Received December 18, 2019, accepted January 9, 2020, date of publication January 24, 2020, date of current version January 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2969188

EPC-Based Efficient Tag Selection in RFID Systems

LIYUE ZHU[1], XIUJUN WANG[2], YANGZHAO YANG[3], SHUBIN XU[3], XUANGOU WU[2], WEI ZHAO[2], (Member, IEEE), AND HUIBIN FENG[4]
[1]Academy of Broadcasting Science, Beijing 100866, China
[2]School of Computer Science and Technology, Anhui University of Technology, Ma'anshan 243032, China
[3]Research Institute of Cyberspace Security of CETC, Beijing 100041, China
[4]Electronic Information and Control of Fujian University Engineering Research Center, Minjiang University, Fuzhou 350121, 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, Grant 61672038, and Grant 61702006, 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 Provincial Key Research and Development Program of Anhui Province under Grant 201904a05020071.

ABSTRACT Tag-selection problem, which selects a set of wanted tags from a tag population, is vital for boosting efficiencies of the real-time applications in RFID systems. However, prior arts for the problem can not be applied to RFID systems directly, given that they either require additional computing functions implemented in tag's chips or require a time-consuming pre-process with a large communication cost. This paper studies the tag-selection problem and propose an efficient Electronic Product Code (EPC)-based tag selection method with theoretical analysis. In particular, firstly, we prove a nontrivial lower bound of communication overhead for a protocol which is capable of solving the tag-selection problem. Secondly, we propose an efficient protocol, denoted by TagSP, which only uses the ''select'' command (a mandatory command that all RFID tags shall support) and EPC. The proposed TagSP can be applied directly into off-the-shelf RFID systems with a communication overhead close to the lower bound. Extensive simulations are conducted and the simulation results show TagSP's superiority compared with existing protocols.

INDEX TERMS RFID systems, tag selection, lower bound of communication overhead, EPC.

I. INTRODUCTION

Radio frequency identification (RFID) is now widely used in many applications, such as warehouse controlling [1]–[8], library inventory [9]–[11] and object tracking [8], [12]–[18]. In comparison to the traditional bar-codes, RFID enables non-line-of-sight and contactless identification and supports a much larger set of different tag users [19], which leads to its wide applications.

This paper focuses on the tag-selection problem in RFID system, where the task is to pick a subset P of tags from a tag population S by using as little communication time as possible. This is a basic problem that has a wide range of applications in RFID systems. For example, when a batch of tagged goods is going to be shipped out in a warehouse, the manager may need to write a piece of information (e.g., shipping destination, expiry date, mode of transportation, shipper code) into the tag's memory for tracking purposes. Such writing task can be rapidly finished by first picking out the tags that share the same information (e.g., have the same shipping destination) from the tag population, and then writing the shared information into the memory of these picked tags simultaneously in one transmission. Another example is in privacy-sensitive RFID systems, where the access passwords in tags need to be updated periodically for enhancing the security level. In this case, instead of assigning each tag a new password which is time-consuming, the manager can partition all tags into different groups and then generate a new password for each group periodically. Other examples [20]–[26] include over-the-air tag programming, live query of tag information, etc.

A. PRIOR ARTS AND LIMITATIONS

There are mainly two problems in regard to the tag-selection problem. The first is the group writing problem recently raised in [20]. This problem aims at quickly picking a tag subset P from a tag population S, and then writing group-specific information into the tags in P. The authors in [20] propose a scheme to solve this problem, where each tag needs to be written with a specially designed bit-vector in

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

its user memory such that a single "select" command is capable of picking multiple tags by exploiting the features in these bit-vectors. Experimental results show that the number of "select" commands used by this proposed protocol is much less than that of the intuitive method which issues a single "select" command to pick each tag in P. Another related research is the tag grouping problem as studied in [27]–[30]. For example, the state-of-the-art grouping protocol CCG [30], can rapidly inform each tag about which group it belongs to, such that the tags in the same group end up having the same group-ID. In each communication round, CCG randomly hashes the tags who do not have their group-IDs to a number of slots, and then use homogenous slots (If the tags that are hash-mapped to a slot s belong to the same group, then s is a homogenous slot) to transmit the group-IDs for the tags that are hash-mapped to these slots. CCG uses multiple rounds to ensure that all tags are informed of their group-IDs. In [27], the authors reduce the communication time of CCG based on multiple hash functions, and propose a grouping protocol ComLab. Experimental results show that ComLab reduces the communication time by about $10\% \sim 25\%$. In [28], the authors study the tag grouping problem in a dynamic scenario, where the aim is to rapidly inform each tag about its new group-ID based on its previous group-ID. The basic grouping idea is to find the difference between the previous group-ID and the new one of a tag. Then the protocol can notify the tag to flip its bits according to the difference to join the new group.

