The following publication X. Liu, B. Xiao, S. Zhang and K. Bu, "Unknown Tag Identification in Large RFID Systems: An Efficient and Complete Solution," in IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 6, pp. 1775-1788, 1 June 2015 is available at https://doi.org/10.1109/TPDS.2014.2326651.

Unknown Tag Identification in Large RFID Systems: An Efficient and Complete Solution

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

Abstract—Radio-Frequency Identification (RFID) technology brings revolutionary changes to many fields like retail industry. One important research issue in large RFID systems is the identification of unknown tags, i.e., tags that just entered the system but have not been interrogated by reader(s) covering them yet. Unknown tag identification plays a critical role in automatic inventory management and misplaced tag discovery, but it is far from thoroughly investigated. Existing solutions either trivially interrogate all the tags in the system and thus are highly time inefficient due to re-identification of already identified tags, or use probabilistic approaches that cannot guarantee complete identification of all the unknown tags. In this paper, we propose a series of protocols that can identify all of the unknown tags with high time efficiency. We develop several novel techniques to quickly deactivate already identified tags and prevent them from replying during the interrogation of unknown tags, which avoids re-identification of these tags and consequently improves time efficiency. To our knowledge, our protocols are the first non-trivial solutions that guarantee complete identification of all the unknown tags. We illustrate the effectiveness of our protocols through both rigorous theoretical analysis and extensive simulations. Simulation results show that our protocols can save up to 70 percent time when compared with the best existing solutions.

Index Terms—RFID system; unknown tag identification; time efficiency; slot pairing; multiple reselections

1 Introduction

Radio-Frequency Identification (RFID) technology brings revolutionary changes to many applications, e.g., warehouse management, object tracking, and inventory control. In these applications, active or passive tags are attached to objects (e.g., persons and products) and the information stored in tags (e.g., tag IDs) is interrogated by reader(s) to facilitate efficient management of objects. The process of collecting tag IDs is usually referred to as *tag identification*. Efficient tag identification plays a critical role in stimulating innovative application of RFID in various fields, and has attracted a lot of research attention in recent years [1]–[9].

In this paper, we focus on an important but not thoroughly investigated problem, *unknown tag identification* in large RFID systems. Unknown tags are those tags that just entered the system but have not been interrogated by the reader(s) covering them yet. For instance, in a large warehouse, the frequent loading of new products introduces unknown tags into the system, which should be interrogated in time in order to support automatic and efficient product management. Misplaced tags due to misoperation of stevedores could also be treated as unknown tags by reader(s) currently covering them. Timely interrogation of these unknown tags should be carried out to avoid considerable economic profit loss caused by misplacement errors [10].

Although many approaches on efficient tag identification have been presented in the past several years [1], [3]–[6], [11]–[14], few of them dedicated to unknown tag identification. Existing work on tag identification mainly focuses on optimizing time/energy efficiency in tag identification [1], [3], [11], or improving throughput by exploiting parallel working of multiple readers [5], [15]–[17], or utilizing mobile reader(s) to facilitate flexible tag reading [13], [14], [18]. However, they all target at interrogating *all the tags* in the system from scratch, which would be highly time inefficient if directly adopted to solve the unknown tag identification problem. The low efficiency of these solutions in identifying unknown tags stems from redundant re-identification of already identified tags (hereinafter referred to as *known tags*), which usually constitute the majority of the tag population in a large RFID system.

The most related work on unknown tag identification is the *continuous scanning* (CU) scheme [19], which is a probabilistic scheme that cannot guarantee complete identification of all the unknown tags. In safety-critical applications like medicine management in hospitals, it is strictly required that all the unknown tags must be identified. The CU scheme cannot meet this requirement. The execution time of CU increases along with the increase of the probability required to interrogate each unknown tag. When the probability is extremely high, the execution time might be comparable to or even longer than the trivial solution that identifies all the tags in the system.

In this paper, we propose a series of protocols that can identify all the unknown tags with high time effi-

- *Xuan Liu and Bin Xiao are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, and the PolyU Shenzhen Research Institute, China. E-mail: {csxuanliu, csbxiao}@comp.polyu.edu.hk.*
- *Shigeng Zhang is with School of Information Science and Engineering at Central South University and the State Key Laboratory for Novel Software Technology at Nanjing University, China. E-mail: sgzhang@csu.edu.cn.*
- *Kai Bu is with the College of Computer Science and Technology at Zhejiang University, China. E-mail:kaibu@zju.edu.cn.*

ciency. We find that the key challenge in speeding up unknown tag identification is *how to effectively prohibit the involvement of known tags when interrogating unknown tags*. Our first contribution is to propose a method to recognize known tags and deactivate them to prohibit their further replies. The recognition is realized by comparing expected replies of those known tags with actual received replies in each slot of a frame.

Our second contribution is to develop two novel techniques, *slot pairing* (Section 5) and *multiple reselections* (Section 6) to resolve the known tag collision. These two techniques can greatly enhance the efficiency in deactivating known tags. They can also be applied as common techniques to improve tag identification efficiency in RFID systems. We build a series of time efficient unknown tag identification protocols on top of these techniques.

Our third contribution is to conduct extensive simulations to evaluate the performance of our unknown tag identification protocols with various parameter settings. The results show up to 70 percent reduction in execution time when compared to the best existing solutions. Furthermore, in order to set optimal frame size in our protocols, we develop a zero-cost algorithm to estimate the number of unknown tags that are currently present in the system and evaluate its accuracy.

We organize our paper as follows. In Section 2 we overview related work. In Section 3 we describe system model and give problem statement. In Section 4 we describe the basic unknown tag identification protocol (BUIP) and point out directions to further improve its performance. The two novel techniques, i.e., slot pairing and multiple reselections, and the two enhanced protocols built on them, i.e., SUIP and MUIP, are described in Section 5 and Section 6, respectively. Section 8 discusses fault tolerance issues. Section 9 presents the zero-cost unknown tag cardinality estimation algorithm. The performance of our protocols are evaluated and compared with state-of-the-art solutions through extensive simulations, and the results are reported in Section 10. Finally, Section 11 concludes the paper.

2 Related work

Early works on RFID tag identification focus on collision arbitration for a single reader. They can be classified into two categories [4]: ALOHA-based protocols and tree-based protocols. In [3] and [20] , the authors investigated optimization of time efficiency and energy efficiency, respectively, by adjusting frame size in ALOHA-based tag identification protocols. In [21] and [22], the authors proposed adaptive tree traversal methods to improve time efficiency of tree-based protocols. Kang et al. [23] and Xie et al. [14] reported that tag identification throughput degrades in real deployment. These works all target at identifying all the tags in the system, and thus are not efficient to solve the unknown tag identification problem considered in this paper.

Mobile readers can provide flexible tag identification [13], [14], [18] in infrastructure-less (i.e., with no pre-installed readers) RFID systems. Xie et al. [14] reported observations on the relationship between identification throughput and the transmitting power of the readers. Based on the observations, they designed algorithms to optimize energy and time efficiency of the reader in large RFID systems containing more than one hundred tags. Zhu et al. [18] discussed how to plan the trajectory of the mobile reader to save energy. These works are also complementary to the protocols proposed in this paper.

The most related work is the CU scheme proposed in [19]. It uses a probabilistic method to identify unknown tags, and thus cannot guarantee *complete identification of all the unknown tags*. CU first predicts which slot should be empty according to the IDs of known tags, assuming no unknown tags exist. In the predicted non-empty slots, the reader sends *ACK* to temporarily prohibit replies from known tags. In the predicted empty slots, the reader sends *NAK* to keep unknown tags active for following identification. It then collects IDs of active unknown tags. Unavoidably, some unknown tags may be prohibited in the predicted non-empty slots and thus cannot be identified. The protocol runs multiple rounds to guarantee that the probability that a required fraction of unknown tags are identified is higher than a threshold. However, CU cannot ensure that all the unknown tags are identified. In contrast, the protocols proposed in this paper adopt deterministic approaches to identifying all the unknown tags in the system. Furthermore, two novel techniques, namely slot paring and multiple reselections, are developed to reduce the time needed to identify unknown tags. These techniques can be commonly used in RFID systems adopting ALOHA arbitration protocols to improve tag identification efficiency.

3 Background

3.1 System Model and Problem Overview

Consider a large RFID system consisting of a reader R and a set of tags $T = \{k_1, \ldots, k_n, u_1, \ldots, u_m\}$, where $k_i (1 \leq i \leq n)$ denotes a know tag that has its ID in the system database, and $u_j (1 \leq j \leq m)$ denotes an unknown tag. All the known tag IDs are recorded in a back-end server. The objective is to *collect all the unknown tag IDs (i.e., $\{u_1, \ldots, u_m\}$) as quickly as possible*. Note that in the current RFID tag identification protocols, the reader has no simple way to differentiate unknown tags from known tags, and thus has to endure the interference from known tags when identifying unknown tags. This work aims to design novel techniques to quickly recognize known tags and prohibit their replying during the identification of unknown tags.

We use the *frame slotted ALOHA* protocol [2] as the underlying MAC layer protocol. The communication between the reader and tags takes the form of *frame*, which is organized as a number of slots synchronized by the reader in which tags transmit information (e.g.,

tag ID) to the reader. The reader starts a frame with a query $<f, r>$, where f indicates the number of slots in the frame and r is a random seed used by tags to determine which slot to transmit. A tag hashes its ID to an integer in [0,f-1] by $S = H(ID||r)$ mod f and transmits in the S-th slot, where H is a uniform hash function. A tag can transmit either its ID or a short response (e.g., a 16-bits random number *RN16* in EPC C1G2 standard [2]) to the reader. We denote by t_{ID} and t_l the time needed to transmit a tag ID and a short response, respectively.

