IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 23, 2021, accepted June 5, 2021, date of publication June 18, 2021, date of current version June 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3090438

Fast RFID Tag Sorting at the Edge for Internet of Things

YANGZHAO YANG[1] AND XIUJUN WANG[2,3,4]

[1]Shenzhen Cyberaray Network Technology Company Ltd., Shenzhen 518000, China
[2]School of Computer Science and Technology, Anhui University of Technology, Ma'anshan 243032, China
[3]Institute for Artificial Intelligence, Hefei Comprehensive National Science Center, Hefei 230091, China
[4]Anhui Engineering Laboratory for Intelligent Applications and Security of Industrial Internet, Ma'anshan 243032, China

Corresponding author: Xiujun Wang (wxj@mail.ustc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61402008, in part by the University Natural Science Research Project of Anhui Province under Grant KJ2020A0249, in part by the Provincial Key Research and Development Program of Anhui Province under Grant 202004a05020009 and Grant 201904a05020071, in part by the Open Fund of Key Laboratory of Anhui Higher Education Institutes under Grant CS2020-006, in part by the Electronic Information and Control of Fujian University Engineering Research Center, Minjiang University, under Grant MJXY-KF-EIC1803, and in part by the Program for Synergy Innovation in the Anhui Higher Education Institutions of China under Grant GXXT-2020-012.

ABSTRACT Internet of Things (IoT) has gained great popularity in various fields including smart warehouse and intelligent manufacturing. As a building block of IoT network, the Radio Frequency IDentification (RFID) technology enables a large and ever-increasing number of physical objects to be monitored across the Internet via tag identification. Efficiently managing massive tags in RFID systems becomes an important research issue for IoT networks. This paper focuses on a fundamental management problem — tag-sorting, which is to (1) put a set S of identified tags into a certain order by informing each tag $t \in S$ of a unique integer $\Delta(t) \in \{1, 2, \cdots, |S|\}$, and meanwhile (2) keep unidentified tags from receiving any of these integers. For RFID systems, it is critical to solve this problem as quickly as possible in the sense that, once sorted, every identified tag $t \in S$ can be manipulated via t's $\log_2(|S|)$-bit integer significantly shorter than t's 96-bit long tag-ID ($\log_2(|S|) \ll 96$), boosting efficiency substantially. The existing works of literature, however, fails to solve this problem rapidly, as they accomplish (1) and (2) separately by using aloha-like protocols and Bloom filters, which incur a long communication time far from the optimum. In this paper, we overcome this drawback by proposing a protocol P_{sort} capable of solving the problem fastly. In particular, this protocol is built with a novel data structure and communication scheme to achieve (1) and (2) simultaneously by using a communication time proven to be much less than the state-of-the-art protocols. The simulation results demonstrate the competence of P_{sort} in achieving about 1.4× speedup than the state-of-the-art solutions.

INDEX TERMS IoT network, edge server, RFID systems, tag management, tag-sorting, unidentified tag, identified tag.

I. INTRODUCTION

Over the past decades, Internet of Things (IoT) has been used in many fields including smart warehouse management [1]–[3], traffic monitoring [4]–[6], and logistical tracking [7]–[9]. In the construction of IoT networks, RFID technology serves as a building block that connects millions and billions of physical objects into the Internet through identifying the RFID tags affixed to objects [10], [11]. This enables massive physical objects to be swiftly sensed and controlled across the IoT network [12], and creates the opportunity for efficient management and accurate analysis made towards these objects [4], [13]–[18].

Recently, as the number of tags attached to objects gets increased dramatically [3], [12], [19], [20], how to rapidly manage these massive tags has become an important research issue in RFID systems for IoT networks. In particular, this issue is critical when massive sensed data from tags are flooded into edge servers where they must be timely processed and analyzed to provide valuable intelligence to real-time applications [3]–[8], [21], [22]. For example, consider the RFID systems used in cold-chain storage and

The associate editor coordinating the review of this manuscript and approving it for publication was Vyasa Sai.

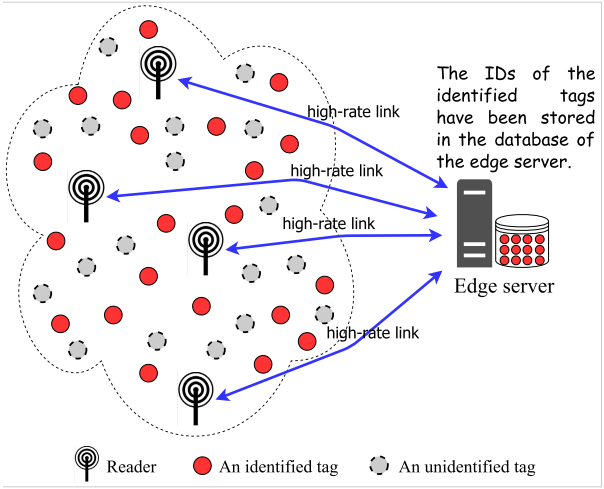

FIGURE 1. An example of the RFID systems for IoT networks.

transportation facilities, where sensor-augmented RFID tags [23] are attached to food items and track their temperature in real-time. In this scenario, allowing all tags to send back their sensed data without control will lead to a high transmission latency and a large volume of redundant/useless information. Therefore, to ensure food quality and safety, a smart temperature surveillance application deployed in these systems must control and schedule tags to send their data appropriately so that those tags (attached to food items) with abnormal temperatures can be detected timely. More example can be found in [3]–[5], [7], [20], [23]–[27].

Generally speaking, an RFID system for IoT networks [1]–[8] contains three key components: an edge server, multiple readers, and massive tags. The edge server communicates with readers (RFID readers) over a high-rate link and provides both computational and storage support for readers. The server also schedules readers and tags to work together appropriately, prevents communication collisions, and sends local analysis results or sensed data back to the central cloud if required. In contrast to the high-speed link between the server and readers, a reader monitors the tags (RFID tags) within an area via a low-rate link through whose communication volume mainly determines protocols' communication cost in RFID systems. A tag is composed of an antenna and a microchip. It carries a 96-bit or 128-bit ID to uniquely characterize itself as well as the attached physical object (the object that this tag is affixed to). When interrogated by a nearby reader, a tag shall backscatter the stored ID to this reader for showing its existence. A tag may also transmit other information related to the attached object back to the reader if required [3], [4], [20], [23]–[26], [28]–[30]. If the 96-bit or 128-bit ID of a tag t has been collected by a nearby reader (t has been recognized by this reader), t is called as an identified tag (the system knows t's existence); otherwise, t is called an unidentified tag. In Fig. 1, we depict an example of the RFID systems for IoT networks.

This paper primarily focuses on the tag-sorting problem which is a fundamental management problem in the RFID systems for IoT networks. Specifically, given a set S of n identified tags, and a set O of an unknown number of unidentified tags, this problem requires to quickly (1) put the n identified tags into a certain order by informing them with unique integers from $\{1, 2, \cdots, n\}$, and meanwhile (2) keep unidentified tags from receiving these integers. Hereafter, we call (1) and (2) the two objectives of the tag-sorting problem.

The studied problem has a natural and deep connection to efficiently managing massive tags in RFID systems, and thus greatly affects system efficiency. **The reason is that, once the tags in an RFID system are sorted, every identified tag t can be manipulated (selected, read and written) via the short unique integer assigned to t, substantially reducing the communication time for managing massive tags. This aspect is of particular importance for RFID systems with massive tags at the edge of IoT networks [1], [2], [7], [20], [21], [23] and thus motivates our research in this paper.** Specifically, given an RFID system with a set S of n identified tags and a set O of unidentified tags, if these tags are sorted (i.e., each identified tag in S is informed with a unique integer $\Delta(t) \in \{1, 2, \ldots, n\}$, while unidentified tags in O are not), we can enjoy three typical benefits as below.

- An RFID reader can quickly select an identified tag t from S by sending out t's integer $\Delta(t)$, a $\log_2(n)$-bit string substantially shorter than t's 96-bit or 128-bit ID ($\log_2(n) \ll 96$) [10], [24].
- An RFID reader can rapidly select a tag subset T from S ($T \subset S$) by sending out the unique integers of the tags in T instead of their 96-bit IDs [8], [10], [25].
- An RFID reader can swiftly read or write data from or to every identified tag in S by following either an increasing or decreasing order of tags' integers [10], [20], [26].

All these bring great conveniences to managing massive tags. Some concrete examples are listed to appreciate the practical significance of this problem. First, considering a supermarket, the manager needs to periodically and quickly find out those identified tags that went missing, such that the loss or theft of items can be caught or even prevented. In this case, it is easy to see that a fast tag sorting protocol helps solve the problem, since it can separate out unidentified tags before checking the existence of remaining identified tags one-by-one following an increasing order of tags' integers. Second, in a sensor-augmented RFID system, tags equipped with sensors can monitor their surroundings and then periodically feed data back to readers for analysis. During each time of data collection, only a subset of the identified tags is usually required to send back their data, while the rest are not. Clearly, a fast tag-sort protocol can speed up the data collection process, as a tag subset can be rapidly selected by their assigned integers. Third, considering a large local logistic center, tags are constantly being shipped to different destinations. In this scenario, the transportation company needs to write on each identified tag some information, e.g., a destination address, shipper code, insurance policy, for live

tracking or querying purposes. Also, an efficient tag-sorting protocol makes it simple to build a fast-writing process, since every identified tag bears a unique integer, while unidentified tags do not. More practical examples can be found in [3], [4], [6], [8], [21], [26], [29], and [32]..

With the above discussions, it is not hard observe that a fast tag-sorting protocol is beneficial to the goods monitoring applications deployed in supermarkets and warehouses [2], [22], [25], [26]. Because this protocol can rapidly deactivate irrelevant (unidentified) tags and get relevant (identified) tags more easily manipulated with the assigned integers, each of which is significantly shorter than the original 96-bit IDs of these tags. Environmental data collection applications dealing with sensor-augmented tags also benefit from a fast tag-sorting protocol. Environmental data collection applications dealing with sensor-augmented tags also benefit from a fast tag-sorting protocol [8], [9], [23]. The explanation is: once tags are sorted, the reader can more quickly isolate a tag subset from a large population by using the assigned short integers than using the original 96-bit IDs. For the same reason, a tag sorting protocol also helps some applications that needs to store the same piece of information in a tag subset for tracking or security reason [14], [20], [24], [29], [30] (since tag subsets can be more quickly selected with these assigned integers after all tags are sorted).

A. LIMITATIONS OF PRIOR WORK

Despite the broad significance of the tag-sorting problem in RFID systems, to our best knowledge, there is no protocol specifically designed to solve this problem. A naive solution would be to let the reader predetermine a unique integer $\Delta(t) \in \{1, 2, .., n\}$ for each identified tag $t \in S$, and then broadcast the 96-bit or 128-bit IDs of the tags in S by following either an increasing or decreasing order of tags' integers: $\Delta(t), t \in S$. This solution, however, yields a high communication cost of $96 \times n$ bits, and then an unacceptable latency for real-time applications, especially when there are massive identified tags (n is large). For saving the communication cost, we may first separate identified tags from unidentified ones by using the classic Bloom filter and then adopt an Aloha-like protocol to assign each identified tag a unique integer. Unfortunately, this scheme still cannot bring the communication time down to a satisfactory level, because both Bloom filter and aloha-like protocols demand a considerable amount of communication cost which is far from the optimum.

Other research works investigating problems related to the tag-sorting problem are analyzed below.