There are two main drawbacks with the existing works as follows.

- No nontrivial lower bound is given for communication overhead: Obtaining a nontrivial lower bound is critical for real-time applications in RFID systems, due to the requirement of pre-allocating sufficient time to every task for guaranteeing efficiency and avoiding failure in these systems. A nontrivial lower bound is also critical to determine how much reduction can be expected in theory when we try to improve a protocol.
- They can not be applied to all off-the-shelf RFID systems directly: The above protocols for the group writing problem and the tag grouping problem can solve the tag-selection problem. However, they can not be applied to every RFID system directly without additional requirements. More specially, the protocols for the tag group problem require that some additional computing functions (For example, see the ROB function in [28] and the simple function that counts the number of '1's in a bit array in [27], [30]) should be implemented in the chip of each tag, which are not specified in the EPC C1G2 standard [31] and will be supported by commercial-off-the-shelf (COTS) tags.[1] The protocols for the group writing problem require a time-consuming preprocess that writes an additional bit-vector into the user memory of each tag.[2] Therefore, they can not be applied to an RFID system that contains read-only tags (read-only tags do not have any user memory into which the user-specific data can be stored).

B. CHALLENGES AND TECHNICAL CONTRIBUTIONS

This paper is targeted for designing an efficient protocol for the tag-selection that can both applied to every off-the-shelf RFID system directly, and has a communication cost close to the nontrivial lower bound. Towards this goal, we need to answer two challenging problems.

What is the nontrivial lower bound of communication overhead for a protocol which is capable of solving the tag-selection problem: The lower bound indicates the minimal amount of communication time for a protocol to solve the tag-selection problem. Deriving a nontrivial lower bound is challenging because we must thoroughly understand the nature of the tag-selection problem, and then carefully formulate a communication process to capture the underlying hardness. More specifically, to obtain a nontrivial lower bound, we must transform a protocol for the tag-selection problem into a process that can represent any tag subset and population (see Theorem 2).

How to design an efficient protocol which can be applied to all off-the-shelf RFID system directly: The inherent difficulty is how to guarantee the universal applicability and a small communication time. Here we propose to use only the tag data and commands that are universally supported in off-the-shelf RFID systems to solve the problem with a small communication time. More specifically, to achieve this, we design a protocol that uses the EPC and "select" command, both of which are supported in every RFID system.[3] To guarantee its efficiency, this protocol first analyzes the tags' EPCs carefully to build a set of filter substrings, and then smartly chooses some substrings from this built set of filter substrings, such that the tag-selection problem can be solved by issuing a few "select" commands (see Algorithm 1).

To this end, we study the efficient tag-selection problem in RFID systems with universal applicability and a small communication time. In particular, firstly, we prove a nontrivial lower bound of communication overhead for a protocol which is capable of solving the tag-selection problem. Then we propose an efficient protocol, denoted by TagSP, which uses only the "select" command and EPC. Lastly, we testify the universal applicability and efficiency of TagSP through rigorous theoretical analysis and comprehensive simulation results. The main contributions of this paper are summarized as follows:

[1] The EPC C1G2 standard [31] defines the operating procedures, commands and physical interactions between RFID readers and Tags. Any command or computing function not specified in this standard will not be supported by COTS tags.

[2] It is time-consuming because N "write" commands are needed when the tag population S contains N tags. A "write" command is used for writing a specially designed bit-vector into the user memory of a tag of S, please see page 4 of [20] for details.

[3] EPC is a tag data written in all tags, and the "select" command is a mandatory command that is universally supported in every RFID systems.