The protocols proposed in this paper can also be applied to RFID systems containing multiple readers. In such cases, we resort to existing reader scheduling algorithms [5], [15]–[17] to obtain a conflict-free schedule of readers and run our protocols on every reader. Our protocols could also be tailored for mobile reader scenario(s) by treating tags identified at the previous site as known tags at the current site. The reader(s) can access the back-end server via wired or wireless link(s) to retrieve the known tag list before running the protocols and update the list afterwards.

3.2 Assumptions

It is possible that known tags might leave the system, and there are some researches on detecting such missing tags [24]–[27]. In this work, we mainly focus on the unknown tag identification problem and assume that no known tags leave the system *during the identification of unknown tags*, which is usually very short, e.g., several minutes. This assumption has been taken in many excellent previous works [28], [29]. To do this, the reader needs to trace which known tags have left the system before executing our protocols, and update the known tag list in the back-end server accordingly. In case that the reader cannot trace which known tags have left, we resort to *missing tag detection* protocols [24]–[26] to find which known tags have left and update the known tag list accordingly. After unknown tags have been identified, we insert their IDs into the database to keep the known tag list updated.

The reader uses *indicator vectors* [28] to send some frame arranging information to tags. An indicator vector is a vector of bits that can be received and interpreted by tags. For example, we can set the bit associated with a slot to "1" to prevent tags selecting this slot from transmitting. This technique has been used in many existing researches to improve protocol efficiency. Here we adopt the method given in [28] to implement indicator vector on top of EPC G1G2 compliant tags. The indicator vector is divided into segments of 96 bits long and each segment is encapsulated into a tag ID. The reader broadcasts segments one after other. Tags need to buffer only one segment in which their corresponding bit resides in. We can also add cyclic-redundancy check (CRC) code to each segment to ensure that the segment could be correctly received. The indicator vector technique based on the method given in [28] has been adopted in many excellent existing researches including [24], [27]–[32].

4 BUIP: The Basic Unknown Tag Identification Protocol

In this section, we propose the Basic Unknown tag Identification Protocol (BUIP) and analyze its performance.

4.1 Protocol Design

BUIP consists of two phases: the *known tag deactivation* phase and the *unknown tag collection* phase. In the first phase, the reader recognizes and deactivates all the known tags to prevent them from interfering with the identification of unknown tags in the second phase. In the second phase, the reader completely identifies unknown tags by collecting their IDs.

The known tag deactivation phase consists of multiple rounds. In each round, the reader collects replies from tags, based on which it recognizes and deactivates known tags (Section 4.2). The reader also recognizes and *labels* unknown tags to prevent them from interfering with the recognition of known tags. To shorten the execution time of each round, the reader broadcasts an indicator vector to prohibit tags in collision slots from replying, because collision slots could not be used to recognize known tags or unknown tags.

The number of active known tags decreases after each round. The reader traces how many known tags have been deactivated. When all the known tags have been deactivated, it enters the second phase to identify all the unknown tags. Otherwise, it starts a new round to deactivate the remaining known tags.

4.2 Known Tag Deactivation and Unknown Tag Labeling

In this section we develop a technique to recognize known tags and unknown tags by comparing the actually received replies in a slot with the expected relies.

Being aware of known tags' ID, the reader knows what tags will transmit in which slot. Consequently, the reader knows the *expected reply number* in each slot (i.e., the number of known tags transmitting in this slot) if there are no unknown tags. However, replies from unknown tags might make the *actual reply number* in a slot different from the expected value. The matching/mismatching between the two numbers provides us opportunities to recognize known tags and unknown tags: If the actual reply number equals the expected reply number, then all the replying tags *must* be known tags; on the other hand, if the actual reply number is larger than the expected value, there *must be* some unknown tags replying.

We implement the known/unknown tag recognition method as follows. In the ALOHA protocol the reader can obtain only coarse-grained status of a slot: empty (the reply number is 0), singleton (the reply number is 1), and collision (the reply number is larger than 1). Because the reader cannot know the exact reply number in a collision slot, it cannot determine whether the actual reply number in a collision slot matches its

expected value or not. Thus, we use only expected empty slots (whose expected reply number is 0) and expected singleton slots (whose expected reply number is 1) to recognize known and unknown tags:

- **Recognize known tags:** If the reader receives only one reply in an expected singleton slot, it recognizes the replying tag as a known tag.
- **Recognize unknown tags:** If the reader receives one or more replies in an expected empty slot, it recognizes the replying tag(s) as unknown tag(s).

After recognizing known tags and unknown tags, the reader deactivates or labels them accordingly. For recognized known tags, the reader deactivates them in both of the two phases in our protocol. In contrast, for recognized unknown tags, the readers make them *temporally inactive* only in the first phase, which we call *unknown tag labelling*. The purpose of labelling is to prevent unknown tags from interfering the recognition of known tags. All the labelled unknown tags will become active again in the second phase to perform identification.

Known tag deactivation and unknown tag labelling are implemented by sending different types of acknowledgements to the tag. The EPC specification [2] provides two different types of acknowledgements: *ACK* to deactivate a tag and *NAK* to keep a tag active. Noting that in our protocol we need to differentiate between deactivation of known tags and labelling of unknown tags, we extend *ACK* to implement the two different purposes: We use ACK_d for known tag deactivation and use ACK_l for unknown tag labelling. We append one bit to *ACK* to differentiate ACK_d and ACK_l: $ACK_d = ACK +' 0'$ and $ACK_l = ACK +' 1'$. If a tag receives ACK_d, it will not respond to the reader during the execution of the whole protocol. If a tag receives ACK_l, it enters the labeled status and responds to only the ID-collection command issued by the reader in the second phase.

4.3 Protocol Description

BUIP consists of two phases: the known tag deactivation phase and the unknown tag collection phase. In the second phase, the reader employs existing protocols like DFSA [33] to collect all unknown tag IDs. We focus on the first phase.

The first phase consists of multiple rounds. In each round, the reader first broadcasts a query $<f, r>$ and an indicator vector $\mathbf{v}_c$ to tags. $\mathbf{v}_c$ is an f-bits long vector in which the S-th bit indicates whether the S-th slot would be colliding or not. The reader constructs $\mathbf{v}_c$ according to the remaining active known tags in the current round. The reader predicts in which slot each known tag will response according to the tag's ID and the query $<f, r>$. It thus knows the expected status of every slot and sets bits in $\mathbf{v}_c$ as follows: The S-th bit is set to '1' if the S-th slot would be colliding, and '0' otherwise.

After receiving the query, every tag first calculates its transmission slot index as $h = H(ID||r) \bmod f$. It then checks the h-th bit in $\mathbf{v}_c$. If the h-th bit is '0', it transmits a short response (e.g., *RN16* in EPC C1G2 standard [2]) to the reader in the selected slot; otherwise, it keeps silent because the selected slot must be colliding.

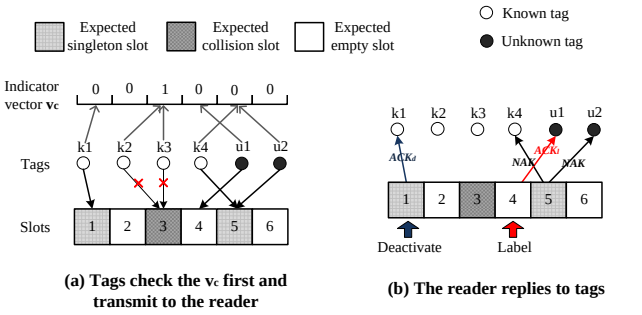


Fig. 1. Illustration of BUIP: (a) Prohibiting replies from tags mapped to expected collision slots; (b) Deactivating known tags and labelling unknown tags.

The reader collects replies from tags, recognizes known/unknown tags and deactivates/labels them. In the S-th slot, the reader sends different acknowledgements: 1) If the slot is an expected singleton slot and the reader receives only one reply, it recognizes a known tag and sends an ACK_d to deactivate that tag. 2) If the slot is an expected empty slot and the reader receives some replies, it recognizes unknown tags and sends an ACK_l to label them. 3) In all other cases, the reader sends a *NAK* to keep tags active. At the end of each round, the reader counts the accumulated number of deactivated known tags up to the current round. If all the known tags have been deactivated, the reader enters the second phase and collects unknown tag IDs. Otherwise, it starts a new round to deactivate the remaining known tags.

Fig. 1 illustrates an example of BUIP with four known tags and two unknown tags. The reader first broadcasts $\mathbf{v}_c$ that indicates the third slot is colliding. The two known tags mapped to the third slot, i.e, k_2 and k_3, find $\mathbf{v}_c[2] =$ '1' and do not reply in this round, turning the third slot to an empty slot. Other tags reply in their chosen slots. The reader recognizes k_1 in the first slot and replies ACK_d to deactivate it. Meanwhile, the reader recognizes u_1 in the fourth slot and replies ACK_l to label it. In this example, the reader needs to start a new round to deactivate the rest three known tags.