There are some researches [24], [29], [32]–[34] devoted to rapidly collecting tag information in RFID systems. For example, in [32], Chen *et al.* studied how to collect the information from a whole tag population. They proposed the Multi-hash Information Collection method (MIC) that employs multiple hash functions to resolve hash collisions in each communication round to boost the chance that a tag can successfully send back its information. With this strategy, more tags can send their information back in one round, and thus MIC shows an improved time efficiency than the Single-hash based approaches. The authors in [33] studied how to collect the information from a subset S of identified tags instead of the whole population of tags. They designed the Enhanced Tag-Ordering Polling method (ETOP) that utilizes the classic Bloom filter for separating those tags in subset S from other tags and then polls their information. Liu *et al.* [34] proposed a tree-based polling protocol (TPP) which collects tag information by polling tags' hashed indexes and reduces the communication cost by reducing the length of polling vectors. Reference [29] studied how to use minimal perfect hashing to collect the information of a tag subset. For this purpose, they proposed a protocol called the minimal Perfect hashing-based Information Collection (PIC) which uses multiple indicate vectors to recursively appoint one slot solely to one tag, and then filters out other non-target tags.

There are several studies [2], [35]–[37] that deal with detecting missing tags or unidentified tags in RFID systems. For instance, Zhang *et al.* [2] focused on quickly finding out missing tags in mobile RFID systems. They designed a protocol called Efficient Bit-Detecting (EBD), which assumes that each tag has been assigned with a unique reading order, and then uses a single bit-array to detect the presence of missing tags. Reference [35] studied how to quickly detect missing tags in the presence of unknown tags. They designed a missing tag detection protocol that uses a compressed Bloom filter to reduce the communication cost for pinpointing missing tags.

In summary, however, when applying these existing protocols to the tag-sorting problem, they suffer from a long communication delay, because they handle the two objectives of the tag-sorting problem separately.* Specifically, given an RFID system with n identified tags and an unknown number of unidentified tags, the communication costs of the existing protocol are lower bounded by $1.44\, n \log_2(1/\varepsilon) + 2.71n$ (see Theorem 5). The first subformula $1.44\, n \log_2(1/\varepsilon)$ is the communication cost of using a Bloom filter to keep unidentified tags from receiving any integers with an error probability ε [38]; The second subformula $2.71n$ is the communication cost for informing each identified tag of a unique integer [29], [39]. We will explain in the following sections how we break this upper limit by handling the two objectives simultaneously.

Please note that, there are a number of recent works [40]–[45] that use the term ''tag sorting'' to represent the RFID-based localization problem, which is to acquire the location information of different tags and find the spatial order of these tags. In contrast, the tag-sorting problem studied here is to assign each identified tag a new and unique integer regardless of their locations and spatial order. These works are therefore not discussed.

* Recall that the sorting problem strictly requires that unidentified tags shall not receive unique integers that are meant for identified tags only.

B. TECHNICAL CHALLENGES AND CONTRIBUTIONS

There are two technical challenges to address when designing a fast tag-sorting protocol.

- **How to rapidly separate the identified tags from unidentified tags.** Unidentified tags can be filtered out simply if we use a Bloom filter to encode the set S of identified tags and then send this filter out. However, a Bloom filter and its variant structures are not optimal in terms of memory space, thus still demands a considerable amount of communication cost [38]. Hence, we need to design a new data structure capable of smartly encoding the information of identified tags with as few bits as possible. This point is indeed challenging, as the following sections reveal our great efforts towards fulfilling this goal.
- **How to rapidly inform identified tags of unique integers.** When assigning integers to tags, people usually think of the traditional aloha-like protocols [24], [29], [32]–[34], [37] which randomly distribute tags to different slots, and then assign different integers to different tags by using the singleton slots (a slot is called a singleton slot if there is exactly one tag mapped to this slot). Although this scheme is popular in RFID systems, it still suffers from low communication efficiency because a large proportion of transmitted slots are not singleton slots and thus unused. To be more specific, on average, about $1-e^{-1}$ of the slots in these aloha-like protocols are not singleton [29]. Therefore, we need to design a new scheme that boosts communication efficiency by reducing the number of useless slots. In fact, we achieve this point by reusing these useless slots to deactivate unidentified tags (separating unidentified tags from identified ones). This point is challenging, because, as far as we know, it has never been touched by other research works, and it requires much effort to utilize those slots which appear to have no functionality to deactivate unidentified tags.

Briefly, the core contribution of this paper compared to these existing works contains two elements as below.

(1) **We have proposed a protocol that solves the tag-sorting problem in a way that the two objectives of this problem can be achieved simultaneously with much less communication time.** Remember that given an RFID system with both identified and unidentified tags, the tag-sorting problem has two objectives of (1) informing each identified tag of a unique integer and (2) keeping unidentified tags from receiving these integers.

(2) **We have examined the proposed protocol with rigorous theoretical analysis and proven that the proposed protocol requires much less communication time compared to the existing standard protocols.** More specifically, the proposed protocol reduces the communication time by 30% as compared to existing state-of-the-art protocols.

The rest of the paper is organized as follows. Section II introduces the system model, problem definition, and assumptions. Section III presents the designed tag-sorting protocol, followed by a theoretical analysis of its performance. Section IV conducts extensive simulations for comparing the proposed protocol with the other existing protocols. Section V presents the conclusion of this paper.

II. DEFINITION AND ASSUMPTIONS

A. SYSTEM MODEL AND PROBLEM DEFINITION

We consider an RFID system consisting of an edge server, a reader R, and a number of tags each of which is attached to a physical object. The reader R fully covers all the tags and connects an edge server via a high-speed link from which it can get storage and computational support quickly. Each tag t carries a unique 96-bit ID by which we can identify both the tag t and the associated physical object. Tag t can perform certain simple computations. Moreover, if the ID of tag t has already been recognized by R, t is called as an identified tag; otherwise, t is an unidentified tag. We denote the set of identified tags by $S = \{t_1, t_2, \cdots, t_n\}$ and the set of unidentified tags by $O = \{u_1, u_2, \cdots\}$.

The communication between reader R and tags follows the classic framed slotted Aloha (FSA) scheme which is given by the EPC Global C1G2 standard [10]. This scheme has been used widely by a large number of tag management protocols working in RFID systems [2], [24], [30], [35]–[37]. To be more specific, this scheme contains multiple rounds, and each one works as follows.

- First, the reader R starts a communication round by sending out the necessary parameters including a random seed and a frame size.
- Second, every tag, upon receiving a request, performs some calculations, and then either sends back some information or waits to receive further messages from readers.

In this paper, we take the communication cost of a protocol to be primarily determined by the number of the transmitted bits between reader R and tags, which are sent over a low-speed link [10].

With the above notations, it gets ready to define the tag-sorting problem as below.

Definition 1: Let ε be a user-specific error probability. Given an RFID system with a reader R, a set S of n identified tags, and a set of unidentified tags, the tag-sorting problem requires to design a protocol running on reader R and tags, such that the following two requirements can be satisfied.

(1) $\forall t \in S$, t is informed of a unique integer from $\{1, 2, \cdots, n\}$;

(2) $\forall u \in O$, the probability that u is incorrectly informed of (or incorrectly receives) an integer $I \in \{1, 2, \cdots, n\}$ is no more than ε.

In Fig. 2, we show an example of the tag-sorting problem in an RFID system with 5 identified tags and 5 unidentified tags.

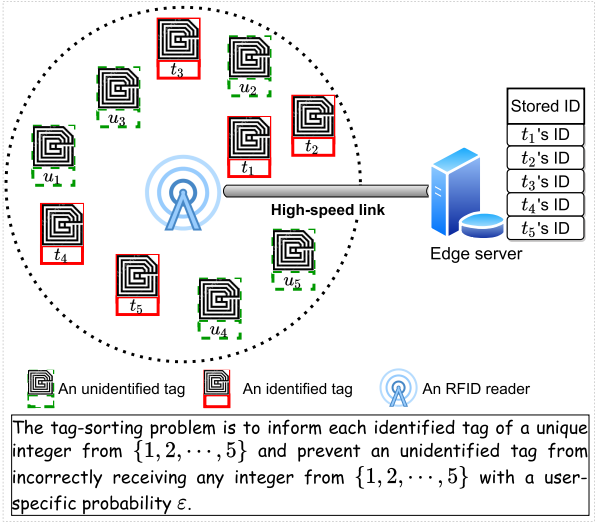

FIGURE 2. An example of the tag-sorting problem.

Please note that the problem definition and the proposed scheme in this paper can be easily extended to the multi-reader scenarios, when readers are well scheduled to work jointly by the edge server, which is usually true in practice [1]–[3], [24], [30], [35]–[37]. There already exist a number of efficient methods [46]–[48] that schedule the readers in RFID systems to work jointly well and free of collisions. It should also be noted that the goal of the tag-sorting problem studied in this paper is to assign each identified tag a single, new, and unique integer that does not take into account their locations or spatial orders. Thus, it is different from the RFID-based localization problem that uses "tag sorting" term [40]–[45] to represent finding the spatial order of the tags.

Based on Definition 1, it is clear that the main sources of communication overhead in the tag-sorting problem consist of two parts: the transmission cost for achieving the objective (1) and the transmission cost for achieving the objective (2). In the following section, we show how to overcome this overhead by proposing a novel data structure and communication scheme which can achieve the two objectives jointly. In particular, the novel data structure can smartly encode the information of identified tags with a minimum number of bits (much less than the classic Bloom filter) to satisfy objective (2); the data structure also helps the related communication scheme to satisfy objective (1) by using a much-reduced number of communication slots than aloha-like protocols.

B. ASSUMPTIONS

Besides the above definition, we adopt the following two assumptions in this paper.

We assume that the reader can transmit messages to all tags using the same rate (e.g., 26.5Kbit/s), but does not consider the difference in the signal strength of RFID tags. Three reasons account for this assumption. First, the maximum interrogating range of an RFID reader is limited to the distance where the passive tags' chips can receive enough power to turn on themselves (the range is usually less than 10 meters) [26], [49]. Second, most RFID systems are installed in supermarkets, warehouses or on the conveyor belts, where the communication condition of every tag within the reader's range are almost the same [2], [8], [9], [24], [26], [29], [30], [32]–[37], [50]. Note that, in typical RFID systems, tags communicate and power themselves using same RF waves radiated by the reader antenna [10], [51]. In other words, the reader sends its energy through the antenna as RF waves to tags in order for them to become energized and respond back to the reader. Third, this transmission setting assumption has been widely used to evaluate the performance of various promising tag management protocols in RFID systems [2], [9], [24], [29], [30], [32]–[35], [37], [50], [52].

Since this paper primarily focuses on managing RFID tags with as little communication time as possible and mainly solves the tag-sorting problem, we do not concern the security issues on tags. There could be potential security risks to the proposed protocol since the new identifiers assigned to tags come from a fixed set of integers, making them relatively easy for malicious attackers to guess and exploit. However, please note that the security feature of tags is largely preserved since the proposed protocol does not make any changes to the Kill password and Access password stored on each tag. Note that the Access password allows a user to lock a tag, preventing modification of various parts of the memory (EPC, User, etc.); the Kill password is needed to disable a tag [10]. We do think the security issue is vital and will consider this with more details in our future.

III. THE DESIGN OF P_{SORT}

The designed protocol P_{sort} is a two-phase protocol. The first phase assigns unique integers to a large fraction of the identified tags in set S (a faction of $1 - \varepsilon^{1.6}$ of them) and meanwhile separates unidentified tags from identified tags with a user-specific probability ε. Please note that the unique integer $\Delta(t)$ assigned to tag t is called the order of tag t throughout this paper. The second phase finishes the rest of the job by assigning the other tags in S (a faction of $\varepsilon^{1.6}$ of them) with unique integers directly by pooling their IDs.

In P_{sort}, we use three states to characterize the current status of a tag.