Algorithm 1 TagSP: An Efficient Method for the Tag-Selection Problem Based on Multiple Filter-Strings

```
Input: A tag population $S$ of $N$ tags chosen from $U$, and a tag subset $P$ of $n$ tags ($P \subset S$).
||* First stage: Preprocess all the substrings in the n EPCs of P to build a set $\mathbb{A}$ of candidate filter-strings.*||
1: Initialize a set $\mathbb{A} = \emptyset$;
2: Set $L_{\max} = \lceil \log_2(N^2) \rceil$;
3: For each $t \in P$
4:     For each substrings $t[i,j]$ of $t$ ($i \le j \in \{1, 2, \ldots, 96\}$)
5:         If $j - i + 1 \le L_{\max}$
6:             If $\forall u \in S - P$, $u[i,j] \neq t[i,j]$ Then
7:                 $\mathbb{A} = \mathbb{A} \cup \{t[i,j]\}$; End If   End If
8: End For  End For
||* Second stage: Use candidate filter-strings in $\mathbb{A}$ to pick subset P from S in a greedy way.*||
9: Initialize a counter $C = 0$;
10: While $P \neq \emptyset$
11:     Find a substring $t[i,j]$ from $\mathbb{A}$ such that the number of tags in $P$ which share $t[i,j]$ in their EPCs is maximized;
        ||* a tag $u \in P$ shares $t[i,j]$ in its EPC, if the substring that starts at the i-th bit and ends at the j-bit of tag u's EPC equals $t[i,j]$.*||
12:     Issue a command: Select MemBank = 1, Pointer = $i$, Length = $j - i + 1$, Mask = $t[i,j]$;
13:     Delete the tags that can be picked by the above command from $P$;
14:     $C = C + 1$;
15: End While
```

1) We have proven a nontrivial lower bound of communication overhead for a protocol which is capable of solving the tag-selection problem.
2) We have designed a protocol, denoted by TagSP, to solve the studied problem. Comparing to the existing protocols, the proposed TagSP can be applied directly into off-the-shelf RFID systems with a communication overhead close to the lower bound.
3) We have conducted extensive simulations to verify the performance of the proposed TagSP, and the results show that the communication time of TagSP is within a constant factor of the lower bound.

The remainder of this paper is organized as follows. Section II presents a nontrivial lower bound of the communication time for the tag-selection problem. Section III describes the proposed TagSP protocol. Section IV evaluates the proposed protocol through simulation results. Finally, concluding remarks are presented in Section V.

II. A NONTRIVIAL LOWER BOUND OF COMMUNICATION OVERHEAD

In this section, we obtain a nontrivial lower bound of communication overhead for any protocol to solve the tag-selection problem.

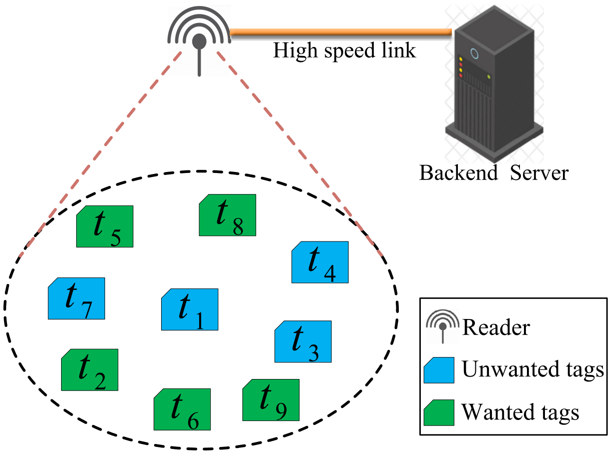


FIGURE 1. An example for the tag-selection problem (picking P from S). Note: Tag population $S = \{t_1, t_2, \ldots, t_9\}$ and subset $P = \{t_2, t_5, t_6, t_8, t_9\}$.

TABLE 1. Symbol definitions.

Symbol	Definition
U	$U = \{ \underbrace{00\cdots00}_{\vert\leftarrow\text{96-bit}\rightarrow\vert}, \underbrace{00\cdots01}_{\vert\leftarrow\text{96-bit}\rightarrow\vert}, \underbrace{00\cdots10}_{\vert\leftarrow\text{96-bit}\rightarrow\vert}, \cdots\cdots, \underbrace{11\cdots11}_{\vert\leftarrow\text{96-bit}\rightarrow\vert} \}$ is the set of all the possible EPCs (tag-IDs)
S	A tag population of N EPCs (tags) chosen from U.
P	A tag subset of n EPCs chosen from S, $P \subset S$.
N	The number of EPCs in S, i.e., $\vert S\vert = N$.
n	The number of EPCs in P, i.e., $\vert P\vert = n$.
t	t is denoted as both tag t and t's EPC, $t \in U$.
$t[i,j]$	The substring that starts at the i-th bit and ends at j-th bit of t.
T_s	The time unit for sending a 96-bit string to tags.
A	A protocol that solves the tag-selection problem.
T_{BD}	The lower bound of time for a protocol which is capable of solving the tag-selection problem, see Theorem. 2.2.