4.4 BUIP Analysis and Discussions

We first derive how to set the frame size to maximize the known tag deactivation efficiency. Consider the i-th round. Let T_i and D_i be the execution time and the number of deactivated known tags in the i-th round, respectively. We define the *amortized cost* to deactivate a known tag as

$$C_i = \frac{T_i}{D_i}. \tag{1}$$

C_i represents the average time needed to deactivate a known tag in the i-th round, and thus should be minimized to optimize the known tag deactivation efficiency.

When the frame size is set at f_i, the execution time is $T_i = t_{ID} * \frac{f_i}{96} + f_i * t_l$, where the first part is the time used to broadcast the indicator vector $\mathbf{v}_c$, and the second part is the time used to collect replies from tags. Let P_d be the probability that a tag can be deactivated in an arbitrary slot S. Then the expected number of deactivated known tags is $D_i \approx f_i * P_d$. Thus, the amortized cost to deactivate a known tag when the frame size is f_i is

$$C_i = \frac{T_i}{D_i} = \frac{f_i * (\frac{t_{ID}}{96} + t_l)}{f_i * P_d} \propto \frac{1}{P_d}. \quad (2)$$

We can see that C_i is inversely proportional to P_d. Thus, we can reduce the amortized cost by increasing the probability P_d. However, as indicated by Remark 1, P_d cannot be increased arbitrarily by tuning only frame size.

***Remark** 1:* In BUIP, P_d is maximized when $f_i = m_i + n_i - 1$, and $P_d \leq e^{-1} \approx 0.368$.

Proof: A slot S can be used to deactivate a known tag only when exactly one known tag selects S and no other tags select it, with probability

$$P_d = \binom{n_i}{1} \frac{1}{f_i} (1 - \frac{1}{f_i})^{n_i + m_i - 1} \approx \frac{n_i}{f_i} e^{-\frac{n_i + m_i - 1}{f_i}}. \quad (3)$$

It is easy to derive that P_d is maximized when $f_i = n_i + m_i - 1$, in which case

$$P_d \approx \frac{n_i}{f_i} e^{-1} \leq \frac{1}{e} \approx 0.368. \quad (4)$$

□

Remark 1 shows that we cannot obtain arbitrary large P_d by adjusting only frame size in BUIP. To further improve the deactivation efficiency of BUIP, we need to develop new techniques.

5 SUIP: Single-Pairing Unknown Tag Identification Protocol

In this section, we first present a novel technique called *slot pairing* that can greatly increase P_d, then present the Single-pairing Unknown tag Identification Protocol (SUIP).

5.1 Protocol Design

The novelty of SUIP is that it uses a novel technique to turn expected collision slots into expected singleton slots and thus improves known tag deactivation efficiency. Recall that in BUIP only expected singleton slots are used to deactivate known tags, and replies in expected collision slots are prohibited. In SUIP, we pair every expected collision slot with an expected empty slot and let tags mapped to the collision slot make a reselection between the two slots. For example, if two known tags k_1 and k_2 are mapped to an expected collision slot S, they are given another chance to reselect one slot from S and its pairing empty slot S'. After the reselection, it is possible that k_1 selects S and k_2 selects S' or vice versa. In this case, both S and S' turn into expected singleton slots and can be used to deactivate known tags. We call this technique *slot pairing*. Note that slot pairing can also resolve collisions when there are more than two tags colliding. For example, if there are k ($k \geq 3$) known tags selecting the collision slot, then it is possible that only one of them reselects S (S') and all the other $k-1$ tags reselect S' (S), which also generates an expected singleton slot after slot reselection.

5.2 Slot Pairing

In slot paring, every *expected collision slot* is paired with an *expected empty slot*. As there are more expected empty slots than expected collision slots when the frame size is optimally set (i.e., $f_i = n_i + m_i - 1$), we can assign a pairing empty slot for *every* expected collision slot. Note that the number of expected empty slots ($f_i * e^{-n_i/f_i}$) is always larger than the number of expected collision slots ($f_i * (1 - \frac{n_i}{f_i} e^{-n_i/f_i} - e^{-n_i/f_i})$) when $n_i/f_i \leq 1$. Meanwhile, when $f_i \geq n_i$, most collisions are two-collisions [34] (i.e., exactly two known tags select this slot), in which case two expected singleton slots could be generated if the two known tags select different slots after reselection.

We pair the j-th expected collision slot with the j-th expected empty slot. A tag that originally selects the j-th expected collision slot may reselect the j-th expect empty slot in the second chance. The challenge is how to inform a tag of the index of its paired slot in the frame.

In SUIP, we exploit two indicator vectors: the indicator vector $\mathbf{v}_c$ that is used to indicate the expected collision slots, and the indicator vector $\mathbf{v}_e$ that is used to indicate the expected empty slots. Both $\mathbf{v}_c$ and $\mathbf{v}_e$ have f bits. Each bit is associated with a slot at the same index location in the frame. However, these two indicator vectors are constructed in different ways:

- **In $\mathbf{v}_c$**: If the k-th slot is expected to be colliding, $\mathbf{v}_c[k]$ = '1'; otherwise, $\mathbf{v}_c[k]$ = '0'.
- **In $\mathbf{v}_e$**: If the k-th slot is expected to be empty, $\mathbf{v}_e[k]$ = '1'; otherwise, $\mathbf{v}_e[k]$ = '0'.

A tag t mapped to an expected collision slot has two kinds of index values: $I_f(t)$ denotes its slot index in the frame, which indicates how many slots there are before its slot in the frame, and $I_c(t)$ denotes specifically the collision slot index, which indicates how many expected collision slots there are before its slot in the frame (i.e., the number of '1's before its corresponding bit in $\mathbf{v}_c$). It determines the paired expected empty slot in two steps:

- First, t examines the collision index $I_c(t)$ of its slot by counting how many '1's there are before its slot bit in the vector $\mathbf{v}_c$.
- Second, t determines the index of its pairing empty slot in the frame by searching the $(I_c(t)+1)$-th '1' in the vector $\mathbf{v}_e$. Let the index of the $(I_c(t)+1)$-th '1' in $\mathbf{v}_e$ be $I'_f(t)$. It then uses the $I'_f(t)$-th slot as the paired slot.

5.3 Protocol Description

The second phase of SUIP is the same as in BUIP, thus we describe only the first phase.

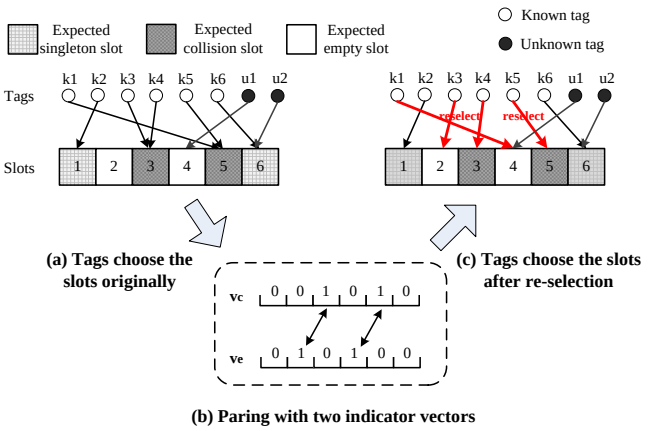


Fig. 2. The process of slot paring: (a) before slot reselection (same as BUIP); (b) slot pairing and reselection; (c) after slot reselection.

The known tag deactivation phase consists of multiple rounds. At the beginning of each round, the reader computers the slot indexes of expected empty slots, expected singleton slots and expected collision slots in the frame with the known tag IDs, and then constructs two indicator vectors $\mathbf{v}_c$ and $\mathbf{v}_e$. The reader broadcasts a query $<f, r>$ and two indicator vectors $\mathbf{v}_c$ and $\mathbf{v}_e$. It also broadcasts another seed r' for tags to reselect either of the two pairing slots.

Tags use the received information to determine their transmission slots. Upon receiving f and r, a tag t first calculates its slot index $I_f(t) = H(ID||r) \bmod f$. It then checks the bit value $\mathbf{v}_c[I_f(t)]$ to determine whether it should perform slot reselection or not. If $\mathbf{v}_c[I_f(t)] =$ '0', the tag responses in the original chosen slot $I_f(t)$. If $\mathbf{v}_c[I_f(t)] =$ '1', the tag needs to find its paired slot and perform reselection. Tag t determines its paired slot as previously described and randomly chooses either slot as its final transmitting slot. The reselection result is controlled by r': If $H(ID||r') \bmod 2 = 0$, the tag replies in its original slot; otherwise it replies in the paired slot.

When receiving the responses from tags, the reader deactivates the known tags in expected singleton slots and labels unknown tags in expected empty slots, the same as in BUIP. Note that with r' and IDs of known tags, the reader exactly knows all reselection results of known tags. Thus the reader knows the new expected singleton slots transformed from the expected collision slots and expected empty slots after reselection.

Fig. 2 illustrates an example of SUIP. We can see that before slot pairing and reselection (Fig. 2(a), actually as same as BUIP), only one known tag (k_2) can be deactivated. In contrast, after slot pairing and reselection, four known tags (k_2, k_3, k_4, and k_5) can be recognized and deactivated. Fig. 2 also illustrates the negative effects of slot pairing on unknown tag recognition. In Fig. 2(a) the unknown tag u_1 can be recognized. However, after slot pairing and reselection, there are no expected empty slots and thus no unknown tags can be recognized. As the execution time of our protocols is mainly affected by the efficiency in deactivating known tags, SUIP achieves much better overall performance than BUIP.