- Tag t is in **unsorted state**, if t doesn't know whether it is an identified or unidentified tag or not, i.e., has no idea about whether it belongs to S or O.
- Tag t is in **sleep state**, if t knows for sure that it is an unidentified tag; A tag in sleep state keeps in this state until the end of P_{sort}.
- Tag t is in **sorted state**, if t has determined to be an identified tag and has obtained a unique order $\Delta(t)$.

Initially, every tag t in this RFID system is in the unsorted state, and maintains a local variable **G** which is initialized to 0 to keep tracking of the number of tags that have entered the sorted state.

A. THE FIRST PHASE OF P_{SORT}

The first phase of P_{sort} has $f = \lceil \log_2(\frac{1}{\varepsilon}) \rceil$ communication rounds $\mathbb{R}_1, \mathbb{R}_2, .., \mathbb{R}_f$, each of which puts a number of unidentified tags into the sleep state and a number of identified tags into the sorted state. To be more specific, by the end of the first phase (after l rounds), (1) a fraction of about $1 - \varepsilon^{1.6}$ of identified tags will be assigned with unique integers from $\{1, 2, \cdots, n\}$ and put to the sorted state; (2) a fraction of $1 - \varepsilon$ unidentified tags enter the sleep state without getting any integer.

For ease of illustration, let S_k denote the set of identified tags that are in the unsorted state and n_k denote its size ($n_k = |S_k|$), before the start of the k-th round $\mathbb{R}_k$ ($k = 1, 2, \cdots, f$)). Thus, $S_1 = S$, $n_1 = n$, initially. Additionally, without loss of generality, $c = \log_2(\frac{n_k}{\ln 2})$ is assumed to an integer in each round $\mathbb{R}_k$.

The basic idea of each round $\mathbb{R}_k$ is: the reader R constructs two arrays: B and D which will be used to determine unique orders for some identified tags, and put some unidentified tags to sleep, such that any unidentified tag enters the sleep state with probability ε. Detailed steps of $\mathbb{R}_k$ ($k = 1, 2, \cdots, f$) are shown below.

Step 1: The reader R broadcasts two parameters: n_k and a random number r for reader R as well as every unsorted tag t to compute a hash function $h(t) = H(t_{ID}, r) \mod 2^{c+f+1-k}$, where $H()$ is a hash function and t_{ID} is the 96-bit ID of tag t. The value of $h(t)$ is a binary number of $(c + f + 1 - k)$ bits: $h(t) = \mathbf{v}[0, 1, \cdots, c + f - k]$. The bit sequence in $\mathbf{v}$ is further divided into three disjoint segments, as shown in Eq. (1).

$$\begin{aligned} v_t^1 &= \mathbf{v}[0, 1 \cdots, c-1] \ \ (c \text{ bits}), \\ v_t^2 &= \mathbf{v}[c] \ \ (1 \text{ bit}), \\ v_t^3 &= \mathbf{v}[c+1, c+2, \cdots, c+f-k] \ \ (f-k \text{ bits}). \end{aligned} \tag{1}$$

These three segments are used for the following purposes, respectively. (1) The binary number of v_t^1 points to the location in array B where tag t is mapped to; (2) the single bit v_t^2 is used for resolving hash collisions, if exactly one another identified tag is mapped to the same position in B; (3) v_t^3 is used as the fingerprint of tag t and will be stored in a corresponding cell of array D. More explanations will be provided in the following steps.

Step 2: Upon receiving n_k and r, each tag t in unsorted state computes $h(t) = (v_t^1, v_t^2, v_t^3)$ using its own ID t_{ID}.

Step 3: Reader R also produces the hash value $h(t) = (v_t^1, v_t^2, v_t^3)$ for each tag $t \in S_k$.

Step 4: Reader R constructs 2^c cells for array B ($B[0, 1, \cdots, 2^c - 1]$), where each cell $B[i] (0 \le i \le 2^c - 1)$, is a 1-bit, or 2-bit, or 3-bit string determined by rules of (2).

$$B[i] = \begin{cases} \text{'0'} & \text{if } |\{z \in S_k | v_z^1 = i\}| = 0; \\ \text{'10'} & \text{if } |\{z \in S_k | v_z^1 = i\}| = 1; \\ \text{'110'} & \text{if } |\{z \in S_k | v_z^1 = i, v_z^2 = 0\}| = 1 \\ & \quad \text{and } |\{z \in S_k | v_z^1 = i, v_z^2 = 1\}| = 1; \\ \text{'111'} & \text{otherwise}. \end{cases} \tag{2}$$

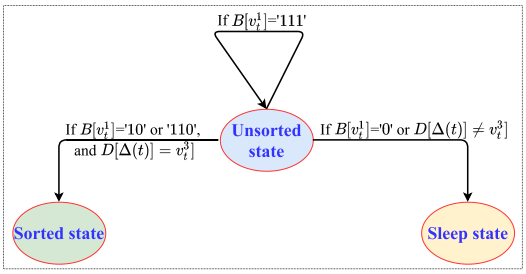

FIGURE 3. State diagram of a tag t in the first phase of P_{sort}.

Let N_t^{10} and N_t^{110} denote, respectively, the number of cells of '10' and '110' that occur before cell $B[v_t^1]$. Let N^{10} and N^{110}, respectively, be the total number of cells of '10' and '110' among all 2^c cells in B. Similarly, we can define N_t^{111}, N^{111} and N^0. These notations are summarized in (3).

$$\begin{cases} N_t^b = |\{B[i] = \text{'}b\text{'}, i = 0, 1, \cdots, v_t^1 - 1\}|, \\ \quad \textit{where } b \in \{\text{'10'}, \text{'110'}, \text{'111'}\}. \\ N^b = |\{B[i] = \text{'}b\text{'}, i = 0, 1, \cdots, 2^c - 1\}|, \\ \quad \textit{where } b \in \{\text{'0'}, \text{'10'}, \text{'110'}, \text{'111'}\}. \end{cases} \tag{3}$$

Based on Eq. (2), the following two facts are true.

- ⋆ If $B[v_t^1] =$ '10', then tag t is the only identified tag mapped to this cell. Note that we will not count an unidentified tag mapped to the same cell.
- ⋆ If $B[v_t^1] =$ '110', then exactly two identified tags are mapped to the same cell $B[v_t^1]$. Thus, in addition to tag t, there exists another unique tag $t' \in S_k$ mapped to $B[v_t^1]$ also. This collision can be resolved by the fact that $v_t^2 \neq v_{t'}^2$. Again, unidentified tags are not counted.

Step 5: Consider the set: $L_k = \{t \in S_k | B[v_t^1] =$ '10' *or* '110'$\}$, whose tags can be uniquely identified by above observations. It is easy to see that the number of tags in L_k is $N^{10} + 2N^{110}$. For example, in Fig. 4, $N^{10} = 3$ ($B[1] = B[2] = B[6] =$ '10') and $N^{110} = 1$ ($B[7] =$ '110'). Thus $L = \{t_3, t_4, t_2, t_7, t_5\}$. Note that both t_5 and t_7 map to $B[7]$ but they differ in bit v_t^2 ($v_{t_5}^2 \neq v_{t_7}^2$), as shown in Tab. 1. Thus, in this step, reader R constructs array $D[0, 1, \cdots, N^{10} + 2N^{110} - 1]$ that has $N^{10} + 2N^{110}$ cells, one for each tag in set L_k. The position in D for tag t is $\Delta(t)$ determined by Eq. (4). The value of $D[i], i \in \{0, 1, \cdots, (N^{10} + 2N^{110} - 1)\}$ is a $(f - k)$-bit long string ($f = \log_2(1/\varepsilon)$) which is copied from v_t^3, $D[i] = D[\Delta(t)] = v_t^3$, called fingerprint of tag t.

$$\Delta(t) = \begin{cases} N_t^{10} & \text{if } B[v_t^1] = \text{'10'} \\ N^{10} + 2N_t^{110} + v_t^2 & \text{if } B[v_t^1] = \text{'110'}. \end{cases} \tag{4}$$

Step 6: After the construction of B and D, reader R broadcasts array B and D out to all tags. This step needs a lengthy explanation which is discussed in section III-B.

Step 7: Upon receiving B and D, a tag t shall make decisions based on $B[v_t^1]$ and $D[\Delta(t)]$ as follows:

(7.1) If $B[v_t^1] =$ '0', tag t enters into the deactivated state, because by Eq. (2), no tag in S_k maps to $B[v_t^1]$. Tag t must be an unidentified tag;

TABLE 1. An example of the hash values for 7 tags.

hash values \ tags	t_1	t_2	t_3	t_4	t_5	t_6	t_7
$v_t^1(decimal)$	4	6	1	2	7	4	7
$v_t^2(decimal)$	1	0	0	1	1	1	0
$v_t^3(binary)$	'110'	'010'	'111'	'011'	'000'	'101'	'110'

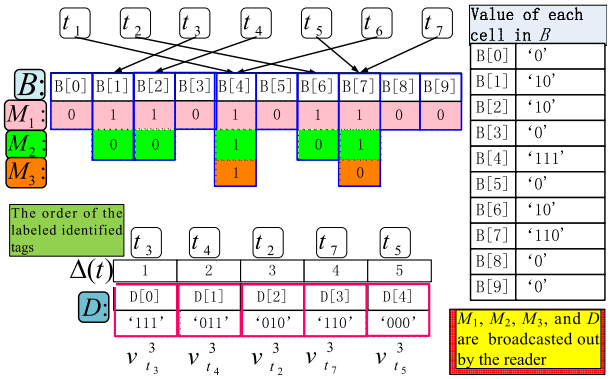

FIGURE 4. An example of *B* and *D* according to Table 1.

(7.2) If $B[v_t^1]$ = '10' *or* '110', then if $D[\Delta(t)] = v_t^3$, then tag t gets a unique order $\Delta(t) = \Delta(t) + \mathbf{G}$, and enters into the sorted state. Otherwise, t must be an unidentified tag and enters into the sleep state.

(7.3) If $B[v_t^1]$ = '111', t updates $\mathbf{G} = \mathbf{G} + N^{10} + 2N^{110}$ and stays in unsorted state waiting for the start of next round $\mathbb{R}_{k+1}$.

Step 8: Reader R sets $S_{k+1} = S_k - L_k$, and starts the next round $\mathbb{R}_{k+1}$.

The state diagram of a tag is as in Fig. 3. Clearly, by **(7.2)**, every tag $t \in L_k$ enters into the sorted state.

B. HOW ARRAY B AND D ARE BROADCASTED AND DECODED

In this section, we explain in detail how to implement **Step 6** in each round of the first phase of P_{sort} discussed in Section III-A.

Procedure 1 shows a pseudo-code on how reader R broadcasts arrays B and D. The basic idea is to send B in a bit-wise way. That is, in the first step, reader R transmits a sequence (M_1) of 2^c bits, which consists of exactly one bit (the first bit) from each cell of B in order. In the second step, reader R transmits a sequence (M_2) of bits which consists of the second bit from each of those cells of B whose values are of 2-bit string or 3-bit string. They are also the cells whose first bits are '1's. The third step transmits M_3 which consists of the third bit from each of those cells whose values are of 3-bit strings. They are also those cells whose both first and second bits are '1's. Since the code of each cell is either a 1-bit, or 2-bit, or 3-bit string, array B can be transmitted in 3 steps. Recall that, here, the index of the 3 sequences starts from 0. As an example, suppose we have a set of 7 identified tags: $S = \{t_1, t_2, \cdots, t_7\}$, and their hash values are shown in Tab. 1.