A. SYSTEM MODEL AND PROBLEM DEFINITION

We consider an RFID system where a reader R is deployed to monitor a set S of N tags, i.e., $S = \{t_1, t_2, \cdots, t_N\}$ (we call S as a tag population). Fig. 1 shows such an example with a reader R and a set S of 9 tags. The reader R is connected to a powerful backend server for computation and storage. Each tag t has a unique EPC (a 96-bit ID) which exclusively represents tag t as well the object to which t is attached. The reader R communicates with the server through a high speed connection link, over which the communication time delay can be ignored. All the tags in S stay within the interrogating range of R and can communicate with R via a low-speed connection link, over which the communication overhead mainly determines the communication time of a protocol in RFID systems.

Before we present the definition of the tag-selection problem, we define a few notations. Let U denote the set of all 2^{96} possible EPCs. Let S denote a tag population which is a set of unique EPCs (tags) chosen from U, and let $|S| = N$. Let P be a proper subset of S, and $|P| = n$. We use t to denote both tag t and t's EPC. Some common symbols used in this paper are also summarized in Table 1.

Initially, all the tags in S have no idea about whether they belong to subset P or not, and reader R knows the tags' EPCs in P and $S-P$, respectively. Then, the tag-selection problem is defined as follows.

Definition 1: Suppose we have an RFID system which is described in the above, for any $S \subset U$ and $P \subset S$, the tag-selection problem requires to devise a protocol to guarantee that reader R can pick all the tags in P from S, i.e., to inform all tags in S whether they belong to P or not.

Please note that, in Definition 1, S can be any subset of N different EPCs chosen from U because we make no restriction on the tags' EPCs in an RFID system; P is allowed to be any proper subset of S because we do not assume any distribution of P. In practice, this is quite possible to happen, especially in a large scale RFID system where users or applications partition a large number of tags into different subsets according to various selection criteria or local attributes of tags at a time point, and want to pick different subsets from S later at different time points.

B. LOWER BOUND ANALYSIS

In the following theorem, we obtain a nontrivial lower bound of the number of bits that reader R sends to tags, and then accordingly a nontrivial lower bound of time, denoted by $\mathrm{T_{BD}}$. The basic idea is to transform a protocol A for the tag-selection problem and a specially designed message, denoted by B, into a process that can represent a subset P and a population S. Then the lower bound of A can be derived by calculating the difference between $\{P, S\}$ and B. Please note that in the following analysis, we assume that the communication time of a protocol in RFID systems is mainly determined by the time used for sending bits from reader R to tags. We ignore the cost of communications between R and the server. Because R can communicate with the server quickly via a high speed connection link, in which the data speed is usually larger than 10Mbits/s.[4] In contrast, the data speed from reader R to all tags is only about 26Kbits/s~128Kbits/s [31].

Let $\mathrm{T_s}$ represent the time length for sending a 96-bit string from reader R to all tags. Then R needs a time of $\frac{m\times \mathrm{T_s}}{96}$ to transmit an m-bit message to tags, because R can spilt this message into 96-bit segments and transmit each of them in a time slot of length $\mathrm{T_s}$. Therefore, if a protocol A needs to transmit M bits in total to tags, the time of A equals $\frac{\mathrm{M}\times \mathrm{T_s}}{96}$.

Theorem 2: Let A be a protocol which is capable of solving the tag-selection problem, and $|A|$ be the total number of bits sent to all tags when A runs. Then A must satisfy the inequality (1) and equation (2):

$$|A| \geq \zeta = n\log_2\binom{N}{n} + (N-n)\log_2\left(\frac{N}{N-n}\right), \quad (1)$$

$$\mathrm{T_{BD}} = \frac{\zeta \times \mathrm{T_s}}{96}, \quad (2)$$

where n and N are the numbers of tags in P and S, respectively, and $\mathrm{T_{BD}}$ represents the lower bound of time.

