12)$$

Then we get that the optimal L_s is

$$L_s = ln2 \frac{f}{N_i} \frac{t_1}{t_b} \approx \frac{\ln 2}{1 - e^{-\rho}} \frac{t_1}{t_b}. \qquad (13)$$

L_s is affected by the load factor ρ and the ratio of t_1 to t_b. Fig. 6 plots the optimal segment length for different load factors when t_1 and t_b are set according to the EPC C1G2 specification [34].

5.3 FinalLock

In SegLock, even when a segment contains only one non-empty slot, the reader needs $\lg L_s$ searching operations to find that slot. To further improve the time efficiency, we can let the tag directly transmits its key plus the index of its slot in the segment to the reader in the segment polling phase. With this method, the reader can simultaneously obtain the tag's key and its slot index in one transmission. We call this protocol as FinalLock. As most segments contain only one non-empty slot, FinalLock avoids the searching process and hence reduces time to locate non-empty slots.

FinalLock works as follows. At the beginning of the i-th round, the reader broadcasts four parameters to tags as in BinLock and SegLock, including the frame size f, a random seed r, the key length l, and the segment length L_s. Tags determine their reply slots and corresponding segment indexes as in SegLock. However, different from SegLock in which tags transmit a one bit short response to the reader in the segment polling phase, in FinalLock each tag transmits the combination of its key and the offset of its slot in the segment to the reader. For instance, as shown in Fig. 7, u_1 maps to the second slot of the third segment. Thus u_1 transmits 'KEY4'||'1' to the reader in the third slot of the segment polling phase. After receiving this information, the reader knows that 'KEY4' is received in the ninth slot (2*4+1=9). Similarly, u_2 will transmit 'KEY2'||'2' in the fourth slot of the segment polling phase, and the reader knows that it receives 'KEY2' in the 14th slot (3*4+2=14).

FinalLock uses only segment polling phase to collect keys from tags. Compared with BinLock and SegLock, the phase to poll the original frame is completely eliminated. When there are two or more tags transmitting in the same slot, the reader replies "NAK" to keep these tags active and participate in the next round. For example, t_1 and t_2 both transmit in the second slot, and thus the reader cannot successfully receive the data. In

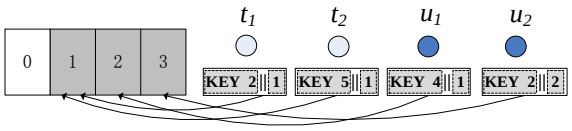

Fig. 7. Illustration of FinalLock. A tag transmits its key plus the offset of its slot in corresponding segment to the reader. The frame polling phase is removed.

this case, the reader replies "NAK" to t_1 and t_2. Upon receiving the "NAK" from the reader, t_1 and t_2 will keep silent in this round and participate in the next round.

We now analyze the performance of FinalLock. In the i-th round, there are M_i active local tags and $f_s = f/L_s$ slots in the segment polling phase. So the probability that a tag can be successfully scanned in this round is

$$p = \frac{f_s * \frac{M_i}{f_s}(1-\frac{1}{f_s})^{M_i-1}}{M_i} \approx e^{-\rho L_s}. \quad (14)$$

According to Eq. 13, we have

$$\rho L_s = \frac{\rho}{1-e^{-\rho}} \frac{t_1 * \ln 2}{t_b}. \quad (15)$$

It is easy to know that ρL_s monotonically increases when ρ increases, and thus p monotonically decreases when ρ increases. Furthermore, numerical results show that ρL_s does not change a lot when ρ varies ($0.306 \leq \rho L_s \leq 0.386$ when ρ changes from 0.01 to 0.5), and thus p also does not change a lot when ρ varies (p =0.737 when $\rho = 0.01$ and p =0.680 when $\rho = 0.5$). For simplicity in analyses, we treat p as a constant[1]. Denote the time to transmit a combination of a key and the offset in a segment by t_l. If a tag is scanned in the i-th round, it needs to transmit exactly i times. Thus, the expected time to scan a tag is