Procedure 1 How Reader R Broadcasts B and D

For $k = 1$ **to** $f = \log_2(\frac{1}{\varepsilon})$

||* In each round $\mathbb{R}_k$, R considers only the tags in S_k, which is the set of identified tags that are in the unsorted state. *||

1: Based on two parameter n_k (the size of S_k) and r (a random number), reader R computes $h(t)$ for $t \in S_k$ to construct a B and a D for round $\mathbb{R}_k$; ||*see **Step 4** and **Step 5** in Section III-A for the details *||

2: R broadcasts n_k and r out to tags;

3: R builds M_1, a bit-array that contains the first bits in all the cells of B;

4: R builds M_2, a bit-array that contains the second bits in all the cells of B; ||*the cells with value $'0'$ are omitted, since they contain only one bit.*||

5: R builds M_3, a bit-array that contains the third bits in all the cells of B; ||*the cells with value $'0'$ and $'10'$ are omitted, since they do not contain 3 bits.*||

6: R broadcasts M_1, M_2, M_3, and then D out;

7: R updates S_k to be S_{k+1}; **End For**

Fig. 4 shows that reader R constructs array B of 10 cells according to Eq. (2). Thus, reader R shall broadcast three bit arrays, $M_1 = (0, 1, 1, 0, 1, 0, 1, 1, 0, 0)$ (shown in pink color), $M_2 = (0, 0, 1, 0, 1)$ (green color), and $M_3 = (1, 0)$ (orange color).

In Procedure 2, we describe how an identified tag t extracts the codeword of $B[v_t^1]$ from M_1, M_2 and M_3, and then calculates its order $\Delta(t)$. Note that $I_t = N_t^{10} + N_t^{110} + N_t^{111}$ in line 3 is the position of the second bit of $B[v_t^1]$ in M_2, because only those cells with value $b \in$ {'10', '110', '111'} have their second bits transmitted in M_2. For example, in Fig. 4, t_5 is hashed to the cell $B[7]$ = '110' ($v_{t_5}^1 = 7$, see Tab. 1), and the second bit of $B[7]$ is sent in $M_2[4]$. An important fact is that tag t can get the value of $(N_t^{10} + N_t^{110} + N_t^{111})$ by counting how many '1's in M_1 locating before $M_1[7]$. Similarly, after receiving the second bit, a tag can determine whether its code has the third bit and if yes, where to find it in M_3 (see line 6 of Procedure 2).

The transmission of array D is straightforward. Reader R transmits array D cell by cell, from $D[0]$ to $D[N^{10} + 2N^{110} - 1]$, where $N^{10} + 2N^{110}$ is the size of set $L_k = \{t \in S_k | B[v_t^1] =$ '10' *or* '110'$\}$. Since each cell consists of $(f - k)$ bits, reader R transmits a total of $(f - k)(N^{10} + 2N^{110})$ bits in this step.

Taking t_2 in Fig. 4 as an example to see how a tag decodes array B and D to determine its state and determine if it gets a unique order. Tag t_2 is hashed to $B[6]$ ($v_{t_2}^1 = 6$, see Tab. 1), and $B[6]$ = '10'. By lines 1 to 3 in Procedure 2, t_2 gets the first bit of $B[6]$ which is $M_1[6]$ = '1' and gets $I_{t_2} = 3$, since three positions with value '1' located before $M_1[6]$. Then by lines 4 to 5, t_2 gets the second bit of $B[6]$ which is $M_2[I_{t_2}] = M_2[3] =$ '0'. Because its second bit = 0, tag t_2 finishes decoding from array B and gets its order $\Delta(t_2) = N_{t_2}^{10} = 2$ ($B[1] = B[2] =$ '10'). Finally, t_2 checks $D[2]$ = '010', then determines that it is an identified tag and enters into the sorted state, because

$D[2] = v_{t_2}^3 =$ '010' (If not, t_2 must be an unidentified tag and would enter into the sleep state).

As another example, we look at t_1, which is hashed to $B[4]$. By lines 1 to 3, t_1 gets that the first bit of $B[4]$ which is '1', and gets $I_{t_1} = 2$. By lines 4 to 6, t_1 gets the second bit of $B[4]$ which is '1', and gets $I_{t_1} = 0$. By lines 7 to 9, t_1 gets the third bit of $B[4]$ (= '1'), and updates $\mathbf{G} = \mathbf{G} + 3 + 2 \times 1$ ($N^{10} = 3$ because there are three '0's in M_2, $N^{110} = 1$ since there is one '0' in M_3). Finally, tag t_1 stays in the unsorted state and goes back to line 1 when the next round starts.

Next, we show the case of an unidentified tag u. If u is hashed to $B[0]$ ($v_u^1 = 0$), then by line 2 in Procedure 2, u realizes that it is an unidentified tag and enters into the sleep state. If u is hashed to $B[6]$ ($v_u^1 = 6$), then u will receive $B[6]$ and $D[2]$ in the same way as tag t_2 does, the only difference is that u will be in the sorted state with a probability 2^{-3}. This is because: R knows the ID of t_2, and sets $D[2]$ to be equal to $v_{t_2}^3$. Then the probability that the v_u^3 equals to $D[2]$ is 2^{-3}. Remember that all the hash values $v_t^3, t \in S \cup O$ contain 3 bits, and these hash values are independent and uniformly distributed random variables, thus v_t^3 can take any 3-bit value with equal probability. In line 11 of Procedure 2, if $v_u^3 =$ '010', u will enter into the sorted state, and get an order the same way as tag t_2 does. If $v_u^3 \neq$ '010', then u enters into the sleep state. Theorem 2 shall formally prove that an unidentified tag changes to the sorted state during the first phase of P_{sort} with a probability of ε.

C. THE SECOND PHASE OF P_{SORT}

After the first phase, the unidentified tags will no longer be considered, as we will prove in Theorem 2 that, after the first phase, any unidentified tag u enters into the sleep state with a probability of $1 - \varepsilon$. Then the second requirement of the tag-sorting problem is fulfilled.

However, after the first phase, there may exist a small number of identified tags that still stay in the unsorted state, thus we need the second phase to label all the remaining identified tags and assign them with unique integers.

The second phase is simple. Because the reader R has the complete knowledge of the IDs of identified tags in S, it can predicate the decisions made on each of them, and find out which of them are still in the unsorted state. Therefore, in the second phase, reader R sends out the IDs of these tags one by one (polling). Upon receiving an ID from reader R, a tag t in the unsorted state does the followings:

(1) t updates its local counter $\mathbf{G}$ by one;
(2) t compares the received ID with its own ID, and if they are the same, t shall enter into the sorted state, otherwise t remains in the unsorted state.

It is straightforward to see that reader R can have all remaining identified tags assigned with unique integers in the second phase.

D. THE ANALYSIS OF P_{SORT}

In this section, we analyze the performance of P_{sort} in theory.

Procedure 2 How a Tag t Receives B and D

Initialize a local variable $\mathbf{G} = 0$; ||* tag t stores the number of identified tags in sorted state, and updates $\mathbf{G}$ in each round $\mathbb{R}_k$*||
For $k = 1$ **to** $f = \log_2(\frac{1}{\varepsilon})$ ||*round $\mathbb{R}_k$ of the first phase P_{sort}*||
0: **Receiving** n_k and r from R, tag t computes a hash value $h(t) = (v_t^1, v_t^2, v_t^3)$;||* see **Step 1-2** in Section III-A*||
1: **Receiving** M_1 from R, tag t checks $M_1[v_t^1]$;
2: **If** $M_1[v_t^1] = 0$
Then tag t gets $B[v_t^1] =$ '0' and exits this procedure by setting itself into deactivated state; ||* t knows that it is an unidentified tag, and ignores all further messages.*||
3: **Else** t gets $I_t = N_t^{10} + N_t^{110} + N_t^{111}$;
EndIf
4: **Receiving** M_2, tag t checks $M_2[I_t]$;
5: **If** $M_2[I_t] = 0$
Then tag t gets $B[v_t^1] =$ '10' and sets $\Delta(t) = N_t^{10}$; **go to** 10; ||*do not receive M_3 *||
6: **Else** t sets $I_t = N_t^{110} + N_t^{111}$;
EndIf
7: **Receiving** M_3, t checks $M_3[I_t]$;
8: **If** $M_3[I_t] = 0$
Then tag t gets $B[v_t^1] =$ '110' and sets $\Delta(t) = N^{10} + 2\,N_t^{110} + v_t^2$; **go to** 10;
9: **Else** t gets $\mathbf{G} = \mathbf{G} + N^{10} + 2N^{110}$ and $B[v_t^1] =$ '111'; t stays in unsorted state; **go to** 1;
||* t knows all $N^{10} + 2N^{110}$ tags in L_k will change to sorted state; t ignores D and waits for the next round $\mathbb{R}_{k+1}$ *||
EndIf
10: **Receiving** D, tag t checks $D[\Delta(t)]$;
11: **If** $D[\Delta(t)] = v_t^3$
Then tag t gets its order $\Delta(t) = \Delta(t) + \mathbf{G}$ and enters into sorted state; ||*t determines that it is an identified tag*||
12: **Else** t exits this procedure by setting itself into deactivated state; ||* t is an unidentified tag*||
EndIf
End For
||* I_t in 3 is the position of the second bit of the cell $B[v_t^1]$ in M_2, and I_t in 6 is the position of the third bit of $B[v_t^1]$ in M_3;*||

Theorem 2: For any unidentified tag $u \in O$, the probability that u is in the sorted state is equal to ε, after the $f = \log_2(\frac{1}{\varepsilon})$ communication rounds in the first phase of P_{sort}.

Proof: Before round $\mathbb{R}_k$, there are $n_k = |S_k|$ identified tags staying in the unsorted state, and array B has $L = 2^c$ cells, where $c = \log_2(\frac{n_k}{\ln 2})$. The probabilities that $B[i] =$ '0', '10', '110', '111', $i \in \{1, \cdots, n_k\}$ are denoted by p_0, p_{10}, p_{110} and p_{111}, respectively.

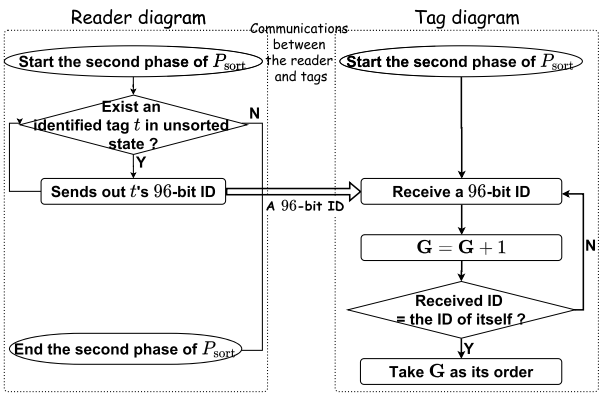

FIGURE 5. Overview of the second phase of protocol P_{sort}.

The hash values of $h(t) = H(t_{ID}, r) \mod 2^{c+f+1-k}, t \in S_k \cup O$ (see **Step 1** in Section III-A) are independent and uniformly distributed random variables.† Hence, v_t^1, v_t^2, v_t^3 are also independent and uniformly distributed random variables. Remember that $v_t^1 \in \{0, 1, \cdots, L-1\}$, $v_t^2 \in \{0, 1\}$ and $v_t^3 \in \{0, 1, \cdots, 2^{f-k}-1\}$ (see Eq. (1)). Furthermore, it is easy to get the following facts:

- $p_0 = (1-1/L)^{n_k}$, because $B[i]$ = '0' if and only if $\forall t \in S_k, v_t^1 \neq i$;
- $p_{10} = n_k(1/L)^1(1-1/L)^{n_k-1}$, because $B[i]$ = '10' if and only if there is only one tag $t \in S_k$ with $v_t^1 = i$;
- $p_{110} = \frac{1}{2}\binom{n_k}{2}(1/L)^2(1-1/L)^{n_k-2}$, because $B[i]$ = '110' if and only if there are exactly two tags t_1 and t_2 from S_k having the same hash value $v_{t_1}^1 = v_{t_2}^1 = i$ but $v_{t_1}^2 \neq v_{t_2}^2 \in \{0, 1\}$;
- p_{111} can be computed from equation $p_0 + p_{10} + p_{110} + p_{111} = 1$.