6 MUIP: Multi-Pairing Unknown Tag Identification Protocol

In this section, we propose a Multi-pairing Unknown tag Identification Protocol (MUIP) that resolves collision slots with a higher probability and further improves the efficiency in deactivating known tags.

6.1 Protocol Overview

MUIP is similar to SUIP except that it makes two changes to further resolve the collisions between tags.

First, MUIP provides multiple chances to separate collided known tags between pairing slots, which could resolve the expected collision slots with a higher probability. Reselecting with one seed r_i as in SUIP, colliding tags could be separated with a probability of only about 50%. We observe that a collision slot that cannot be resolved with r_i may be resolved with another seed r_j. If the reader sends *multiple seeds* and let tags use the most suitable one, more collision slots could be resolved.

Second, in MUIP, unknown tags label themselves without sending responses to the reader, which avoids collisions between unknown tags and known tags in expected empty slots after slot reselection. An expected empty slot may be actually *non-empty* if some unknown tags hash to it. When a known tag reselects such a non-empty slot, it would collide with the unknown tags and cannot be deactivated successfully. For example, as shown in Fig. 2, although k_1 reselects the expected empty slot 4 which resolves the collision in slot 5, it collides with u_1 and still cannot be deactivated. If we can guarantee that the paired slots are actually empty, the reader would deactivate more known tags.

6.2 Slot Reselection Using Multiple Seeds

The reader will send h seeds $\{r_1, r_2, ..., r_h\}$ to provide multiple reselection chances for known tags that hash to expected collision slots. Then the collided tags have h possible reselection results by hashing with the h seeds. If one of these results separates the collided known tags (i.e., at least one expected singleton slots will be generated with a seed r_i), the expected collision slot could be resolved successfully. Remember that, when reselecting with a seed r_i, the tag calculates $H(ID||r_i) \bmod 2$. It replies in its original slot if the value equals 0, and replies in the paired slot otherwise. The process that a tag finds its paired slot is the same as in SUIP.

The reader examines the hash results of the known tags with the h seeds, and determines which one should be used for each expected collision slot. The problem is how to inform different known tags that which seed is the suitable one.

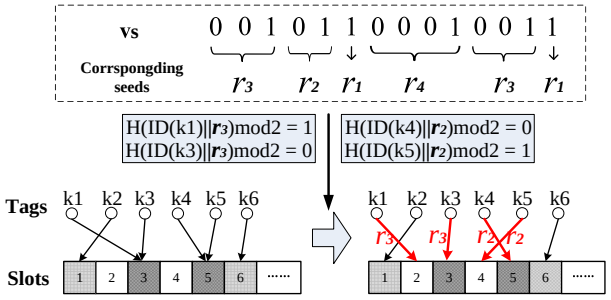


Fig. 3. The process of multiple reselections: Tags mapped to different expected collision slots use different seeds in slot reselection.

We exploit a *seed-selection vector* $\mathbf{v}_s$ to indicate the suitable seed assigned for each expected collision slot. $\mathbf{v}_s$ consists of n_c elements, where n_c is the number of expected collision slots (i.e., one element for each expected collision slot). The i-th element of $\mathbf{v}_s$ is constructed as follows:

- If seed r_{l+1} $(0 \leq l \leq h)$ should be used to conduct reselection by tags mapping to the i-th collision slot, we set $\mathbf{v}_s[i]$ as "$\underbrace{0\ldots0}_{l}1$".

To obtain the assigned seed, a tag t has to find the $I_c(t)$-th element in $\mathbf{v}_s$. When t finds that it chooses an expected collision slot according to $\mathbf{v}_c$, it first obtains the collision index $I_c(t)$ of its chosen slot. Then it locates the $I_c(t)$-th '1' in the vector $\mathbf{v}_s$, and counts the number of '0's of $I_c(t)$-th element in $\mathbf{v}_s$ (i.e., the number of '0's between the $(I_c(t)-1)$-th '1' and the $I_c(t)$-th '1'). If there are l '0's, then the tag uses the seed r_{l+1}. If all the h seeds cannot resolve the expected collision slot, the corresponding element in $\mathbf{v}_s$ will be set as "1", which indicates that seed r_1 should be used.

Fig. 3 illustrates an example of the construction of the indicator vector $\mathbf{v}_s$. We can see that in $\mathbf{v}_s$, each element indicates a seed. The example shows the reselection of the known tags in the first two expected collision slots. Two tags, k_1 and k_3, know that they choose the first expected collision slot after receiving $\mathbf{v}_c$ and $\mathbf{v}_e$. These two tags find their assigned seed r_3 indicated by the first element in $\mathbf{v}_s$. Thus k_1 and k_3 reselect either of the two pairing slots with the hash function $H(ID||r_3)$ mod 2. As the result, k_1 chooses the second slot and k_3 chooses the third slot, which resolves the first expected collision slot successfully. The reselection process of k_4 and k_5 in the second expected collision slot is similar.

6.3 Automatical Unknown Tag Labeling

Except for reselecting with a suitable seed, to successfully deactivate the known tag in the paired expected empty slot, the other condition is that the slot must be actually empty (i.e., no known tag chooses the slot). Otherwise, the reselecting known tag will collide with unknown tags. As shown in Fig. 2, the reselecting known tag k_1 still collides with the unknown tag u_1. However, in SUIP, the reader needs the responses of unknown tags in expected empty slots to label them, which consequently reduces the efficiency of known tag deactivation.

In MUIP, unknown tags label themselves if they choose expected empty slots. Obviously, if a tag chooses an expected empty slot, it must be an unknown tag. In this case, the tag could label itself without receiving the *NAK* from the reader. A tag could test whether it chooses an expected empty slot by checking corresponding bit in the indicator vector $\mathbf{v}_e$ at the location of its slot index. If the bit value is '1', it labels itself without responding to the reader. Otherwise, it determines its slot and responds to the reader in the chosen slot.

6.4 MUIP Description

The known tag deactivation phase of MUIP consists of multiple rounds. At the beginning of each round, the reader broadcasts a query $< f, r >$ and three indicator vectors $\mathbf{v}_c$, $\mathbf{v}_e$ and $\mathbf{v}_s$. It also broadcasts h reselection seeds $\{r_1, r_2, ..., r_h\}$ for tags to reselect either of the two pairing slots. Upon receiving these parameters, tags determine their transmission slots. Assume that tag t originally selects slot k. It checks the expected status of its chosen slot in both $\mathbf{v}_c$ and $\mathbf{v}_e$:

- If $\mathbf{v}_c[k] =$ '0' and $\mathbf{v}_e[k] =$ '0', which indicates that it chooses an expected singleton slot, the tag responds to the reader in the chosen slot k.
- If $\mathbf{v}_c[k] =$ '0' and $\mathbf{v}_e[k] =$ '1', which indicates that it chooses an expected empty slot, the tag labels itself and keeps silence until receiving the ID-collection command.
- If $\mathbf{v}_c[k] =$ '1', which indicates that it chooses an expected collision slot, the tag needs to reselect its transmission slot. It first examines its paired slot in $\mathbf{v}_e$ and then reselects its slot with the assigned seed indicated in $\mathbf{v}_s$. The tag responds to the reader in the reselected slot.

When receiving responses from tags, the reader deactivates known tags in expected singleton slots and labels unknown tags in expected empty slots as done in SUIP.

7 Analyses of SUIP and MUIP

In this section, we analyze the performance of SUIP and MUIP. As shown in Section 4.4, the average time to deactivate a known tag is inversely proportional to P_d, thus we mainly focus on analyzing P_d in SUIP and MUIP.

7.1 Analysis of SUIP

Without loss of generality, we consider an arbitrary slot in the i-th round. Denote by P_E, P_S, P_C the probability that this slot is an expected empty slot, an expected singleton slot, and an expected collision slot, all before slot pairing and reselection, respectively. Denote by $P\{D|S\}$ the probability that a known tag is deactivated in an expected singleton slot. Similarly, denote by $P\{D|C\}$ and

$P\{D|E\}$ the probability that a known tag is deactivated in an expected collision and in an expected empty slot after slot reselection, respectively. Then we have

$$P_d = P_S * P\{D|S\} + P_C * P\{D|C\} + P_E * P\{D|E\}. \quad (5)$$

It is easy to calculate P_E, P_S and P_C as

$$P_E = e^{-n_i/f_i}, P_S = \frac{n_i}{f_i} e^{-n_i/f_i}, P_C = 1 - P_E - P_S,$$

where $f = m_i + n_i - 1$. In the following, we discuss how to calculate $P\{D|S\}$, $P\{D|C\}$ and $P\{D|E\}$.