Proof: We use Msg to represent all of the bits that reader R sends to all tags when protocol A runs. We can view Msg as a single bit-string in which the number of bits equals $|A|$. During the execution of protocol A, it is easy to see that all tags within the interrogating range of reader R use a common decision procedure to decide whether they belong to subset P or not. Because, initially, each tag $t \in S$ knows only its EPC and stays in the same state (non-selected state), and then makes decisions during the execution of protocol A. This common decision procedure can be represented by a function $F()$. Without loss of generality, we can assume that, after reader R broadcasts Msg, each tag $t \in S$ computes $F(t, Msg)$, and makes decisions as follows:

(a) If $F(t, Msg) = 1$, tag t decides that it is in P;
(b) If $F(t, Msg) = 0$, tag t decides that it is not in P.

Because, protocol A is assumed to be capable of solving the tag-selection problem, then by the Definition 1, we have the followings three statements:

(1) If $x \in P$, $\mathbf{Pr}[F(x, Msg) = 1] = 1$;
(2) If $x \in S-P$, $\mathbf{Pr}[F(x, Msg) = 0] = 1$;
(3) If $x \in U-S$, $\mathbf{Pr}[F(x, Msg) = 1] \in [0, 1]$.

Please note that $\mathbf{Pr}[]$ stands for probability. The first two statements are easy to be obtained from Definition 1. The third statement is because Definition 1 does not make any restriction on the result of $F(t, Msg)$, if $t \in U-S$.

We show how each tag t can know which n EPCs are in subset P and which $N-n$ EPCs are in $S-P$ in the following steps:

S1: Since U is known to all tags, each tag $t \in S$ is capable of computing $F(x, Msg)$ over each EPC $x \in U$. Then t can get two sets: $U_0 = \{x, |x \in U \text{ and } F(x, Msg) = 1\}$ and $U_1 = \{x, |x \in U \text{ and } F(x, Msg) = 1\}$. By statement (1)-(3), we know $P \subset U_1$ and $S-P \subset U_0$ (the value of $F()$ over an EPC in $U-S$ can be either 1 or 0). Therefore, tag t can know which n EPCs from U truly belong to P, if reader R tells t which EPCs in U_1 are not in P; t can know which $N-n$ EPCs from U truly belong to $S-P$, if R tells t which EPCs in U_0 are not in $S-P$.

S2: Reader R can get U_0 and U_1 the same way as each tag t does. Then, since reader R knows P and $S-P$, it can get $U_0-(S-P)$ and U_1-P. Hence reader R can tell each tag t about the EPCs in $U_0-(S-P)$ and U_1-P by broadcasting a message B to all tags.

S3: Each tag $t \in S$ can perfectly know which n EPCs are in P as well as which $N-n$ EPCs are in $S-P$, after receiving B.

We need to use $\log_2\binom{\Psi-n}{N-n} + \log_2\binom{\Psi}{n}$ bits to accurately represent a subset P and a population S ($|P| = n$ and $|S| = N$), where $\Psi = 2^{96}$. This is because an arbitrary set containing N different EPCs from U can become S, and an arbitrary set containing n EPCs from S can become P. Let $|U_0| = z$ (the number of EPCs in U_0 is z). We have $|U_1| = \Psi - z$, and $N-n \leq z \leq \Psi - n$, because statement (3) allows an arbitrary number of EPCs in $U-S$, over each of which the value of $F()$ equals 1, and statement (2) requires: $\forall x \in S-P, F(x, Msg) = 0$. Then, there are $\binom{z}{N-n}$ different

[4] As a typical example, ZEBRA FX7500 RFID READER and IMPINJ SPEEDWAY REVOLUTION R220 UHF RFID READER use Ethernet cable to join a network and connect to the backend server.

TABLE 2. **The 8 actions of a `Select` command.**

Action	Tag Matching	Tag Not-Matching
"000"	assert **SL** or **inventoried** → A	deassert **SL** or **inventoried** → B
"001"	assert **SL** or **inventoried** → A	do nothing
"010"	do nothing	deassert **SL** or **inventoried** → B
"011"	negate **SL** or (A → B, B → A)	do nothing
"100"	deassert **SL** or **inventoried** → B	assert **SL** or **inventoried** → A
"101"	deassert **SL** or **inventoried** → B	do nothing
"110"	do nothing	assert **SL** or **inventoried** → A
"111"	do nothing	negate **SL** or (A → B, B → A)

ways of choosing $N-n$ EPCs from U_0, and $\binom{\Psi-z}{n}$ different ways of choosing n EPCs from U_1. As a result, we know $|B|$ (the number of bits used in B) equals $\log_2\binom{z}{N-n}+\log_2\binom{\Psi-z}{n}$ bits.