$$T = \sum_{i=1}^{\infty} p(1-p)^{i-1} * i * t_l = \frac{t_l}{p}. \quad (16)$$

For example, if we take $\rho = 0.5$ (in which case $p = 0.68$) and take t_l =0.84ms as specified in Section 6, T is approximately 1.24ms, which well coincides with our simulation results to be reported in Section 6. Moreover, T does not change much when ρ changes. For example, $T \approx$1.16ms when $\rho = 0.001$, and $T \approx$1.39ms when $\rho = 1$.

6 Simulation Results

We develop a simulator with Java to evaluate the performance of the proposed protocols. We mainly focus on spot scanning scenarios, e.g., when the reader locates at an arbitrarily specified position in a large RFID system. To our knowledge, LOCK-based protocols are the first solutions to the spot scanning problem. Thus we compare the proposed protocols with the baseline *ID-collection* solution, which directly collects IDs of all the tags in the reader's interrogation range. To demonstrate the advantages of LOCK, we also implement FUTI [14], the state-of-the-art solution to collecting IDs of unidentified tags, and make comparison between FUTI and LOCK-based protocols. We also evaluate the performance of LOCK in performing continuous scanning at the last part of this section.

We use *per tag scanning time* as the performance metric in spot scanning, and use the *total scanning time* as the performance metric in continuous scanning[2]. We consider three parameters that may affect the performance of different protocols, namely the key length (l), the collection probability of unidentified tags (α), and the ratio of unidentified tags to all the tags in the system (η). In the default setting, $l = 10$, $\alpha = 0.95$, and $\eta = 0.5$. Because missing tags do not affect the performance of LOCK-based protocols as discussed in Section 4.4, we assume that there are no missing tags, i.e., all the recorded tags are present in the system.

6.1 Simulation Scenarios and Time Setting

Spot scanning: In spot scanning scenarios, we deploy tags in a $20r \times 20r$ square region, where r is the interrogation radius of the reader. In the default setting, 64,000 tags are deployed (i.e., $N = 64,000$), resulting approximately 500 local tags in the reader's interrogation range (i.e., $M = 500$). In each simulation, we randomly select one position in the region to place the handheld reader and perform spot scanning. For each combination of parameters, we randomly generate 100 positions of the reader and report the averaged data over the 100 runs.

Continuous scanning: In continuous scanning scenarios, tags are also deployed in a $20r \times 20r$ square region. We move the reader in a grid pattern to cover the whole system and perform continuous scanning. Similarly, for each parameter setting, we randomly generate 100 instances and report the total scanning time of different protocols averaged over the 100 runs.

Time setting: We set the time duration of different slots according to the specification of EPC Class-1 Generation-2 UHF RFID tags [34]. The data rate between the reader and tags is set at 62.5Kbps. Under this data rate, the time of different slots are as follows: $t_{ID} = 2.42$ms, $t_1 = 0.2$ms, $t_k = 0.68$ms (l=10), $t_l = 0.84$ ms, and $t_b = 0.46$ms.

6.2 Impact of Key Length on LOCK's Performance

Fig. 8 plots the per tag scanning time in BinLock and FinalLock when the key length l increase from 10 to 16, assuming that there are no unidentified tags in the system ($\eta = 0$). It can be observed that when l increases, the per tag scanning time in BinLock slightly decreases when $l \leq 14$ and then becomes stable. The reason is as follows. As pointed out in Section 4.2, the key length affects BinLock's performance from two aspects.

1. It is easy to calculate the bounds of the execution time of FinalLock by using the lower bound and the upper bound of p.

2. Tags may be scanned more than one times in continuous scanning, thus it is more meaningful to compare the total scanning time.