Calculation of P{D|S}: As same as in BUIP, the probability that a known tag can be deactivated in an expected singleton slot equals the probability that no unknown tag selects this slot, i.e.,

$$P\{D|S\} = (1 - \frac{1}{f_i})^{m_i} \approx e^{-m_i/f_i}. \quad (6)$$

Calculation of P{D|C}: For an expected collision slot, we denote by $P_C(k_1, k_2)$ the conditional probability that exactly k_1 known tags and k_2 unknown tags select it. Because all the n_i known tags select replying slot independently, k_1 follows a binomial distribution $B(n_i, 1/f_i)$ that can be approximated with a *Poisson distribution* with parameter $\lambda_n = n_i/f_i$. Similarly, k_2 also follows a Poisson distribution with parameter $\lambda_m = m_i/f_i$. Noting $\lambda_n + \lambda_m \approx 1$, we have

$$\begin{aligned} P_C(k_1, k_2) &= \frac{1}{P_C} * e^{-\lambda_n} * \frac{\lambda_n^{k_1}}{k_1!} * e^{-\lambda_m} * \frac{\lambda_m^{k_2}}{k_2!} \\ &\approx \frac{e^{-1}}{P_C} \frac{\lambda_n^{k_1} \lambda_m^{k_2}}{k_1! k_2!}. \end{aligned} \quad (7)$$

Let $P\{D|k_1, k_2\}$ be the probability that a known tag is deactivated in this collision slot after reselection. Then $P\{D|k_1, k_2\}$ equals the probability that exactly one of the k_1 known tags selects this slot, and the other $k_1 - 1$ known tag(s) and k_2 unknown tags all select the paired slot after reselection, which is

$$P\{D|k_1, k_2\} = k_1 * \frac{1}{2} * (1 - \frac{1}{2})^{k_1 + k_2 - 1} = k_1 * (\frac{1}{2})^{k_1 + k_2}. \quad (8)$$

Combining equation 7 and equation 8, we have (the detailed derivation is given in Appendix)

$$\begin{aligned} P\{D|C\} &= \sum_{k_1=2}^{n_i} \sum_{k_2=0}^{m_i} P_C(k_1, k_2) * P\{D|k_1, k_2\} \\ &\approx \frac{e^{-1/2} \lambda_n}{2 * P_C} (1 - e^{-\lambda_n/2}). \end{aligned} \quad (9)$$

Calculation of P{D|E}: An expected empty slot needs to satisfy three conditions to be able to deactivate a known tag: (1) it is a paired slot of some collision slot; (2) no unknown tag selects it; and (3) after reselection, only one known tag reselects it. As there are $f_i * P_E$ expected empty slots and only $f_i * P_C$ expected collision slots, the probability for an expected empty slot to satisfy the first condition is $(f_i * P_C)/(f_i * P_E) = P_C/P_E$. The probability to satisfy the second condition is approximately e^{-m_i/f_i}.

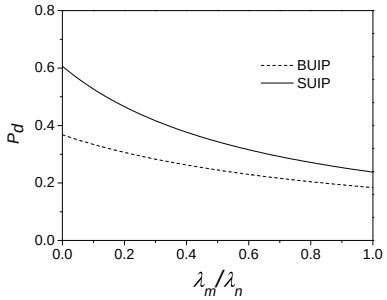


Fig. 4. P_d in SUIP vs. P_d in BUIP when λ_m/λ_n varies.

The probability to satisfy the third condition equals $P\{D|C\}$. Thus, we can calculate $P\{D|E\}$ as

$$P\{D|E\} = \frac{P_C}{P_E} * e^{-m_i/f_i} * P\{D|C\}. \quad (10)$$

Substituting equations 6, 9, 10 into equation 5, we obtain

$$P_d = \lambda_n e^{-1} + \frac{e^{-1/2} \lambda_n}{2} (1 - e^{-\lambda_n/2})(1 + e^{\lambda_m}). \quad (11)$$

***Remark** 2:* In SUIP, P_d is a monotonically decreasing function of λ_m/λ_n, and it achieves maximum value when $\lambda_n = 1$ (i.e., $\lambda_m = 0$), which is

$$P_d^{max} = e^{-1} + e^{-1/2}(1 - e^{-1/2}) \approx 0.6065. \quad (12)$$

Fig. 4 plots P_d in SUIP vs. P_d in BUIP when λ_m/λ_n increases from 0 to 1. We observe significant increase of P_d in SUIP compared with in BUIP: P_d in SUIP is 65 percent higher than that in BUIP when $\lambda_m/\lambda_n = 0$ (i.e., there are no unknown tags in the system), and is 29 percent higher when $\lambda_m/\lambda_n = 1$ (i.e., when $n_i = m_i$). The improvement gradually decreases due to interferences caused by unknown tags.

7.2 Analysis of MUIP

7.2.1 *Impact of h on P_d*

In this section we derive $P_d(h)$, the probability that a known tag can be deactivated in an arbitrary slot when h reselection seeds are used. We denote by $P_h\{D|C\}$ and $P_h\{D|E\}$ the probability that a known tag is deactivated in an expected collision slot and in an expected empty slot when h seeds are used, respectively. Using the same notations defined in Section 7.1, we have

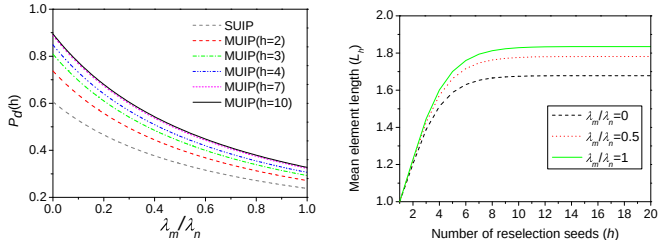
$$P_d(h) = P_S * P\{D|S\} + P_C * P_h\{D|C\} + P_E * P_h\{D|E\}. \quad (13)$$

Recall that the probability of a known tag t can be deactivated in an original expected collision slot with one seed is $P\{D|C\}$. Thus the probability that t cannot be deactivated with all of the h seeds is $(1 - P\{D|C\})^h$, which leads to

$$P_h\{D|C\} = 1 - (1 - P\{D|C\})^h. \quad (14)$$

As in SUIP, we can calculate $P_h\{D|E\}$ as

$$P_h\{D|E\} = \frac{P_C}{P_E} * e^{-m_i/f_i} * P_h\{D|C\}. \quad (15)$$



(a) n (b) m

Fig. 5. (a) $P_d(h)$ in MUIP vs. P_d in SUIP when λ_m/λ_n varies, and (b) Mean segment length when h increases.

TABLE 1
Ratio of singleton slots in MUIP and MIC [28] with different h.

h	2	3	4	5	6	7
MIC	0.580	0.696	0.764	0.808	0.839	0.861
MUIP	0.607	0.737	0.809	0.849	0.870	0.881

Substituting equations 14, 15 into equation 13, we have

$$P_d(h) = \lambda_n e^{-1} + P_C * (1 + e^{1-\lambda_n}) * \Psi, \quad (16)$$

where

$$\Psi = 1 - \left[1 - \frac{e^{-1/2}\lambda_n}{2 * P_C}(1 - e^{-\lambda_n/2})\right]^h. \quad (17)$$

***Remark** 3:* When λ_m/λ_n is fixed, $P_d(h)$ is a monotonically increasing function of h.

Proof: From equation 17 it is easy to see that Ψ is a monotonically increasing function of h, which immediately implies Remark 3 according to equation 16. □

Fig. 5(a) plots $P_d(h)$ for different h when λ_m/λ_n varies from 0 to 1. We observe that when h is small (e.g., when $h = 2, 3, 4$), increasing h can significantly increase $P_d(h)$. On the other hand, when h is large enough, increasing h leads only marginal increase of $P_d(h)$. For example, we can observe that $P_d(10)$ is nearly the same as $P_d(7)$. Actually, the increase of h cannot arbitrarily increase $P_d(h)$: Because $\lim_{h\to\infty} \Psi = 1$, we have

$$\lim_{h\to\infty} P_d(h) = \lambda_n e^{-1} + P_C * (1 + e^{1-\lambda_n}). \quad (18)$$

When simulating MUIP in Section 10, we set $h = 7$.

7.2.2 Multiple Reselections Technique Efficiency

We compare the performance of our multi-pairing technique with the multiple hashing approach MIC proposed in [28] from two aspects: The probability of a slot becoming singleton when h seeds are used, and the total length of the bit vectors sent from the reader to tags.

TABLE 1 lists the ratio of expected singleton slots in MUIP and MIC when different number of seeds are used. It shows that MUIP generates more singleton slots than MIC does with the same number of seeds. In MIC, the colliding tags can reselect any slot that has not been occupied. With one reselection chance, the probability that a colliding tag can reselect an expected singleton slot is about $1/e \approx 0.368$. Thus the efficiency of this technique is affected mainly by *the number of reselection chances*. On the other hand, in MUIP, a colliding tag reselects between two paired slots. For a two-collision slot (i.e. the collision is caused by two tags responding simultaneously), the slot paring technique has a probability of 0.5 to generate two expected singleton slots. For a collision slot in which k tags responding, slot pairing can still generate one expected singleton slot with a probability of $k/2^{k-1}$. The efficiency of MUIP is affected mainly *by the number of colliding tags in a slot*. With the optimal frame size in both MUIP and MIC, most collisions are two-collisions. As a result, MUIP performs better than MIC.

In MIC, for every slot the reader broadcasts three bits to represent the seed index that tags mapped to this slot should use to select replying slot. Thus the total number of bits broadcasted in MIC is $3 * f_i$. In contrast, in MUIP the reader broadcasts two f_i-bits long vectors to notify tags which slots are empty and collision. Besides, in MUIP the reader also broadcasts a seed-selection vector $\mathbf{v}_s$ for *only collision slots*. Let L_h be the mean length of elements in $\mathbf{v}_s$. The total number of bits broadcasted in MUIP is $2 * f_i + f_i * P_C * L_h$, where P_C is the probability that a slot is collision.