Since Step(1)-(3) can help a tag t to accurately find out P and $S-P$, we have the following inequality:

$$\begin{aligned} |A| &\geq [\log_2\binom{\Psi-n}{N-n}+\log_2\binom{\Psi}{n}]-[\log_2\binom{z}{N-n}+\log_2\binom{\Psi-z}{n}] \\ &\geq \log_2(\frac{(\Psi-n)^{\underline{N-n}}}{z^{\underline{N-n}}})+\log_2(\frac{\Psi^{\underline{n}}}{(\Psi-z)^{\underline{n}}}). \qquad (3) \end{aligned}$$

Let $z=\alpha\Psi$, where $\alpha\in[0,1]$. By substituting z with $\alpha\Psi$ into (3), and we have:

$$\begin{aligned} |A| &\geq \log_2(\frac{\Psi^{\underline{n}}}{((1-\alpha)\Psi)^{\underline{n}}})+\log_2(\frac{(\Psi-n)^{\underline{N-n}}}{(\alpha\Psi)^{\underline{N-n}}}) \\ &\approx \log_2(\frac{\Psi^{\underline{n}}}{((1-\alpha)\Psi)^{\underline{n}}})+\log_2(\frac{(\Psi)^{\underline{N-n}}}{(\alpha\Psi)^{\underline{N-n}}}) \qquad (4) \\ &\geq \log_2(\frac{1}{1-\alpha})^n+\log_2(\frac{1}{\alpha})^{(N-n)}. \qquad (5) \end{aligned}$$

Please note that the approximation ($\Psi-n\approx\Psi$) used in (4) comes from the fact that Ψ is far larger than n. It is easy to see the function of α shown in (5) is minimized when $\alpha=(N-n)/n$. Therefore, we get (1).

Finally, since T_s is the time length of a slot for sending a 96-bit string from R to all tags, and protocol A needs to transmit at least ζ bits, we know that the time of A is no less than $\zeta\times T_s/96$. ■

III. TAGSP: AN EFFICIENT PROTOCOL FOR THE TAG-SELECTION PROBLEM

In this section, we study how to efficiently solve the tag-selection problem based on the EPC and "select" command. We first explain the basic working principle of using the EPC and "select" command to pick tags from a population. Then, we analyze the possibility of solving the tag-selection problem by using a single "select" command. At the end of this section, we provide the designed protocol TagSP with a detailed analysis of its theoretical performance.

A. ILLUSTRATION FOR `SELECT` COMMAND

EPC C1G2 standard defines a mandatory command "select", represented by `Select` in the following. This command provides the RFID reader with the ability to pick up multiple tags based on user-defined criteria (see pages 74-76 in [31]). A `Select` command contains 6 parameters as shown in the following:

- Target and Action: The parameter Target specifies the object that a `Select` command will change. This object can either be the selected flag, denoted by **SL**, or the **inventoried** flag of a tag. Action specifies how to change the object. According to the EPC C1G2 standard [31], there are 8 different actions for a tag to response to an Action parameter, which are shown in Tab. 2. In summary, these two parameters together specify: if a tag t matches the selection criterion, t sets its flag (**inventoried** or **SL**) to a value, otherwise, t sets its flag to another value. For example, when Target = **SL** and Action = "000″, tag t asserts its **SL** if t matches the selection criterion; tag t desserts its **SL** flag, otherwise. The selection criterion used in a `Select` command is defined by the other four parameters shown below.
- MemBank, Pointer, Length, and Mask: These four parameters jointly specify a contiguous sequence of bits that starts at Pointer and ends Length bits later in the memory of MemBank. Then the selection criterion used in a `Select` command is defined as: if the substring that starts at Pointer and ends Length bits later in the memory of MemBank is the same as MASK, the corresponding tags are matched. There are 4 different memory banks in a tag t: MemBank-0 stores t's access password; MemBank-1 stores t's EPC; MemBank-2 stores t's manufacturing information; MemBank-3 stores t's customized information. For example, when MemBank = MemBank-0, tag t uses Pointer, Length and MASK to locate a substring in its EPC, and then compare this substring with MASK to determine whether it matches or not. In other words, MASK is compared with the substring that starts at the Pointer-th bit and ends at the (Pointer+Length-1)-th bit of tag t's EPC. When MemBank takes other values, tag t finds the substring in the corresponding memory bank and compares this substring with MASK in the same way as we have explained for MemBank = MemBank-0.