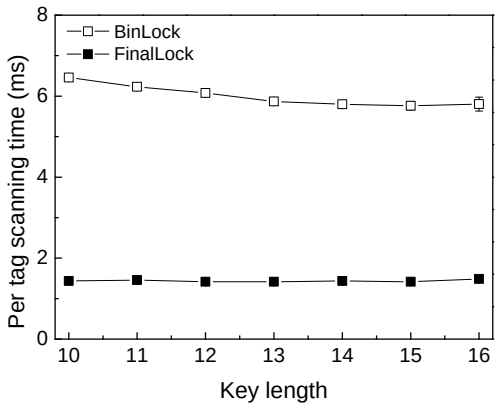

Fig. 8. Per tag scanning time in BinLock and FinalLock when the key length (l) varies.

When l is large, the required frame size to guarantee the collection probability becomes small, and thus the searching operations needed to locate each non-empty slot decreases. However, a longer key takes more time to be transmitted, which in contrast increases the per tag scanning time. Different from BinLock, FinalLock uses nearly constant time to scan a tag as it removes the frame polling phase and directly collects keys in the segment polling phase, which avoids binary searching operations that is affected by the frame length. As we mainly focus on the performance of FinalLock, in the following simulations we set the key length at 10bits.

6.3 Spot Scanning Scenarios

6.3.1 Scanning Time of Recorded Tags

This section studies per tag scanning time of recorded tags when there are no unidentified tags in the system ($\eta = 0$). We tune the the interrogation radius of the reader to change the number of local tags (M). TABLE 4 lists per tag scanning time in different protocols when M varies. When M increases, the per tag scanning time in BinLock slightly increases, while in SegLock and FinalLock the per tag scanning time keep nearly constant. The reason is that when M increases, the frame size also becomes larger, and thus BinLock needs to spend more time in locating non-empty slots. In contrast, SegLock and FinalLock effectively reduce the time needed to locate non-empty slots by segmenting. The per tag scanning time in FUTI decreases when M increases. In FUTI, the reader broadcasts multiple vectors of length N, i.e., the total number of tags in the system, and thus its execution time is mainly affected by N. When M becomes larger, the execution time of FUTI is amortized by more local tags, and thus in FUTI per tag scanning time decreases. Compared with ID-collection and FUTI, FinalLock reduces per tag scanning time by up to 70% and 98%, respectively.

TABLE 4
Per tag scanning time (ms) when M varies.

M	200	600	1000	1400	1800
ID-collection	4.33	4.33	4.33	4.33	4.33
FUTI	56.08	18.70	11.22	8.01	6.23
BinLock	5.11	5.97	6.35	6.62	6.78
SegLock	2.02	2.03	2.04	2.06	2.06
FinalLock	**1.34**	**1.32**	**1.32**	**1.33**	**1.33**

TABLE 5
Per tag scanning time (ms) when η changes.

η	0.1	0.2	0.3	0.4	0.5
ID-collection	4.33	4.33	4.33	4.33	4.33
FUTI	22.87	23.30	23.73	24.17	24.60
BinLock	6.17	6.49	6.87	7.22	7.59
SegLock	2.24	2.51	2.74	2.97	3.19
FinalLock	**1.55**	**1.79**	**2.04**	**2.28**	**2.54**

6.3.2 Impact of Unidentified Tag Ratio

We define the ratio of unidentified tags as

$$\eta = \frac{N_u}{N_u + N}, \tag{17}$$

where N_u is the number of unidentified tags in the system. Table 5 gives per tag scanning time in different protocols when η changes. When η increases, the per tag scanning time in all the considered protocols increases except ID-collection, because they need to spend more time to collect IDs of unidentified tags. BinLock performs worse than ID-collection, but still beats FUTI significantly. SegLock and FinalLock significantly outperform ID-collection. Even when $\eta = 0.5$, which means that half of tags in the reader's interrogation range are unidentified, SegLock and FinalLock reduce per tag time by 26%-48% and 41%-64%, respectively, when compared with ID-collection.