Then because n_k is relatively large, $L = n_k/\ln 2$ and $(1-1/L)^{n_k} \approx e^{-n_k/L} = 0.5$, we can obtain:

$$\begin{aligned} p_0 &= (1-1/L)^{n_k} \approx e^{-n_k/L} = e^{-n_k/(n_k/\ln 2)} = 0.5, \\ p_{10} &= n_k(\frac{1}{L})^1(1-\frac{1}{L})^{n_k-1} \approx \frac{n_k}{n_k/\ln 2} e^{-\frac{n_k}{n_k/\ln 2}} = \ln(2)/2, \\ p_{110} &= \frac{\binom{n_k}{2}}{2}(\frac{1}{L})^2(1-\frac{1}{L})^{n_k-2} \approx \frac{n_k^2}{4n_k^2/\ln^2(2)} e^{-\frac{n_k}{n_k/\ln 2}} \\ &= \ln^2(2)/8, \\ p_{111} &= 1 - 0.5 - 0.5 \times \ln 2 - \ln^2(2)/8. \end{aligned} \tag{5}$$

By the state diagram shown in Fig. 3, an unidentified tag u enters into the sorted state if and only if the following two events: E_u^B and E_u^D, happen simultaneously:

(1) E_u^B represents the event that u is hashed into a cell $B[v_u^1]$ with value '10' or '110';

(2) E_u^D represents the event that v_u^3 (the fingerprint of u) is equal to $D[\Delta(u)]$ ($D[\Delta(u)]$ stores the fingerprint of some identified tag t with $\Delta(t) = \Delta(u)$).

Based on (5), we can get the probabilities of these two events:

$$\begin{aligned} Pr\left(E_u^B\right) &= p_{10} + p_{110} = 0.5 + 0.5 * \ln 2, \text{ and} \\ Pr\left(E_u^D\right) &= (1/2)^{f-k} = (1/2)^{\log_2(1/\varepsilon)-k}. \end{aligned}$$

Now, we analyze the decisions made by unidentified tags. After receiving B and D, an unidentified tag u enters into the sorted state with probability:

$$X_k = Pr(E_u^B)Pr(E_u^D) = [\ln(2)/2 + \ln^2(2)/8]2^k\varepsilon. \tag{6}$$

If unidentified tag u does not enter into the sorted state, then it may enter into the sorted state in the following round only if tag u is mapped to a cell with value '111'. Thus, the probability that u stays in the unsorted state is $Y_k = p_{111}$.

For illustration, let $\alpha = \ln(2)/2 + \ln^2(2)/8$. From the above analysis, in round $\mathbb{R}_k$, an unidentified tag u has probability $X_k = \alpha 2^k\varepsilon$ to enter into the sorted state and has probability $Y_k = 0.5 - \alpha$ to remain in the unsorted state. Then, the probability that u wrongly decides it is an identified tag after the $\log_2(\frac{1}{\varepsilon})$ communication rounds (P_{err}), is computed as follows:

$$\begin{aligned} P_{err} &= X_1 + Y_1X_2 + \cdots + \prod_{j=1}^{f-1} Y_jX_f + \prod_{j=1}^{f} Y_j \\ &= \alpha 2^1\varepsilon + (0.5-\alpha)\alpha 2^2\varepsilon + \cdots + (0.5-\alpha)^{f-1}\alpha 2^f\varepsilon \\ &\quad + (0.5-\alpha)^f \\ &= \alpha\varepsilon \sum_{j=0}^{f-1} (0.5-\alpha)^j 2^{j+1} + (0.5-\alpha)^f \\ &= \varepsilon - \varepsilon((0.5-\alpha)2)^f + (0.5-\alpha)^f = \varepsilon. \end{aligned} \tag{7}$$

Note that $\prod_{j=1}^{f} Y_j$ is the probability that u stays in the unsorted state after $\mathbb{R}_{\log_2(1/\varepsilon)}$, and then u is assumed to enter into the sorted state in the second phase of P_{sort}. ■

Theorem 3: Given a set S of n identified tags, and a mis-assignment rate ε, let $|\mathbb{R}|$ denote the total number of bits transmitted during the running of $\mathbb{R}_1, \mathbb{R}_2, \cdots, \mathbb{R}_{\log_2(1/\varepsilon)}$ in the first phase. Then we have:

$$|\mathbb{R}| = n\log_2(1/\varepsilon) + 2n(1-\varepsilon^{1+\beta}), \tag{8}$$

where

$$\beta = \log_2(1/(1-0.5\ln 2)) \approx 0.61. \tag{9}$$

Proof: First we consider the number of bits that are contained in array B and D in $\mathbb{R}_k$, $k \in \{1, 2, \cdots, f\}$. Array B has $n_k/\ln(2)$ cells. The probabilities that a cell $B[i]$ takes value of '0', '10', '110', and '111' are given in Eq. (5). Then, the number of bits contained in B in $\mathbb{R}_k$ is

$$\begin{aligned} |B| &= (p_0 \times 1 + p_{10} \times 2 + (p_{110} + p_{111}) \times 3)\, n_k/\ln(2) \\ &= (2 - \ln 2/2)n_k/\ln(2). \end{aligned} \tag{10}$$

Second, from Section III-B, it is known that the number of bits contained in array D is $(2^{f-k})(N^{10} + 2N^{110})$ bits in each round $\mathbb{R}_k, k \in \{1, 2, \cdots, f\}$.

† As far as we know, almost all RFID protocols that are designed for various purposes including cardinality estimation, missing tag detection, unknown tag identification, and grouping, assume that the hash values of all tags in an RFID system are independent and uniformly distributed random variables, see [2], [30], [33]

Since the size of $(N^{10} + 2N^{110})$ $(= |L_k|)$ changes from round to round, we need a more detailed analysis. Let us look at the first round $\mathbb{R}_1$. Because $N_1 = n$, array B contains $n/\ln 2$ cells, $N^{10} = p^{10}n/\ln 2 = n/2$ and $N^{110} = p^{110}n/\ln 2 = n\ln(2)/8$, where p^{10} and p^{110} are probabilities given by Eq. (5). Set L_1 contains $N^{10} + 2N^{110} = (1/2 + \ln(2)/4)n$ identified tags. Now, the number of remaining unsorted identified tags for round $\mathbb{R}_2$ is:

$$n_2 = n - (1/2 + \ln(2)/4)n = (1/2 - \ln(2)/4)n = \gamma n,$$

where

$$\gamma = 1/2 - \ln(2)/4. \tag{11}$$

Applying the same analysis for $\mathbb{R}_2, \cdots, \mathbb{R}_{k-1}$, we get:

$$n_k = \gamma^{k-1}n, \ k \in \{1, 2, \cdots, f\}, \tag{12}$$

and the number of identified tags in L_k is:

$$n_k - n_{k+1} = \gamma^{k-1}n - \gamma^k n = \gamma^{k-1}(1-\gamma)n. \tag{13}$$

Note that formula (12) and (13) are also true for N_1 and L_1. Thus the number of bits in D for $\mathbb{R}_k$ is

$$|D| = \gamma^{k-1}(1-\gamma)n(f-k). \tag{14}$$

Therefore, we have $|B|+|D| = (2-\ln(2)/2)\gamma^{k-1}n/\ln(2)+\gamma^{k-1}(1-\gamma)n(f-k)$.

Finally, the total number of bits transmitted during the running of $\mathbb{R}_1, \cdots, \mathbb{R}_f$, denoted by $|\mathbb{R}|$, is obtained as follows:

$$\begin{aligned}
|\mathbb{R}| &= \sum_{k=1}^{f}[(2-\frac{\ln 2}{2})\frac{\gamma^{k-1}n}{\ln 2} + \gamma^{k-1}(1-\gamma)n(f-k)] \\
&= \frac{n(2-\frac{\ln 2}{2})}{\ln 2}\sum_{k=1}^{f}\gamma^{k-1} + (1-\gamma)n\sum_{k=1}^{f}[\gamma^{k-1}(f-k)] \\
&= \left(\frac{n(2-\frac{\ln 2}{2})}{\ln 2} + n(1-\gamma)f\right)\frac{1-\gamma^f}{1-\gamma} \\
&\quad - n(1-\gamma)\left(\frac{1-\gamma^f}{(1-\gamma)^2} - \frac{f\gamma^f}{1-\gamma}\right) \\
&= \frac{n(2-\frac{\ln 2}{2})}{\ln 2}\frac{1-\gamma^f}{1-\gamma} + nf(1-\gamma^f) - n\frac{1-\gamma^f}{1-\gamma} + nf\gamma^f \\
&= n[\frac{(2-\frac{\ln 2}{2})}{\ln 2} - 1]\frac{1-\gamma^f}{1-\gamma} + nf \\
&= n\frac{2\log_2(e) - 1.5}{0.5 + \ln(2)/4}(1-\varepsilon^{1+\beta}) + n\log_2(1/\varepsilon) \qquad (15) \\
&= 2n(1-\varepsilon^{1+\beta}) + n\log_2(1/\varepsilon). \qquad (16)
\end{aligned}$$

Remember that, in Eq. (15), we use the equality:

$$\gamma^f = \varepsilon^{1+\beta}, \tag{17}$$

which is based on $\gamma^f = (1/2 - \ln(2)/4)^{\log_2\frac{1}{\varepsilon}} = (1/2)^{\log_2\frac{1}{\varepsilon}}(1-\ln(2)/2)^{\log_2\frac{1}{\varepsilon}} = \varepsilon^{1+\beta}$, where β is defined in Eq. (9). ■

Theorem 4: Given a set S of n identified tags, a set O of unidentified tags and an error probability ε, the total number of bits transmitted by P_{sort}, denoted by $|P_{\text{sort}}|$, is

$$|P_{\text{sort}}| = n\log_2(\frac{1}{\varepsilon}) + 2n + 94\varepsilon^{1+\beta}n \approx n(\log_2(\frac{1}{\varepsilon}) + 2). \tag{18}$$

Moreover, at the end of P_{sort}, every identified tag $t \in S$ is in the sorted state and assigned with a unique order $\Delta(t) \in \{1, 2, \cdots, n\}$.

Proof: Let S° be the set of the remaining unsorted identified tags after the first phase of P_{sort}, and $N^\circ = |S^\circ|$. Since the first phase consists of f rounds, by Eq. (12), (11) and (17), we have

$$N^\circ = \gamma^f n = \varepsilon^{1+\beta}n.$$

N° is also the number of the identified tags staying in the unsorted state when the second phase of P_{sort} starts.

Since the second phase simply assigns each tag in S° by pooling their IDs, its communication cost is less than $96\varepsilon^{1+\beta}n$. Then by the communication cost of the first phase which is shown in Eq. (8), we get (18) easily. The approximation used in (18) comes from the fact $94\varepsilon^{1+\beta} \approx 94\varepsilon^{1.61}$ is much less than $\log_2(\frac{1}{\varepsilon}) + 2$. We should notice that the error probability ε is typically set to be less than 10^{-2} in RFID systems for keeping most of unidentified tags from interfering with identified tags.

Finally, as each tag poses a unique ID, it is clear that every tag $t \in S$ in the sorted state can get a unique order $\Delta(t) \in \{1, 2, \cdots, n\}$. ■

Further, we present the minimal communication cost of the existing state-of-the-art protocols capable of solving the tag-sorting problem [29], [32]–[34] and compare it with $|P_{\text{sort}}|$ to demonstrate the superiority of the proposed protocol P_{sort}.