We derive L_h as follows. Recall that in MUIP if the l-th seed should be used in slot reselection, the length of corresponding element is l. Denote by $P_s(l)$ the probability that the l-th seed should be used in slot reselection. Noting that the first seed is used when none of the h seeds can resolve the collision slot, we can calculate the mean element length as

$$L_h = 1 * \left[1 - \sum_{l=2}^{h} P_s(l)\right] + \sum_{l=2}^{h} l * P_s(l). \quad (19)$$

$P_s(l)$ equals the probability that the collision slot cannot be resolved by $r_1, \ldots, r_{l-1}$ but can be resolved by r_l, thus

$$P_s(l) = (1 - P\{D|C\})^{l-1} * P\{D|C\}. \quad (20)$$

Substituting equation 20 into equation 19, with a given λ_m/λ_n, we can calculate L_h for different h. Fig. 5(b) plots L_h for different λ_m/λ_n when h increases. When $h = 7$ and $\lambda_m/\lambda_n = 0$, i.e., the scenario that MIC addresses, $L_h \approx 1.6533$. Note that $P_C = 1 - e^{-n_i/f_i} - \frac{n_i}{f_i}e^{-n_i/f_i} \leq 1 - 2 * e^{-1} \approx 0.2642$ when $\frac{n_i}{f_i} \leq 1$. Thus the total number of bits broadcasted in MUIP is only $2 * f_i + f_i * P_C * L_h \leq (2 + 0.2642 * 1.6533) * f_i = 2.4369 * f_i$, much less than the $3 * f_i$ bits in MIC.

It can also be observed from Fig. 5(b) that L_h is always smaller than 2, which shows that our approach represents seed index more efficiently than MIC does. The small average length of elements owe to the *variable length coding* of the seed index: Most collision slots are resolved by the first or the second seed (see Fig. 5(a)), and when none of the h seeds can resolve the collision slot we just use the first seed.

8 Fault Tolerance

In this section we discuss how to cope with channel errors to enhance the robustness of our protocols.

8.1 Reader to Tag Transmission Error

The indicator vector transmitted from the reader to tags might be corrupted due to channel errors. If the bit corresponding to a tag's replying slot is incorrectly received (i.e., the sent bit is "0" but the received bit is "1" or vice versa), the tag might take incorrect actions. In order to help tags check whether the received information is correct or not, we can add a cyclic-redundancy check (CRC) code in each segment when transmitting the indicator vector. For example, we can divide the indicator vector into 80 bits long segments and add a 16 bits CRC code to each segment. After receiving the indicator vector, the tag first checks whether its indicator bit is correctly received. If the segment is correctly received, the tag takes its action according to its corresponding bit as described in our protocols. Otherwise, it does not participate in the current round, but will keep active to participate in the next round. For SUIP and MUIP, a tag participates in the current round only when it correctly receives all its corresponding bits.

With this mechanism, the correctness of our protocols can be guaranteed even when transmission error happens. The key point to guarantee the correctness of our protocols is that the reader have to correctly trace how many known tags have been deactivated, according to which it determines when to terminate the first phase. Recall that the reader deactivates known tags only when it receives exact one reply in expected singleton slots. If a known tag receives corrupted information, it will not participate in the current round. However, the other unknown tags selecting the same slot, if there are some, will also not participate in the current round due to transmission error. Thus the reader will not receive any response in the slot. If the reader receives no reply in an expected singleton slot, it skips this slot and does not count the corresponding known tag as being deactivated. In this way, the reader can trace the *correct number of deactivated known tags*, and thus can guarantee the correctness of the protocols.

The time efficiency of the protocols would degrade slightly when we use this scheme. The EPC C1G2 specification [2] provides two types of CRC code: CRC16 that uses 16 bits and CRC5 that uses 5 bits. If we adopt CRC16, according to the time setting given in the specification [2], in each round the execution time of BUIP will be increased by

$$\frac{t_{ID} * (\frac{f_i}{80} - \frac{f_i}{96})}{t_{ID} * \frac{f_i}{96} + f_i * t_l} = \frac{2.4 * (\frac{1}{80} - \frac{1}{96})}{2.4 * \frac{1}{96} + 0.44} = 0.0108, \quad (21)$$

which is only about 1.1 percent. Similarly, when CRC16 is used, the execution time of SUIP and MUIP will be increased by no more than 2 percent and 3 percent, respectively. If we use the shorter CRC5 code, the execution time will be increased by only 0.3, 0.6, and 0.8 percent in BUIP, SUIP, and MUIP, respectively.

8.2 Tag to Reader Transmission Error

Our protocols can tolerate the tag to reader transmission errors. Note that in our protocols tags transmit 16-bits long short responses (*RN16*) to the reader rather than a single bit. The main purpose of the responses is to help the reader differentiate different status of a slot (e.g., empty, singleton, or collision) to recognize known tags and unknown tags. The transmission error will not affect the status of a slot, and thus will not break the correctness of our protocols.

9 Zero-Cost Estimation of Unknown Tags

Our protocols need to know the number of active unknown tags (m_i) to set the optimal frame size. If we use a separate estimation algorithm [35]–[38] to estimate m_i, it will inevitably increase the execution time. We develop a zero-cost estimation algorithm to estimate m_i by using the information collected in the deactivation phase.

The algorithm is motivated by the observation that the actual status of a slot may differ from its expected status because of unknown tags' interference. For example, if an unknown tag replies in an expected empty (singleton) slot, the actual status of the slot will be singleton (collision). The ratio of slots whose actual status are different from their expected status is related to the number of active unknown tags, i.e., m_i. Thus, by counting how many slots change their status in the previous round, we can estimate m_i in the current round.

Our estimation algorithm works as following. Consider the i-round in which there are n_i known tags and m_i unknown tags, respectively, and assume that the frame size is f_i. Let R_U be the ratio of tags whose actual status differ from expected status. For slot S, its actual status differs from its expected status if and only if there are at least one unknown tags select S, with probability

$$1 - (1 - \frac{1}{f_i})^{m_i} \approx 1 - e^{-m_i/f_i}. \quad (22)$$

If we know R_U and f_i, we can estimate m_i by letting

$$R_U = 1 - e^{-m_i/f_i}, \quad (23)$$

from which we can derive that

$$m_i = -f_i * \ln(1 - R_U). \quad (24)$$

The problem is that f_i also depends on m_i (recall that f_i should be set to $n_i + m_i - 1$). To solve this dilemma, we assume $m_1 = 0$ and set $f_1 = n$ in the first round. At the end of the first round, the reader can obtain R_U by counting the number of slots changed from empty to non-empty or from singleton to collision. It then estimates m_1 with equation 24. Note that m_1 is

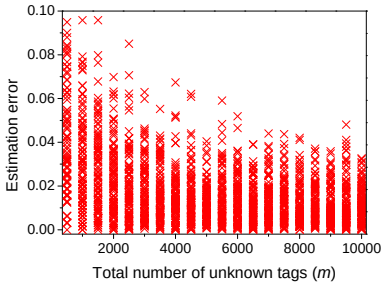


Fig. 6. Estimation error when m changes (n=10,000).

the number of active unknown tags *at the beginning of this round*, but we need to know the number of active known tags *at the end of this round* to set the frame size in the next round. Let Δm_1 be the number of labelled unknown tags in the first round, then we have

$$m_2 = m_1 - \Delta m_1. \tag{25}$$

We then can use m_2 to set frame size in the second round. With this method, we can estimate m_{i+1} with the information collected in the i-th round and set the frame size in the $(i+1)$-th round accordingly.

We estimate the number of the labelled unknown tags in the i-th round, i.e., Δm_i, as following. Let r_e be the ratio of expected empty slots in the i-th round. Because unknown tags select their transmission slots uniformly, Δm_i can be estimated as

$$\Delta m_i = m_i * r_e, \tag{26}$$

where r_e can be calculated as the ratio of the number of expected empty slots N_e to the total number of slots in the frame, i.e.,

$$r_e = \frac{N_e}{f_i}. \tag{27}$$

Fig. 6 plots the estimation error of our algorithm when m varies from 500 to 10000 stepped by 500, assuming $n = 10000$. For each m we plot the estimation error of our algorithm in 100 independent executions of BUIP. It can be seen that the estimation error decreases along with the increase of m, and is smaller than 0.05 in most cases. We point out here that our estimation algorithm incurs no additional cost to our unknown tag identification protocols, because it utilizes the information collected in the i-th round to estimate m_{i+1} in the $(i+1)$-th round.

10 Performance Evaluation

10.1 Simulation settings

We develop a simulator with JAVA to evaluate the performance of our protocols. Our protocols are compared with two methods: A *Baseline* method that collects IDs of all the tags in the system, and an *Ideal* method that collects IDs of only unknown tags. The execution time of the Baseline method is an upper bound on the time needed to identify all the unknown tags, and the execution time of the Ideal method represents a lower bound. We also compare our protocols with the CU scheme [19], which is the most efficient probabilistic unknown tag identification algorithm up to now.

We adopt the timing scheme specified for the EPC Global Class 1 Generation 2 UHF tags [2] to compute the execution time of different protocols. We set the bidirectional transmission rate between the reader and tags at 62.5 Kbps. With this setting, it takes about 2.4ms to transmit a tag ID and 0.44ms to transmit a 16-bit short response, i.e., $t_{ID} = 2.4ms$ and $t_l = 0.44ms$. An empty slot takes 0.184ms. A slot in CU takes the same time as an empty slot.