It can be observed that FinalLock outperforms ID-collection by a factor of over 40% even when η is as high as 0.5. The intrinsic reason is that LOCK can quickly detect an unidentified tag and locates its transmitting slot. Then the reader can collect the tag's ID immediately without collision. In contrast, in ID-collection, a tag needs to transmit its ID more than one times before it could be successfully collected due to collisions. Moreover, LOCK outperforms ID-collection even when all the M local tags are unidentified tags. Table 6 shows that, compared with ID-collection, FinalLock reduces per tag scanning time by 13.4% and 11.4% when $\alpha = 0.95$ and $\alpha = 0.99$, respectively, when all the local tags are unidentified tags.

FUTI performs much worse than other protocols. In FUTI, the time used to recognize unidentified tags dominates the total scanning time, and thus its per tag scanning time is mainly affected by the ratio of N/M, and is not affected a lot by the ratio of unidentified tags. Table 7 lists the ratio of the time spent in recognizing unidentified tags to the total scanning time in FUTI and

TABLE 6
Per tag scanning time (ms) for different α ($\eta = 1$).

α	0.95	0.96	0.97	0.98	0.99
ID-collection	4.33	4.33	4.33	4.33	4.33
FUTI	26.76	29.97	29.97	33.17	39.58
FinalLock	**3.72**	**3.65**	**3.77**	**3.79**	**3.81**

TABLE 7
Ratio of unidentified tag recognition time to the total scanning time in FUTI and FinalLock ($\alpha = 0.95$).

η	0.1	0.2	0.3	0.4	0.5
FUTI	.981	.963	.945	.928	.912
FinalLock	**.859**	**.752**	**.666**	**.597**	**.540**

FinalLock. In FUTI, more than 90% time is spent in recognizing unidentified tags even when $\eta = 0.5$, while the corresponding ratio in FinalLock drops to 54% in the same case.

6.3.3 Impact of Collection Probability

Table 8 gives the per tag scanning time of different protocols when the collection probability α varies. We observe that when α increases, in all the considered protocols (except ID-collection) the per tag scanning time increases. For BinLock and SegLock, a long frame is needed to guarantee a large α, and thus more time is spent to locate non-empty slots in the frame. However, the performance of FinalLock is nearly not affected by α because it removes the searching phase to find non-empty slots. The performance of FUTI degrades significantly when α increases, because it needs to execute more rounds to achieve the desired collection probability of unidentified tags. Compared with ID-collection, FinalLock reduces per tag scanning time by 41% when $\alpha = 0.95$ and by 39% when $\alpha = 0.99$, respectively.

6.3.4 Impact of Reader Occupy Ratio

We define the reader occupy ratio as the ratio of M to N. The reader occupy ratio affects the frame size and consequently per tag scanning time in FUTI and LOCK-based protocols. Fig. 9 plots per tag scanning time of different protocols when the reader occupy ratio varies, assuming that $\eta = 0.05$. When M/N is small, as shown in Fig. 9(a), FUTI and BinLock perform worse than ID-collection, while SegLock and FinalLock perform better than ID-collection. The performance of FUTI and

TABLE 8
Per tag scanning time (ms)e when α changes (η=0.5).

α	0.95	0.96	0.97	0.98	0.99
ID-collection	4.33	4.33	4.33	4.33	4.33
FUTI	24.60	27.80	27.80	31.01	37.42
BinLock	7.56	7.67	7.84	8.08	8.57
SegLock	3.18	3.20	3.51	3.64	4.10
FinalLock	**2.54**	**2.57**	**2.62**	**2.61**	**2.64**