Theorem 5: Given a set S of n identified tags, a set O of unidentified tags and an error probability ε, the minimal communication cost of the existing protocols [29], [32]–[34] is

$$1.44n\log_2(1/\varepsilon) + 2.71n, \tag{19}$$

which is much larger than the communication cost of the proposed protocol P_{sort}.

Proof: The existing protocols [29], [32]–[34] in general solve the tag-sorting problem by following two steps.

- Step-1: They encode the information of identified tags into a Bloom filter, and send this filter out to deactivate unidentified tags but keep identified tags active.
- Step-2: They inform identified tags of unique integers by an aloha-like protocol. The basic idea is to randomly hash tags to different slots of a communication frame, and use the singleton slots to assign different integers to different tags. Note that a slot is called a singleton slot if there is exactly one tag hashed to this slot.

We analyze the minimal communication costs needed in these two steps, which in turn provide the minimal communication cost of the existing protocols. Step-1 needs to send at

least $1.44\, n\log_2(1/\varepsilon)$ bits, because a Bloom filter requires at least a number of $1.44\, n\log_2(1/\varepsilon)$ bits to encode a set of n identified tags and deactivate unidentified tags with a probability of $1-\varepsilon$ [38]. Step-2 requires a minimal communication cost of $2.71\, n$ bits because every communication frame in an aloha-like protocol generally needs $e \approx 2.71$ slots to assign a unique integer to a tag, and each slot requires at least one bit to represent [39]. So, we have (19) as the minimal communication cost of the existing protocols, Clearly, (19) is much larger than $|P_{\text{sort}}|$ (shown in (18)). ■

Lastly, we analyze the computational costs on the reader side and the tag side. On the reader side, the average computational cost depends on the expected number of times the reader R executes the hash function $h()$ (see Step 3 of round $\mathbb{R}_k$ in section III-A). On the tag side, the average computational cost depends on the expected number of times the tag executes the hash function $h()$ (see Step 2 of round $\mathbb{R}_k$ in section III-A). Below is a detailed analysis. Note that the second phase of P_{sort} is omitted as this phase requires no hash function.

Theorem 6: In P_{sort}, the average computational cost of reader R is $O(n)$ and the average computational cost of a tag t is $O(1)$.

Proof: Based on the proof for Theorem 3, we know only a fraction of $\gamma = 1/2 - \ln(2)/4 \approx 0.32$ tags remain in the unsorted state (γ is defined (11)). Based on Step 3 of round $\mathbb{R}_k$, reader R needs to compute the hash function $h()$ again in the next round $\mathbb{R}_{k+1}$ for the tags in the unsorted state. Since n tags stay unsorted initially, the expected number of times that the reader R executes the function $h()$ is $n+\gamma n+\gamma^2 n+\cdots+\gamma^{\lceil\log_2(\frac{1}{\varepsilon})\rceil-1}n = O(n)$.

Based on the proof for Theorem 3, we observe that a tag t has a probability of γ to remain in the unsorted state in each round $\mathbb{R}_k$ for $k = 1 \sim \lceil\log_2(\frac{1}{\varepsilon})\rceil$. By Step 2 of round $\mathbb{R}_k$, any tag in the unsorted state needs to compute the hash function $h()$ again in the next round $\mathbb{R}_{k+1}$. Hence, the expected number of times that a tag calls function $h()$ is $1+\gamma+\cdots+\gamma^{\lceil\log_2(\frac{1}{\varepsilon})\rceil-1} = O(1)$. ■

IV. PERFORMANCE EVALUATION

This section shows simulations comparing the proposed protocol's performance with other protocols. We choose to compare P_{sort} with three existing state-of-the-art protocols: PIC [29], MIC [33], and TPP [34] each of which can efficiently collect the information from a tag population. Because when collecting the information from each tag, they implicitly assign an order (a unique integer) to each collected tag.[‡] But these three existing protocols inherently do not possess the ability to filter out unidentified tags before assigning identified tags with unique integers. For a fair comparison, we equip them with the classic Bloom filter for filtering unidentified tags out. In particular, we choose to encode the information of identified tags into a Bloom filter and then let the reader R send this Bloom filter out to deactivate unidentified at the beginning of each of these three protocols. Clearly, with the help of a Bloom filter, these protocols that collect the information from tags can solve the tag-sorting problem. Please note that, for each of the three protocols (PIC, MIC, and TPP), the communication cost is the sum of the communication cost for assigning tags unique integers and the number of bits used by the Bloom filter.

[‡]A tag t can have its information successfully received by the reader R if no other tags are sending messages back at the same when t sends its information [10].

A. SIMULATION SETTING

We set the simulation according to the specification of the EPC C1G2 standard [10]. The transmission rate from the reader to tags is set to be 26.5Kb/s, and there is an idle time interval of 302μs separating two sequential transmissions. With this time setting, it takes the reader $T_{\text{id}} = \frac{96}{26.5\text{Kb/s}} + 302\mu\text{s} = 3927\mu s$ to transmit a 96-bit string to tags. Since, in the tag-sorting problem, tags need not send feedback to the reader, the transmission rate from a tag to the reader is not set in our simulation. It should be noted that setting other transmission rates affect only the absolute communication time used by each of the four protocols, not the comparison conclusions.

We are mainly interested in the communication cost, or communication time equivalently, of these four protocols. Because it is usually the most time-consuming part of RFID systems and thus affects the system efficiency greatly [1]–[3], [24], [29], [30], [32], [35]–[37]. Remember that the communication cost between RFID readers and tags is viewed as a major issue, because, RFID readers communicate with tags through a low-speed link over which the communication time largely determines the running time of a tag-management protocol. The communication times of the four protocols are measured with respect to n (the number of the identified tags), m (the number of the unidentified tags), and ε (the error probability). For this purpose, we set up 4 different scenarios, each of which bears a unique relationship among n, m and ε. A detailed setting of the 4 scenarios is described below.

- Scenario 1 varies n from 10^3 to 10^4, but keeps m and ε fixed to 10^3 and 10^{-3}, respectively;
- Scenario 2 increases m from 1 to 10^5, but sets n and ε fixed to 10^3 and 10^{-3}, respectively;
- Scenario 3 decreases ε from 10^{-1} down to 10^{-5}, but keeps n and m fixed to 10^4 and 10^4, respectively;
- Scenario 4 varies m from 10^1 to 10^6, but sets $\varepsilon = 1/m$ and $n = 2\times 10^4$.

B. SIMULATION RESULTS

Fig. 6, 7, 8, and 9 display the communication times for the different protocols in the four scenarios, respectively. We then discuss these results in detail for each scenario.

The conclusion for Scenario 1 can be immediately drawn from Fig. 6: with the increase of n, the communication times of the four protocols increase linearly. This fact testifies: the

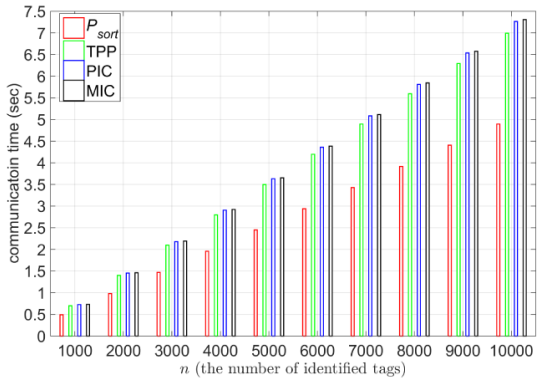

FIGURE 6. Communication time for Scenario 1 where $n \in \{10^3, 2 \times 10^3, \cdots, 10^4\}$ and $m = 10^3$, and $\varepsilon = 10^{-3}$.

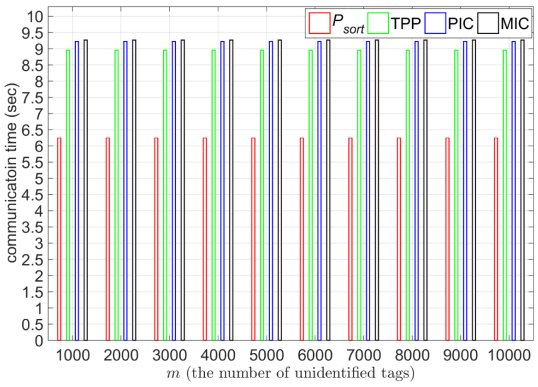

FIGURE 7. Communication time for Scenario 2 where $m \in \{10^3, 2 \times 10^3, \cdots, 10^4\}$ and $n = 10^4$, and $\varepsilon = 10^{-4}$.

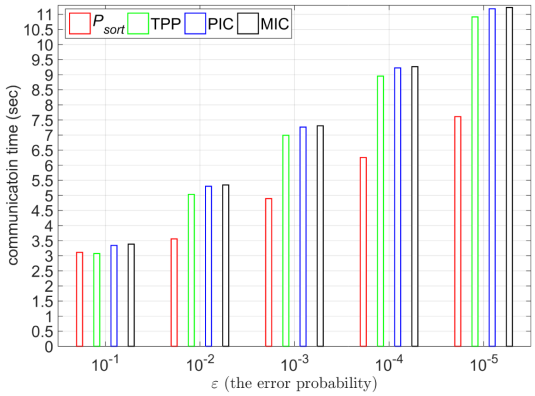

FIGURE 8. Communication time for Scenario 3 where $\varepsilon \in \{10^{-1}, 10^{-2}, \cdots, 10^{-5}\}$ and $n = 10^4$, and $m = 10^4$.

theoretical communication time of P_{sort} increases linearly with n, which is proven in Theorem 4. It is also evident from Fig. 6 that P_{sort} uses the shortest communication time out of all four protocols. For example, when $n = 10^4$, the communication times of P_{sort}, TPP, PIC and MIC are 4.9s, 7.2s, 7.0s, and 7.3s, respectively. On average, the communication time of P_{sort} is at most 70% of the other three protocols.[§]

Based on Fig. 7, we can observe that the communication time of P_{sort} roughly remains unchanged with the increase of m. This is because the tag sorting problem only requires that every unidentified tag is kept from receiving any integer with the same user-specific probability ε no matter how many unidentified tags are present. Indeed, the user-specific probability ε can be viewed as the fraction of the unidentified tags incorrectly receiving integers over all of the unidentified tags. Specifically, suppose there are m unidentified tags. Then, on average, there will be $m\varepsilon$ unidentified tags incorrectly receiving unique integers. Hence, although the communication time remains constant with the rise of the number of unidentified tags, the number of unidentified tags incorrectly receiving integers increases. This fact can also be drawn from (18) of Theorem 4. In Fig. 7, it is very easy to notice that P_{sort} bears a communication time much less than other protocols. For example, when $m = 5000$, P_{sort} has a communication time of 6.3s, while TPP, PIC, and MIC have a communication time of 9.2s, 8.9s, and 9.3s, respectively. Similar to scenario 1, the communication time of P_{sort} is about 70% of that of the other three protocols.

Fig. 8 demonstrates that the communication time of P_{sort} increases when ε decreases. In light of this fact, (18) of Theorem 4 is valid. This figure also shows the comparison result of the proposed P_{sort} with the other three protocols. If $\varepsilon = 10^{-1}$, the four protocols roughly have the same communication time; but if $\varepsilon \leq 10^{-12}$, the communication time of P_{sort} becomes much less than that of the other three (no more than 72% of the communication time of other protocols).

[§] Note that, in our simulation, MIC, PIC, and TPP uses a bloom filter to filter out unidentified tags when solving the tag-sorting problem, and the minimal number of bits used by a classic Bloom filter for filtering out an unidentified with a probability ε is $\log_2(e)\log_2(1/\varepsilon) \times n$, which comes section 2.2 of [38].