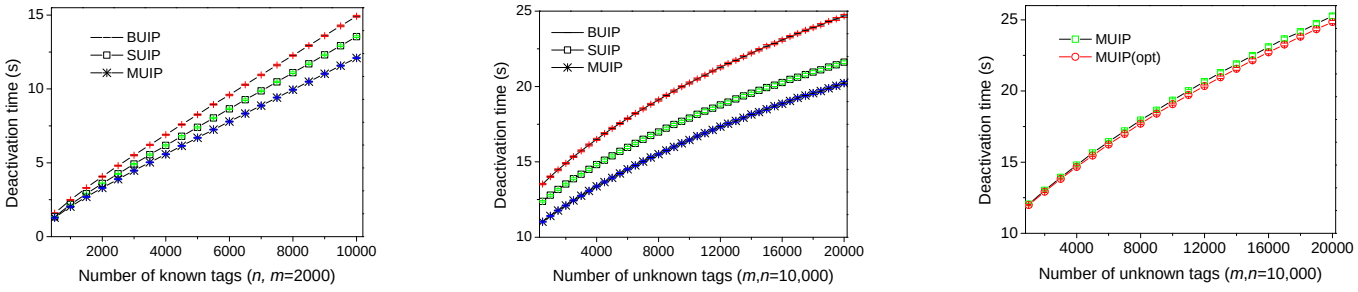
The main performance metric is the execution time. We consider two system parameters, the number of known tags (n) and the number of unknown tags (m), and tune their values to study their impacts on the execution time of different protocols. Their default values are set as $n = 10,000$ and $m = 2,000$, respectively. For MUIP, we set the number of reselection seeds at $h = 7$. For every parameter setting, we run the considered protocols in 100 independent instances and report the average data.

We use the approach proposed in [27] to detect missing tags before executing our protocols, and add this overhead to the execution time of our protocols when comparing with other approaches. We implement the DFSA [33] protocol to identify unknown tags in CU and our protocols. When implementing DFSA, we use the optimal frame size, which is fair to all the considered approaches. We note that the identification throughput may degrade in real environments [14], [23]. However, as this will affect the performance of the considered protocols in the same way, we do not consider this issue in our paper.

10.2 Deactivation Time

We first investigate how the number of known tags (n) and the number of unknown tags (m) affect the deactivation time (i.e., the execution time of the first phase) in our protocols. Intuitively, the larger n and m are, the longer time needed to deactivate all the known tags in the first phase. Fig. 7(a) plots the deactivation time of the three protocols when n changes. We observe that when there are more known tags, the deactivation time in all the three protocols increases. SUIP and MUIP outperform BUIP by using slot pairing and multiple reselections, which greatly increases the probability P_d and consequently decreases the amortized cost to deactivate a known tag. Compared with BUIP, SUIP and MUIP reduce deactivation time by up to 15 percent and 19 percent, respectively.

As we have pointed out in the design of our protocols, replies from unknown tags will disturb the recognition of known tags and thus increase the deactivation time. Fig. 7(b) plots the deactivation time in the three protocols when m increases. In all the three protocols, the deactivation time increases when m increases. However, SUIP



(a) n (b) m (c) m

Fig. 7. Deactivation time of different protocols: (a) when n changes; (b) when m changes; (c) Impact of estimation error.

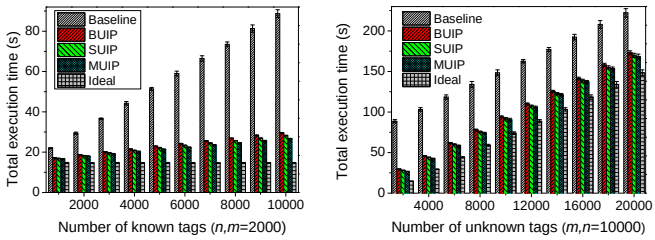
and MUIP always outperform BUIP due to their ability to deactivate known tags in the original collision/empty slots by slot pairing. Compared with BUIP, the deactivation time in SUIP and MUIP is reduced by 12 percent and 19 percent, respectively.

Fig. 7(c) plots the impact of estimation error in the number of unknown tags (m_i) on the deactivation time of MUIP. In this figure, the line marked with MUIP(opt) demonstrates the performance of MUIP when the value of m_i is exactly known, i.e., when the estimation of m_i contains no error. It can be seen that the difference between MUIP and MUIP(opt) is very small. Compared with MUIP(opt), MUIP increases deactivation time by less than 2 percent.

10.3 Total Execution Time

Fig. 8(a) plots the total execution time of our three protocols and the Baseline and the Ideal method when n changes from 1000 to 10,000, assuming that $m = 2000$. As n increases, the execution time of our protocols gradually increases, but increases much slower than the Baseline method does. When n increases from 1000 to 10,000, the execution time of the Baseline method increases 66.7s (from 22s to 88.7s). In contrast, the execution time of BUIP, SUIP, MUIP increases only 12.4s, 11.4s, and 10.1s, respectively. Compared with the Baseline method, our best protocol MUIP saves time by up to 70 percent and 55 percent in average.

The gaps between our protocols and the Ideal method increase slightly when n increases. However, when the number of unknown tags is comparable to the number of known tags, the execution time of our best protocol is only slightly longer than the Ideal method. For example, when $n = 1000$, MUIP uses only 2s more time than the Ideal method (16.7s vs. 14.7s), which accounts for only 12 percent of the total execution time of MUIP. When there are much more known tags than unknown tags, our protocols need longer time to deactivate all the known tags. For example, when $n = 10,000$, MUIP uses 26.8s to identify all the unknown tags, while the Ideal method uses only 14.7s. In this case, the deactivation time accounts for 45 percent of the total execution time of MUIP. In cases where there are much less unknown tags than known tags in the system, our best protocol would perform nearly as well as the Ideal method.



(a) n (b) m

Fig. 8. The total execution time of different protocols: (a) when n changes; (b) when m changes.

Fig. 8(b) plots the execution time of different protocols when m increases from 2000 to 20,000, assuming that $n = 10,000$. Compared with the Baseline method, our best protocol MUIP saves 70 percent time when $m = 2000$. Even when m is as large as 20,000, MUIP still saves 24 percent time compared with the Baseline method. Thus our protocols can effectively reduce identification time even when there are two times more unknown tags than known tags in the system.

10.4 Comparison with CU

The CU scheme proposed in [19] can collect a required fraction of unknown tags with probability higher than a specified threshold β. We compare the time needed for CU to collect 90%, 95% and 99% unknown tags (assuming $\beta = 0.95$) with the execution time of MUIP. The parameters in CU are determined by using the methods given in [19]. Table 2 lists the results, where CU(90%), CU(95%), and CU(99%) mean the time for CU to collect at least 90%, 95%, and 99% of unknown tags with probability higher than or equal to 0.95, respectively. We can see that the execution time of MUIP to identify all unknown tags is even almost less than the execution time of CU to collect only 90% of unknown tags.

MUIP outperforms CU due to the following reasons. First, MUIP runs much less rounds than CU because it can efficiently deactivate known tags. Recall that in

TABLE 2
Execution Time of MUIP and CU When m Changes($n = 10000$).

Alg. Name	Total Execution Time (s)									
	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
CU(99%)	35.17	41.28	50.43	59.71	68.85	78.29	87.37	96.55	101.0	109.8
CU(95%)	25.70	30.82	39.15	47.52	55.98	64.33	72.83	81.30	89.84	98.33
CU(90%)	19.23	27.07	35.01	42.96	51.06	59.26	67.26	75.24	83.52	**85.93**
MUIP	**18.65**	**26.79**	**34.78**	**42.74**	**50.70**	**58.63**	**66.54**	**74.26**	**82.53**	90.55

MUIP known tags can be deactivated with a very high probability in each round (up to 0.89 when $h = 7$), while in CU the unknown tags can be collected in each round with probability only $1 - e^{-1/1.443} = 0.5$. Second, the length of each round in MUIP is significantly shorter than that in CU. In each round of CU, the frame length is $1.443(n + m)$, where n and m are the total number of known tags and the total number of unknown tags, respectively. In contrast, in MUIP the frame length is $n_i + m_i - 1$, where n_i and m_i are the number of known tags and unknown tags in the *current round*, respectively. Third, in MUIP n_i and m_i decreases after every round because some known tags are deactivated and some unknown tags are labeled. In contrast, in CU the number of active known tags are unchanged during the execution because CU does not deactivate known tags.

When m is very large, the execution time of MUIP is dominated by the time spent in the second phase, in which case CU might use shorter time than MUIP does because it will leave a significant number of tags unidentified. For example, as shown in Table 2, CU(90%) uses less time than MUIP when $m = 10,000$, in which case it leaves about 1000 unknown tags unidentified. However, even when m is as large as 16,000, MUIP (122.8s) still uses less time than CU(95%)(128.1s) and CU(99%)(149.9s) do.

11 Conclusion

Equally important as the missing tag identification, unknown tag identification deserves more investigation in RFID systems. It is not the reverse way of missing tag identification to completely identify unknown tags. In this paper, we propose a series of protocols to perform fast and complete unknown tag identification in a large RFID systems. Simulation results show the superior performance of the proposed protocols. While in an ideal unknown identification protocol the execution time should depend on only the number of unknown tags, in the proposed protocols the execution time is still slightly impacted by known tags to some extent. In the future, we will investigate how to further reduce the impact of known tags on collecting unknown tags.