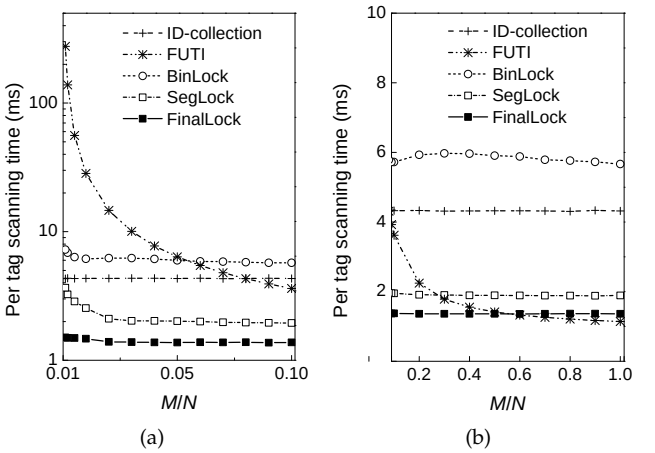

Fig. 9. Per tag scanning time with different reader occupy ratio ($\eta = 0.05$).

SegLock gradually improve when M/N increases, while FinalLock keeps nearly the same per tag scanning time. When M/N is very small, FUTI uses very long time to scan a tag. For example, when $M/N = 0.01$, FUTI spends 28.41ms to scan a tag, while FinalLock uses only 1.47ms, about 95% less that in FUTI. This demonstrates FinalLock's superior performance in performing spot scanning.

Fig. 9(b) plots per tag scanning time in different protocols when $M/N \geq 0.1$. In this case, FUTI performs better than ID-collection, and even outperforms FinalLock when $M/N > 0.6$. This owes to the high efficiency of the bit vector technique adopted in FUTI that delivers frame arranging information to tags. When M/N is large, there are no significant differences between spot scanning and continuous scanning, and thus FUTI performs well. However, in practical large scale RFID systems, the reader occupy ratio would be very small, in which case FinalLock will significantly outperform FUTI.

6.4 Continuous Scanning

Although LOCK-based protocols target at spot scanning scenarios, they can be used to perform continuous scanning by roaming the handheld reader to cover all the tags in the system. In this set of experiments, we divide the deployment region into grides with side length of $r * \sqrt{2}/2$, and move the reader to visit all the grid points to cover the whole system.

6.4.1 Impact of Collection Probability

Fig. 10 plots the total scanning time of different protocols when the collection probability α changes. The total scanning time of ID-collection is not affected by α, while the other protocols are affected by α. In continuous scanning, BinLock performs better than ID-collection because it collects each unidentified tag's ID only once, while ID-collection might collect a tag's ID more than once. FUTI significantly outperforms ID-collection in performing continuous scanning, because it also avoids

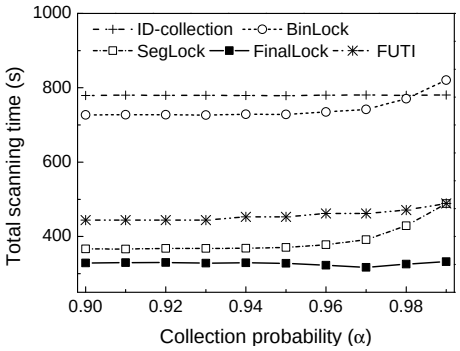

Fig. 10. Total scanning time when α changes (η=0.5).

recollecting of recorded tags' IDs. SegLock performs slightly better than FUTI. FinalLock is also not sensitive to the collection probability α, and it outperforms all other protocols. Compared with FUTI and ID-collection, FinalLock saves total scanning time by 28% and 58%, respectively.

6.4.2 Impact of Unidentified Tag Ratio

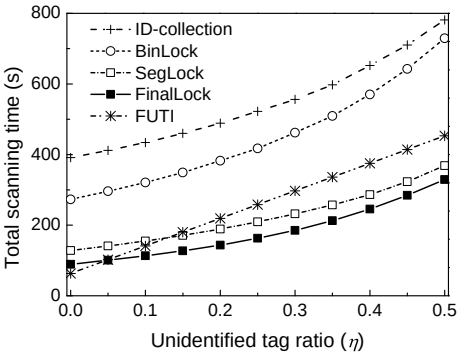

Fig. 11. Total scanning time when unidentified tag ratio changes (α=0.95).