For example, when $\varepsilon = 10^{-1}$, the communication times of the P_{sort}, TPP, PIC, and MIC are 3.1s, 3.3s, 3.1s and 3.4s respectively; but when $\varepsilon = 10^{-2}$, their communication times are 3.6s, 5.3s, 5.0s, and 5.3s, respectively. The reason is that when ε is relatively large, protocol P_{sort} uses a small number of bits to filter unidentified tags out, and thus it cannot reuse many of these bits when assigning integers to identified tags. However, when ε is relatively small, the number of bits for filtering unidentified tags out becomes large, then P_{sort} can reuse many of these bits for assigning integers to identified tags. This fact is in line with Theorem 4, which indicates that when ε is large, the subpart $94\varepsilon^{1+\beta}n$ becomes a large overhead that we cannot omit safely, but when ε turns to be small, this subpart decreases quickly to 0. We should notice that, in a practical RFID system, ε usually needs to be set small, especially for those busy or mobile RFID systems [2], [30], [34], [50].

Fig. 9 shows that P_{sort}'s communication time increases slower than that shown in Fig. 6 and 8, when both m and ε increase. Similar to Fig. 6-8, we observe that P_{sort}'s communication time is about 71% of that of the other three protocols.

In summary, for scenario 1-4, the proposed protocol P_{sort} reduces the communication time by 30% on average. P_{sort}'s high efficiency of stems from the fact that: there are usually

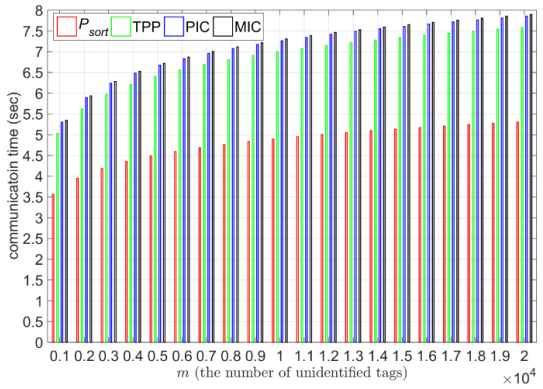

FIGURE 9. Communication time for Scenario 4 where $m \in \{10^3, 2 \times 10^3, \cdots, 2 \times 10^4\}$, $\varepsilon = 10/m$ and $n = 10^4$.

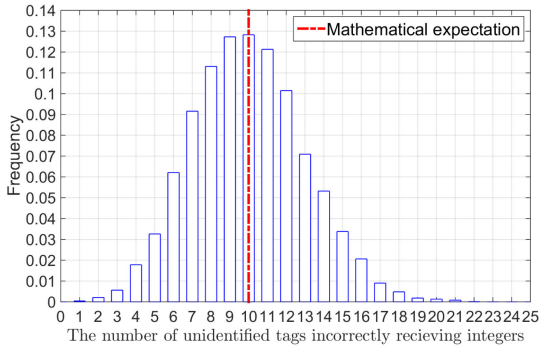

FIGURE 10. The empirical probability mass function of random variable Z for $m = 100$ and $\varepsilon = \frac{1}{10}$.

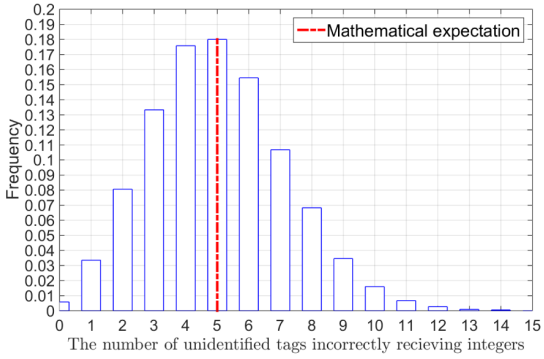

FIGURE 11. The empirical probability mass function of random variable Z for $m = 100$ and $\varepsilon = \frac{1}{20}$.

a large number of bits used for filtering unidentified tags out, and P_{sort} can effectively reuse many of these bits to assigning identified tags unique integers, while the other protocols cannot.

Next, the practical accuracy of the proposed protocol P_{sort} is evaluated by counting the actual total number of unidentified tags incorrectly receiving integers during one execution of P_{sort}. For illustration, let us use Z to represent this number. Clearly, Z is a random variable because every unidentified tag incorrectly receives a unique integer with the same

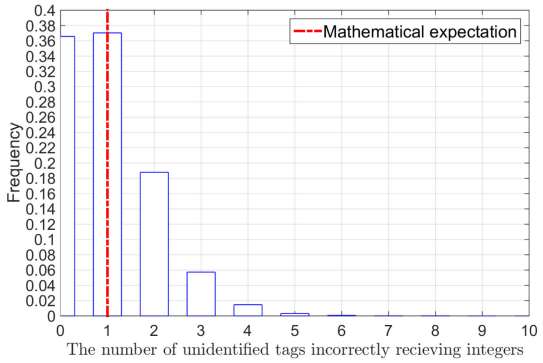

FIGURE 12. The empirical probability mass function of random variable Z for $m = 100$ and $\varepsilon = \frac{1}{100}$.

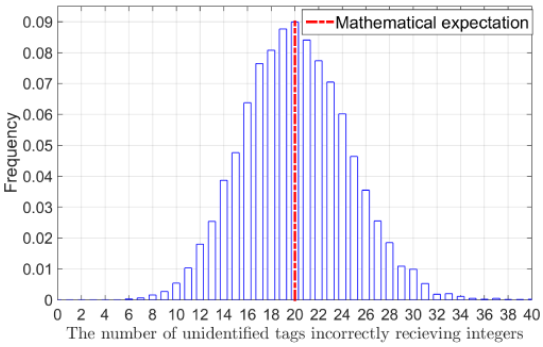

FIGURE 13. The empirical probability mass function of random variable Z for $m = 1000$ and $\varepsilon = \frac{1}{50}$.

probability ε set by the user. Then, the empirical probability mass function of Z is investigated with 10^4 independent trials, and each trial consists of the following three steps:

- s-1 We choose m unidentified tags and n identified tags randomly from the set of all possible tags;
- s-2 We use the proposed protocol to solve the tag-sorting problem over these $m + n$ tags;
- s-3 We record the number of the unidentified tags that incorrectly receive unique integers in this trial.

We show the empirical probability mass function of Z with different values of m and ε. Please note that since n (the number of identified tags) is irrelevant to Z (all tags make their decisions locally and independently), it is fixed to 10^3 in all trails. First, we set $m = 100$, and compute three empirical probability mass functions of Z with three different values of ε: $\frac{1}{10}$, $\frac{1}{20}$, and $\frac{1}{100}$, respectively. The three empirical probability mass functions are shown in Fig. 10, 11, and 12. Second, we change m to 1000, and then compute random variable Z's empirical probability mass functions with three different values of ε: $\frac{1}{50}$, $\frac{1}{100}$, and $\frac{1}{200}$, respectively. The three empirical probability mass functions are shown in Fig. 12, 13, and 14.

Fig. 10-15 show that random variable Z is highly concentrated to its mathematical expectation $E[Z] = \varepsilon m$, which is represented by a red-dot line in each figure. For example,

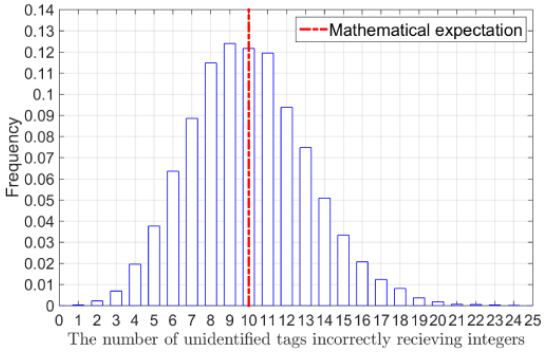

FIGURE 14. The empirical probability mass function of random variable Z for $m = 1000$ and $\varepsilon = \frac{1}{100}$.

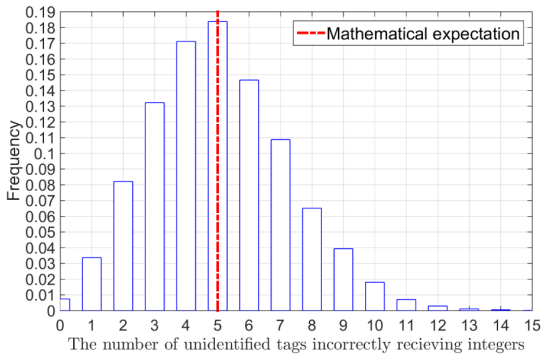

FIGURE 15. The empirical probability mass function of random variable Z for $m = 1000$ and $\varepsilon = \frac{1}{200}$.

in Fig. 12, the probability that $Z > 4\ E[Z]$ is only 3×10^{-3}; in Fig. 15, the probability that $Z > 3\ E[Z]$ is only 4×10^{-4}. We may also explain these experimental results with a brief theoretical analysis as follows. Let O represent the set of unidentified tags. Let X_t be a random variable defined for an unidentified tag $t \in O$ such that $X_t = 1$ represents the event that t incorrectly receives a unique integer in a trial, and $X_t = 0$ otherwise. Then, We have $Z = \sum_{t \in O} X_t$. Since each unidentified tag incorrectly receives a unique integer with the same probability ε, we have $Pr(X_t = 1) = \varepsilon$, $E[X_t] = \varepsilon$, and then $E[Z] = E[\sum_{t \in O} X_t] = \varepsilon m$. Furthermore, since each unidentified tag makes decision independently, we can use the Chernoff bound to random variable Z and get an inequality: $\Pr(Z > 4\ E[Z]) = \Pr(\sum_{t \in O} X_t > (1 + 3)\varepsilon m) < \left(\frac{e^3}{(1+3)^{1+3}}\right)^{\varepsilon m} < (0.08)^{\varepsilon m}$. It is now clear that random variable Z is highly concentrated to its expectation $E[Z]$, as m (the number of unidentified tags) is usually large in RFID systems where new tags arrive continuously. Hence, in practice, we can roughly take $E[Z] = \varepsilon m$ as a reliable estimation of the actual number of unidentified tags incorrectly receiving integers during one execution of protocol P_{sort}.

V. CONCLUSION

This paper investigates the tag-sorting problem, which substantially affects the efficiency of RFID systems for IoT networks. A fast sorting protocol P_{sort} is designed for this problem, and is shown to be able to put the set S of identified tags into a certain order by assigning them with unique integers (orders) from $\{1, 2, \cdots, |S|\}$ and keep unidentified tags from receiving these integers. The communication time of protocol P_{sort} is rigorously analyzed in theory. With extensive simulations, the practical communication time of P_{sort} is carefully tested and shown to be much less than that of the existing protocols. All these results indicate the superiority of the proposed protocol P_{sort}.

REFERENCES

[1] H. Landaluce, L. Arjona, A. Perallos, F. Falcone, I. Angulo, and F. Muralter, "A review of IoT sensing applications and challenges using RFID and wireless sensor networks," *Sensors*, vol. 20, no. 9, p. 2495, Apr. 2020.

[2] L. Zhang, W. Xiang, and X. Tang, "An efficient bit-detecting protocol for continuous tag recognition in mobile RFID systems," *IEEE Trans. Mobile Comput.*, vol. 17, no. 3, pp. 503–516, Mar. 2018.

[3] B. Baruah and S. Dhal, "An IoT based secure object tracking system," *Wireless Pers. Commun.*, vol. 106, pp. 1209–1242, Feb. 2019.

[4] N. B. Soni and J. Saraswat, "A review of IoT devices for traffic management system," in *Proc. Int. Conf. Intell. Sustain. Syst. (ICISS)*, Dec. 2017, pp. 1052–1055.

[5] S. Misbahuddin, J. A. Zubairi, A. Saggaf, J. Basuni, S. Wadany, and A. Al-Sofi, "IoT based dynamic road traffic management for smart cities," in *Proc. 12th Int. Conf. High-Capacity Opt. Netw. Enabling/Emerg. Technol. (HONET)*, Dec. 2015, pp. 1–5.