Acknowledgments

This work is partially supported by HK RGC PolyU 5281/13E, the National Natural Science Foundation of China under Grant Nos. 61373181 and 61103203, and the Fundamental Research Funds for the Central Universities. The authors would also like to sincerely thank Editors and reviewers for their thoughtful, constructive suggestions and comments.

References

[1] Vinod Namboodiri and Lixin Gao. Energy-aware tag anti-collision protocols for RFID systems. In *Proc. of PerCom*, pages 23–36, 2007.
[2] EPC Global. EPC radio-frequency identity protocols class-1 genertion-2 uhf RFID protocol for communications at 860 MHz-960 MHz Version 1.2.0, 2008.
[3] Vinod Namboodiri and Lixin Gao. Energy-aware tag anticollision protocols for RFID systems. *IEEE Transactions on Mobile Computing*, 9(1):44–59, 2010.
[4] Dheerag K. Klair, Kwan-Wu Chian, and Raad Raad. A survey and tutorial of RFID anti-collision protocols. *IEEE Communications Survery and Tutorials*, 2(3):400–421, 2010.
[5] Zongheng Zhou, Himanshu Gupta, Samir R. Das, and Xianjin Zhu. Slotted scheduled tag access in multi-reader RFID systems. In *Proc. of ICNP*, pages 61–70, 2007.
[6] Thomas F. La Porta, Gaia Maselli, and Chiara Petrioli. Anticollision protocols for single-reader RFID systems: Temporal analysis and optimization. *IEEE Transactions on Mobile Computing*, 10(2):267–279, 2011.
[7] Min Chen, Sergio Gonzalez, Victor Leun, Qian Zhang, and Ming Li. A 2g-RFID-based e-healthcare system. *IEEE Wireless Communications*, 17(1):37–43, 2010.
[8] Min Chen, Sergio Gonzalez, Qian Zhang, and Victor Leung. Code-centric RFID system based on software agent intelligence. *IEEE Intelligent Systems*, 25(2):12–19, 2010.
[9] Min Chen, Runhe Huang, Yan Zhang, and Han-Chieh Chao. A smart RFID system. In *Proc. of IWQoS*, pages 1–2, 2010.
[10] Kai Bu, Bin Xiao, Qingjun Xiao, and Shigang Chen. Efficient misplaced-tag pinpointing in large RFID systems. *IEEE Transactions on Parallel and Distributed Systens*, 23(11):2094–2106, 2012.
[11] Bin Zhen, Mamoru Kobayashi, and Masashi Shimizu. Framed aloha for multiple RFID objects identification. *IEICE Transactions*, 88-B(3):991–999, 2005.
[12] Chen Qian, Yunhuai Liu, Hoilun Ngan, and Lionel M. Ni. ASAP: Scalable identification and counting for contactless RFID systems. In *Proc. of ICDCS*, pages 52–61, 2010.
[13] Lei Xie, Bo Sheng, Chiu Chiang Tan, Hao Han, Qun Li, and Daoxu Chen. Efficient tag identification in mobile RFID systems. In *Proc. of Infocom*, pages 1001–1009, 2010.
[14] Lei Xie, Qun Li, Xi Chen, Sanglu Lu, and Daoxu Chen. Continuous scanning with mobile reader in RFID systems: An experimental study. In *Proc. of Mobihoc*, pages 1–10, 2013.
[15] Lei Yang, Jinsong Han, Yong Qi, Cheng Wang, Tao Gu, and Yunhao Liu. Season: Shelving interference and joint identification in large-scale RFID systems. In *Proc. of Infocom*, pages 3092–3100, 2011.
[16] Shaojie Tang, Jingyuan, Xiang-Yang Li, and Guihai Chen. Raspberry: A stable reader activation scheduling protocol in multi-reader RFID systems. In *Proc. of ICNP*, pages 304–313, 2009.
[17] Shaojie Tang, Cheng Wang, Xiang-Yang Li, and Changjun Jiang. Reader activation scheduling in multi-reader RFID systems: A study of general case. In *Proc. of IPDPS*, pages 1147–1155, 2011.

[18] Yanmin Zhu, Wenchao Jiang, Qian Zhang, and Haibing Guan. Energy-efficient identification in large-scale RFID systems with handheld reader. *To appear in IEEE Transactions on Parallel and Distributed Systens*, 2013.
[19] Bo Sheng, Qun Li, and Weizhen Mao. Efficient continuous scanning in RFID systems. In *Proc. of Infocom*, pages 1010–1018, 2010.
[20] Jae-Ryong Cha and Jae-Hyun Kim. Novel anti-collision algorithms for fast object identification in RFID system. In *Proc. of ICPADS*, pages 63–67, 2005.
[21] Jihoon Myung, Wonjun Lee, Jaideep Srivastava, and Timothy K. Shih. Tag-splitting: Adaptive collision arbitration protocols for RFID tag identification. *IEEE Transactions on Parallel and Distributed Systens*, 18(6):763–775, 2007.
[22] Muhammad Shahzad and Alex X Liu. Probabilistic optimal tree hopping for RFID identification. In *Proc. of Sigmetrics*, pages 293–304, 2013.
[23] Lei Kang, Kaishun Wu, Jin Zhang, Haoyu Tan, and Lionel M Ni. DDC: A novel scheme to directly decode the collisions in UHF RFID systems. *IEEE Transactions on Parallel and Distributed Systems*, 23(2):263–270, 2012.
[24] Tao Li, Shigang Chen, and Yibei Ling. Identifying the missing tags in a large RFID system. In *Proc. of MobiHoc*, pages 1–10, 2010.
[25] Chiu Chiang Tan, Bo Sheng, and Qun Li. Efficient techniques for monitoring missing RFID tags. *IEEE Transactions on Wireless Communications*, 9(6):1882–1889, 2010.
[26] Rui Zhang, Yunzhong Liu, Yanchao Zhang, and Jinyuan Sun. Fast identification of the missing tags in a large RFID system. In *Proc. of SECON*, pages 278–286, 2011.
[27] Xiulong Liu, Keqiu Li, Geyong Min, Yanming Shen, Alex X. Liu, and Wenyu Qu. Completely pinpointing the missing RFID Tags in a time-efficient way. *To appear in IEEE Transactions on Computers*, 2013.
[28] Shigang Chen, Ming Zhang, and Bin Xiao. Efficient information collection protocols for sensor-augmented RFID networks. In *Proc. of Infocom*, pages 3101–3109, 2011.
[29] Hao Yue, Chi Zhang, Miao Pan, Yuguang Fang, and Shigang Chen. A time-efficient information collection protocol for large-scale RFID systems. In *Proc. of Infocom*, pages 2158–2166. IEEE, 2012.
[30] Yan Qiao, Shigang Chen, Tao Li, and Shiping Chen. Energy-efficient polling protocols in RFID systems. In *Proc. of MobiHoc*, page 25, 2011.
[31] Yuanqing Zheng and Mo Li. Fast tag searching protocol for large-scale RFID systems. *To appear in IEEE/ACM Transactions on Networks*, 2013.
[32] Min Chen, Wen Luo, Zhen Mo, Shigang Chen, and Yuguang Fang. An efficient tag search protocol in large-scale RFID systems. In *Proc. of Infocom*, pages 899–907, 2013.
[33] Su-Ryun Lee, Sung-Don Joo, and Chae-Woo Lee. An enhanced dynamic framed slotted aloha algorithm for RFID tag identification. In *Proc. of MobiQuitous*, pages 166–174, 2005.
[34] Harald Vogt. Efficient object identification with passive RFID tags. In *Proc. of Pervasive*, pages 98–113, 2002.
[35] Murali S. Kodialam and Thyaga Nandagopal. Fast and reliable estimation schemes in RFID systems. In *Proc. of Mobicom*, pages 322–333, 2006.
[36] Chen Qian, Hoilun Ngan, Yunhao Liu, and Lionel M. Ni. Cardinality estimation for large-scale RFID systems. *IEEE Transactions on Parallel and Distributed Systens*, 22(9):1441–1454, 2011.
[37] Hao Han, Bo Sheng, Chiu Chiang Tan, Qun Li, Weizhen Mao, and Sanglu Lu. Counting RFID tags efficiently and anonymously. In *Proc. of Infocom*, pages 1028–1036, 2010.
[38] Yuanqing Zheng and Mo Li. PET: Probabilistic estimating tree for large-scale RFID estimation. *IEEE Transactions on Mobile Computing*, 11(11):1763–1774, 2012.

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.

Bin Xiao received the BSc and MSc degrees in electronics engineering from Fudan University, China, in 1997 and 2000, respectively, and the PhD degree in computer science from the University of Texas at Dallas in 2003. After his PhD graduation, he joined the Hong Kong Polytechnic University as an assistant professor. Currently, he is an associate professor in the Department of Computing at The Hong Kong Polytechnic University, Hong Kong. His research interests include mobile cloud computing, data management, network security, wireless sensor networks and RFID systems. He is an 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 a recipient of the Best Paper Award of IEEE/IFIP EUC 2011. He is a senior member of the IEEE.

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 is a member of IEEE and CCF.

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 (second author). He is a member of the ACM and the IEEE.