Fig. 11 plots the total scanning time of different protocols when unidentified tag ratio changes from 0 to 0.5. In all the protocols, the scanning time increases when η becomes larger. FUTI performs very well when η is very small. When $\eta \leq 0.05$, FUTI performs even better than FinalLock. However, when $\eta \geq 0.1$, FinalLock significantly outperforms FUTI. Compared with ID-collection and FUTI, FinalLock saves total scanning time by up to 77% and up to 38%, respectively.

7 Conclusion

Spot tag scanning is an important and practical problem in large RFID systems. It provides a flexible way to scan tags in the reader's range at any specified position in the system, and is also helpful to conducting continuous scanning to scan all the tags in the system. We present the first set of protocols to solve the spot scanning problem. Both theoretical analyses and simulation results demonstrate the superior of the proposed protocols in performing spot scanning. Compared with state-of-the-art solutions, our best protocol reduces per tag scanning time by up to 70% in performing spot scanning. It also improves time efficiency in continuous scanning by up to 38%. We consider only one handheld reader in this paper. In the future, we plan to coordinate and schedule multiple handheld readers to further improve the time efficiency of both spot scanning and continuous scanning.

Acknowledgment

This work is partially supported by HK RGC PolyU 5281/13E, the National Science Foundation of China under Grant Nos. 61103203,, 61373181 61402404, and the Fundamental Research Funds for the Central Universities under Grant No. 2014QNA5012. The authors would also like to sincerely thank the Editors and reviewers for their thoughtful, constructive suggestions and comments. Dr. Shigeng Zhang is the corresponding author.

References

[1] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley, 2003.

[2] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk, "Efficient and reliable low-power backscatter networks," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 61–72, 2012.

[3] D. K. Klair, K.-W. Chian, and R. Raad, "A survey and tutorial of RFID anti-collision protocols," *IEEE Communications Survery and Tutorials*, vol. 2, no. 3, pp. 400–421, 2010.

[4] L. Xie, Y. Yin, A. V. Vasilakos, and S. Lu, "Managing RFID data: Challenges, opportunities and solutions," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 3, pp. 1294–1311, 2014.

[5] V. Namboodiri and L. Gao, "Energy-aware tag anticollision protocols for RFID systems," *IEEE Transactions on Mobile Computing*, 2009.

[6] Y. Zhu, W. Jiang, Q. Zhang, and H. Guan, "Energy-Efficient Identification in Large-Scale RFID Systems with Handheld Reader," *IEEE Transactions on Parallel and Distributed Systens*, vol. 25, no. 5, pp. 1211–1222, 2014.

[7] L. Xie, Q. Li, X. Chen, S. Lu, and D. Chen, "Continuous Scanning with Mobile Reader in RFID Systems: An Experimental Study," in *Proceedings of the 14tn ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, 2013, pp. 1–10.

[8] M. Shahzad and A. X. Liu, "Probabilistic optimal tree hopping for RFID identification," in *Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2013, pp. 293–304.

[9] Y. Yin, L. Xie, J. Wu, A. V. Vasilakos, and S. Lu, "Focus and shoot: Efficient identification over RFID tags in the specified area," in *Proceedings of the 10th International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services (MOBIQUITOUS)*, 2013, pp. 344–357.

[10] B. Sheng, Q. Li, and W. Mao, "Efficient continuous scanning in RFID systems," in *Proceedings of the 29th IEEE International Conference on Computer Communications (Infocom)*. IEEE, 2010, pp. 1–9.