[6] Z. Liu and K. Ota, *Smart Technologies for Emergency Response and Disaster Management*. Hershey, PA, USA: IGI Global, 2017.

[7] O. Salman, I. Elhajj, A. Kayssi, and A. Chehab, "Edge computing enabling the Internet of Things," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 603–608.

[8] X. Liu, J. Cao, Y. Yang, W. Qu, X. Zhao, K. Li, and D. Yao, "Fast RFID sensory data collection: Trade-off between computation and communication costs," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1179–1191, Jun. 2019.

[9] X. Wang, Y. Gao, Y. Yang, X. Zheng, X. Wu, and W. Zhao, "An efficient protocol for the tag-information sampling problem in RFID systems," *Mobile Netw. Appl.*, pp. 1–12, Mar. 2021. [Online]. Available: https://link.springer.com/article/10.1007/s11036-021-01738-0, doi: 10.1007/s11036-021-01738-0.

[10] EPCglobal. (2018). *EPC Radio-Frequency Identity Protocols Generation-2 UHF RFID Standard, Specification for RFID Air Interface Protocol for Communications at 860 MHz–960 MHz, Version 2.1*. [Online]. Available: https://www.gs1.org/sites/default/files/docs/epc/gs1-epc-gen2v2-uhf-airinterface_i21_r_2018-09-04.pdf

[11] A. Nordrum, "The Internet of fewer things [news]," *IEEE Spectr.*, vol. 53, no. 10, pp. 12–13, Oct. 2016.

[12] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Mobile edge computing for the Internet of vehicles: Offloading framework and job scheduling," *IEEE Veh. Technol. Mag.*, vol. 14, no. 1, pp. 28–36, Mar. 2019.

[13] Z. Liu, T. Tsuda, H. Watanabe, S. Ryuo, and N. Iwasawa, "Data driven cyber-physical system for landslide detection," *Mobile Netw. Appl.*, vol. 24, no. 3, pp. 991–1002, Jun. 2019.

[14] C. Wu, Z. Liu, D. Zhang, T. Yoshinaga, and Y. Ji, "Spatial intelligence toward trustworthy vehicular IoT," *IEEE Commun. Mag.*, vol. 56, no. 10, pp. 22–27, Oct. 2018.

[15] Z. Liu, J. Li, X. Chen, C. Wu, S. Ishihara, Y. Ji, and J. Li, "Fuzzy logic-based adaptive point cloud video streaming," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 121–130, 2020.

[16] C. Wu, Z. Liu, F. Liu, T. Yoshinaga, Y. Ji, and J. Li, "Collaborative learning of communication routes in edge-enabled multi-access vehicular environment," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1155–1165, Dec. 2020.

[17] Y. Gao, Y. Cui, X. Wang, and Z. Liu, "Optimal resource allocation for scalable mobile edge computing," *IEEE Commun. Lett.*, vol. 23, no. 7, pp. 1211–1214, Jul. 2019.

[18] Z. Zhou, H. Yu, C. Xu, Z. Chang, S. Mumtaz, and J. Rodriguez, "BEGIN: Big data enabled energy-efficient vehicular edge computing," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 82–89, Dec. 2018.

[19] Z. Liu, S. Ishihara, Y. Cui, Y. Ji, and Y. Tanaka, "JET: Joint source and channel coding for error resilient virtual reality video wireless transmission," *Signal Process.*, vol. 147, pp. 154–162, Jun. 2018.

[20] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.
[21] Z. Zhou, J. Feng, L. Tan, Y. He, and J. Gong, "An air-ground integration approach for mobile edge computing in IoT," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 40–47, Aug. 2018.
[22] X. Wang, Z. Liu, S. Ishihara, Z. Dang, and J. Li, "A near-optimal protocol for the subset selection problem in RFID systems," in *Proc. 16th Int. Conf. Mobility, Sens. Netw. (MSN)*, Dec. 2020, pp. 33–42.
[23] D. J. Yeager, A. P. Sample, J. R. Smith, and J. R. Smith, "WISP: A passively powered UHF RFID tag with sensing and computation," in *RFID Handbook: Applications, Technology, Security, and Privacy*. 2008, pp. 261–278.
[24] X. Liu, X. Xie, S. Wang, J. Liu, D. Yao, J. Cao, and K. Li, "Efficient range queries for large-scale sensor-augmented RFID systems," *IEEE/ACM Trans. Netw.*, vol. 27, no. 5, pp. 1873–1886, Oct. 2019.
[25] S. Zhang, X. Liu, S. Guo, A. Y. Zomaya, and J. Wang, "Why queue up? Fast parallel search of RFID tags for multiple users," in *Proc. 21st Int. Symp. Theory, Algorithmic Found., Protocol Design Mobile Netw. Mobile Comput.*, Oct. 2020, pp. 211–220.
[26] H. Chen, G. Ma, Z. Wang, Q. Wang, and J. Yu, "MAC: Missing tag iceberg queries for multi-category RFID systems," *IEEE Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9947–9958, Oct. 2018.
[27] Z. Zhou, H. Liao, B. Gu, K. M. S. Huq, S. Mumtaz, and J. Rodriguez, "Robust mobile crowd sensing: When deep learning meets edge computing," *IEEE Netw.*, vol. 32, no. 4, pp. 54–60, Jul. 2018.
[28] X. Wang, Z. Liu, Y. Gao, X. Zheng, X. Chen, and C. Wu, "Near-optimal data structure for approximate range emptiness problem in information-centric Internet of Things," *IEEE Access*, vol. 7, pp. 21857–21869, 2019.
[29] X. Xie, X. Liu, K. Li, B. Xiao, and H. Qi, "Minimal perfect hashing-based information collection protocol for RFID systems," *IEEE Trans. Mobile Comput.*, vol. 16, no. 10, pp. 2792–2805, Oct. 2017.
[30] X. Wang, Z. Liu, Y. Gao, X. Zheng, Z. Dang, and X. Shen, "A near-optimal protocol for the grouping problem in RFID systems," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1257–1272, Apr. 2021.
[31] R. Pan, Z. Li, J. Cao, H. Zhang, and X. Xia, "Electrical load tracking scheduling of steel plants under time-of-use tariffs," *Comput. Ind. Eng.*, vol. 137, Nov. 2019, Art. no. 106049.
[32] S. Chen, M. Zhang, and B. Xiao, "Efficient information collection protocols for sensor-augmented RFID networks," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 3101–3109.
[33] Y. Qiao, S. Chen, T. Li, and S. Chen, "Tag-ordering polling protocols in RFID systems," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1548–1561, Jun. 2016.
[34] J. Liu, B. Xiao, X. Liu, and L. Chen, "Fast RFID polling protocols," in *Proc. 45th Int. Conf. Parallel Process. (ICPP)*, Aug. 2016, pp. 304–313.
[35] Y. Zhang, S. Chen, Y. Zhou, and Y. Fang, "Missing-tag detection with unknown tags," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1297–1310, Jun. 2020.
[36] J. Yu, W. Gong, J. Liu, L. Chen, K. Wang, and R. Zhang, "Missing tag identification in COTS RFID systems: Bridging the gap between theory and practice," *IEEE Trans. Mobile Comput.*, vol. 19, no. 1, pp. 130–141, Jan. 2020.
[37] X. Liu, S. Chen, J. Liu, W. Qu, F. Xiao, A. X. Liu, J. Cao, and J. Liu, "Fast and accurate detection of unknown tags for RFID systems—Hash collisions are desirable," *IEEE/ACM Trans. Netw.*, vol. 28, no. 1, pp. 126–139, Feb. 2020.
[38] A. Broder and M. Mitzenmacher, "Network applications of Bloom filters: A survey," *Internet Math.*, vol. 1, no. 4, pp. 485–509, Jan. 2004.
[39] S.-R. Lee, S.-D. Joo, and C.-W. Lee, "An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification," in *Proc. 2nd Annu. Int. Conf. Mobile Ubiquitous Syst., Netw. Services*, 2005, pp. 166–172.
[40] L. Shangguan and K. Jamieson, "The design and implementation of a mobile RFID tag sorting robot," in *Proc. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2016, pp. 31–42.
[41] J. Lai, C. Luo, J. Wu, J. Li, J. Wang, J. Chen, G. Feng, and H. Song, "TagSort: Accurate relative localization exploring RFID phase spectrum matching for Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 389–399, Jan. 2020.
[42] J. Xu, L. Yang, Q. Liu, J. Hu, and T. Song, "A sorting algorithm for RFID tags moving on a conveyor belt," in *Proc. IEEE Int. Conf. RFID (RFID)*, Apr. 2018, pp. 1–7.
[43] F. Bernardini, A. Buffi, D. Fontanelli, D. Macii, V. Magnago, M. Marracci, A. Motroni, P. Nepa, and B. Tellini, "Robot-based indoor positioning of UHF-RFID tags: The SAR method with multiple trajectories," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–15, 2021.
[44] A. Tzitzis, S. Megalou, S. Siachalou, T. Yioultsis, A. Kehagias, E. Tsardoulias, A. Filotheou, A. Symeonidis, L. Petrou, and A. G. Dimitriou, "Phase relock-localization of RFID tags by a moving robot," in *Proc. 13th Eur. Conf. Antennas Propag. (EuCAP)*, 2019, pp. 1–5.
[45] C. Li, E. Tanghe, D. Plets, P. Suanet, N. Podevijn, J. Hoebeke, E. D. Poorter, L. Martens, and W. Joseph, "Phase-based variant maximum likelihood positioning for passive UHF-RFID tags," in *Proc. 14th Eur. Conf. Antennas Propag. (EuCAP)*, Mar. 2020, pp. 1–5.
[46] Z. Li, C. He, J. Li, and X. Huang, "RFID reader anti-collision algorithm using adaptive hierarchical artificial immune system," *Expert Syst. Appl.*, vol. 41, no. 5, pp. 2126–2133, Apr. 2014.
[47] F. Campioni, S. Choudhury, and F. Al-Turjman, "Readers scheduling for RFID networks in the IoT era," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2018, pp. 1–6.
[48] B. Cao, Y. Gu, Z. Lv, S. Yang, J. Zhao, and Y. Li, "RFID reader anti-collision based on distributed parallel particle swarm optimization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3099–3107, Mar. 2021.
[49] L. Xie, H. Han, Q. Li, J. Wu, and S. Lu, "Efficient protocols for collecting histograms in large-scale RFID systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 9, pp. 2421–2433, Sep. 2015.
[50] L. Zhu, X. Wang, Y. Yang, S. Xu, X. Wu, W. Zhao, and H. Feng, "EPC-based efficient tag selection in RFID systems," *IEEE Access*, vol. 8, pp. 20546–20556, 2020.
[51] P. Šolić, J. Radić, and N. Rožić, "Energy efficient tag estimation method for ALOHA-based RFID systems," *IEEE Sensors J.*, vol. 14, no. 10, pp. 3637–3647, Oct. 2014.
[52] X. Liu, H. Qi, K. Li, I. Stojmenovic, A. X. Liu, Y. Shen, W. Qu, and W. Xue, "Sampling Bloom filter-based detection of unknown RFID tags," *IEEE Trans. Commun.*, vol. 63, no. 4, pp. 1432–1442, Apr. 2015.

YANGZHAO YANG received the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2014. He is a Senior Researcher with Shenzhen Cyberaray Network Technology Company Ltd., Shenzhen, China. His current research interests include artificial intelligence and social networks.

XIUJUN WANG received the Ph.D. degree in computer software and theory from the University of Science and Technology of China, in 2011. He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University of Technology. His research interests include data stream processing, randomized algorithm, and the Internet of Things.