[11] H. Liu, W. Gong, X. Miao, K. Liu, and W. He, "Towards Adaptive Continuous Scanning in Large-Scale RFID Systems," in *Proceedings of the 33rd IEEE International Conference on Computer Communications (Infocom)*, 2014, pp. 486–494.

[12] X. Liu, S. Zhang, K. Bu, and B. Xiao, "Complete and fast unknown tag identification in large RFID systems," in *Proceedings of the 9th IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS)*, 2012, pp. 47–55.

[13] X. Liu, B. Xiao, S. Zhang, and K. Bu, "Unknown Tag Identification in Large RFID Systems: An Efficient and Complete Solution," *IEEE Transactions on Parallel and Distributed Systems*, vol. pp, no. 99, pp. 1–14, 2014.

[14] X. Liu, K. Li, G. Min, K. Lin, B. Xiao, Y. Shen, and W. Qu, "Efficient Unknown Tag Identification Protocols in Large-Scale RFID Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. pp, no. 99, pp. 1–12, 2014.

[15] J. Cha and J. Kim, "Novel anti-collision algorithms for fast object identification in RFID system," in *Proceeding of IEEE International Conference on Parallel and Distributed Systems (IPDPS)*, vol. 2, 2005, pp. 63–67.

[16] D. Klair, K. Chin, and R. Raad, "An investigation into thie energy eficiency of pure and slotted aloha based RFID anti-collision protocols," in *Proceedings of the IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2007.

[17] L. Kang, K. Wu, J. Zhang, H. Tan, and L. M. Ni, "DDC: A Novel Scheme to Directly Decode the Collisions in UHF RFID Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 2, pp. 263–270, 2012.

[18] M. Zhang, T. Li, S. Chen, and B. Li, "Using Analog Network Coding to Improve the RFID Reading Throughput," in *Proceedings of 2010 International Conference on Distributed Computing Systems (ICDCS)*, 2010, pp. 547–556.

[19] J.-S. Li and Y.-M. Huo, "An efficient time-bound collision prevention scheme for rfid re-entering tags," *IEEE Transactions on Mobile Computing*, vol. 12, no. 6, pp. 1054–1064, 2013.

[20] J. Myung, W. Lee, J. Srivastava, and T. K. Shih, "Tag-Splitting: Adaptive Collision Arbitration Protocols for RFID Tag Identification," *IEEE Transactions on Parallel and Distributed Systens*, vol. 18, no. 6, pp. 763–775, 2007.

[21] Y.-C. Lai, L.-Y. Hsiao, H.-J. Chen, C.-N. Lai, and J.-W. Lin, "A novel query tree protocol with bit tracking in rfid tag identification," *IEEE Transactions on Mobile Computing*, vol. 12, no. 10, pp. 2063–2075, 2013.

[22] S. Chen, M. Zhang, and B. Xiao, "Efficient information collection protocols for sensor-augmented RFID networks," in *Proceedings of the 30th IEEE International Conference on Computer Communications (Infocom)*, 2011, pp. 3101–3109.

[23] T. Li, S. Chen, and Y. Ling, "Efficient Protocols for Identifying the Missing Tags in a Large RFID System," *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 1974–1987, 2013.

[24] W. Luo, S. Chen, T. Li, and Y. Qiao, "Probabilistic missing-tag detection and energy-time tradeoff in large-scale RFID systems," in *Proceedings of the 13tn ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, 2012, pp. 95–104.

[25] Y. Zheng and M. Li, "P-MTI: Physical-layer Missing Tag Identification via compressive sensing," in *Proceedings of the 32nd IEEE International Conference on Computer Communications (Infocom)*, 2013, pp. 917–925.

[26] H. Han, B. Sheng, C. Tan, Q. Li, W. Mao, and S. Lu, "Counting RFID tags efficiently and anonymously," in *Proceedings of the 29th IEEE International Conference on Computer Communications (Infocom)*, 2010, pp. 1–9.

[27] M. Kodialam and T. Nandagopal, "Fast and reliable estimation schemes in RFID systems," in *Proceedings of the 12th annual international conference on Mobile computing and networking (Mobicom)*. ACM, 2006, pp. 322–333.

[28] Y. Zheng and M. Li, "PET: Probabilistic estimating tree for large-scale RFID estimation," *IEEE Transactions on Mobile Computing*, vol. 11, no. 11, pp. 1763–1774, November 2012.

[29] T. Li, S. Wu, S. Chen, and M. Yang, "Energy efficient algorithms for the RFID estimation problem," in *Proceedings of the 29th IEEE International Conference on Computer Communications (Infocom)*, 2010, pp. 1–9.

[30] M. Shahzad and A. X. Liu, "Every bit counts: fast and scalable RFID estimation," in *Proceedings of the 18th annual international conference on Mobile computing and networking (Mobicom)*. ACM, 2012, pp. 365–376.

[31] B. Chen, Z. Zhou, and H. Yu, "Understanding RFID counting protocols," in *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking (Mobicom)*, 2013, pp. 291–302.

[32] W. Gong, K. Liu, X. Miao, and H. Liu, "Arbitrarily Accurate Approximation Scheme for Large-Scale RFID Cardinality Estimation," in *Proceedings of the 33rd IEEE International Conference on Computer Communications (Infocom)*, 2014, pp. 477–485.

[33] L. Xie, H. Han, Q. Li, J. Wu, and S. Lu, "Efficiently collecting histograms over RFID tags," in *Proceedings of the 32nd IEEE International Conference on Computer Communications (Infocom)*, 2014, pp. 145–153.

[34] EPCglobal, "EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz - 960 MHz Version 1.2.0," 2008.

[35] Y. Qiao, S. Chen, T. Li, and S. Chen, "Energy-efficient polling protocols in RFID systems," in *Proceedings of the 12th ACM international symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*. ACM, 2011, pp. 25–33.

Xuan Liu received the BSc degree in Information and Computing Mathematics from XiangTan University, China, and MSc degree in Computer Science from National University of Defense Technology, China, in 2005 and 2008, respectively. Currently, she is a PhD candidate in the Department of Computing at The Hong Kong Polytechnic University, Hong Kong. Her research interests include distributed computing systems, mobile computing, focusing on wireless sensor networks and RFID systems.

Shigeng Zhang received the BSc, MSc, and DEng degrees, all in Computer Science, from Nanjing University, China, in 2004, 2007, and 2010, respectively. He is currently an Assistant Professor in School of Information Science and Engineering at Central South University, China. His research interests include Cloud Computing, Internet of Things, wireless sensor networks, and RFID systems. He has published more than 30 technique papers in international journals and conferences. He is a member of IEEE and CCF.

Bin Xiao received the B.Sc and M.Sc degrees in Electronics Engineering from Fudan University, China, and Ph.D. degree in computer science from University of Texas at Dallas, USA. After his Ph.D. graduation, he joined the Department of Computing of the Hong Kong Polytechnic University as an Assistant Professor. Now he is an associate professor and the director of the Mobile Cloud Computing Lab. His research is mainly on mobile cloud computing, smart phone technology, network security, and wireless networks. He is the editor of 3 books and has published more than 100 technical papers in international journals and conferences. Currently, he is the associate editor of the International Journal of Parallel, Emergent and Distributed Systems and an editor of the International Journal of Distributed Sensor Networks. He is the IEEE Senior member.

Kai Bu received the BSc and MSc degrees in computer science from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2006 and 2009, respectively, and the Ph.D. degree in computer science from The Hong Kong Polytechnic University, Hong Kong, in 2013. He is currently an Assistant Professor in the College of Computer Science and Technology at Zhejiang University, China. His research interests include RFID and wireless networks. He is a recipient of the Best Paper Award of IEEE/IFIP EUC 2011. He is a member of the ACM and the IEEE.