In this paper, we only consider the case that MemBank = MemBank-0, because we focus on using tags' EPCs to solve the tag-selection problem. For ease of illustration, we call the substring specified by the three parameters: Pointer, Length, and Mask, as the filter-string in a `Select` command, because it actually determines the selection criterion for this `Select` command. Please note that, in the following analysis, we omit Target and Action in a `Select` command, because they only specify the action of a tag when it matches (or does not match) the filter-string.

In the following, we provide examples for showing how `Select` commands work for picking tags from a tag population S. We assume that an EPC is a 4-bit string, and consider a specific tag population $S=\{t_i|i=1,2,\ldots,8\}$ with their EPCs as shown in Tab. 3. If reader R wants to pick a tag subset

TABLE 3. The 4-bit EPCs (tag-IDs) of 8 tags.

Tag	4-bit EPC	Tag	4-bit EPC	Tag	4-bit EPC
t_1	0 0 0 0	t_4	0 1 1 0	t_7	1 1 0 0
t_2	0 0 1 0	t_5	1 0 0 0	t_8	1 1 1 0
t_3	0 1 0 0	t_6	1 0 1 0		

$P = \{t_1, t_2, t_3, t_4\}$, it can issue four `Select` commands (one for each tag), which are shown as follows:

(1) `Select` MemBank = 1, Pointer = 1, Length = 4, Mask = "0000"; (for picking t_1 from S);

(2) `Select` MemBank = 1, Pointer = 1, Length = 4, Mask = "0010"; (for picking t_2 from S);

(3) `Select` MemBank = 1, Pointer = 1, Length = 4, Mask = "0100"; (for picking t_3 from S);

(4) `Select` MemBank = 1, Pointer = 1, Length = 4, Mask = "0110" (for picking t_4 from S).

According to EPC C1G2 standard, reader R needs to transmit at least $4 \times (3 + \lceil \log_2(96) \rceil + 8 + 4) = 88$ bits for executing these four commands.[5] Please note that each of these 4 commands uses a different filter-string. However, since reader R knows the EPCs in S, it can analyze the 4 EPCs in P carefully and find out that t_1, t_2, t_3 and t_4 share a common substring "0″ in the first bit of their EPCs, which is not shared by the tags in $S - P$. Hence, reader R can find out that a single command: `Select` MemBank = 1, Pointer = 1, Length = 1, Mask = "0″ is capable of picking P from S, and it needs only $1 \times (3 + \lceil \log_2(96) \rceil + 8 + 1) = 19$ bits for executing this command.[6]

If reader R wants to pick $P = \{t_1, t_2, t_3, t_7\}$, instead of using four `Select` commands (one for each tag in P), R can issue two `Select` commands, after it finds out:

(1) t_1 and t_2 share a common substring "00″ in the first two bits in their EPCs, which is not shared by other tags in $S - P$;

(2) t_3 and t_7 share a common substring "10″ in the second and third bits in their EPCs, which is not shared by other tags in $S - P$.

Furthermore, we can see that when $P = \{t_1, t_2, t_3, t_7\}$, reader R needs at least two `Select` commands to pick P from S. This is because these four tags do not share a common substring in their EPCs, which is not shared by the tags in $S - P$.

Based on the above examples, we can see that the key point for solving the tag-selection problem is to find λ filter-strings from the EPCs in subset P, such that P can be picked by using λ `Select` commands. In the following section, firstly, we show that the probability that a single filter-string (or equivalently, a single `Select` command) can pick P from S

[5] According to Table 6-29, we know that parameter MemBank, Pointer, and Length need 3, $\lceil \log_2(96) \rceil$, and 8 bits, respectively. The number of bits needed in Mask is equal to the value of Length, which is the number of bits in the filter-string.

[6] The filter-string in this command is defined by three parameters: Pointer = 1, Length = 1, Mask="0″, and this filter-string can separate t_1, t_2, t_3 and t_4 from t_5, t_6, t_7, t_8 (t_1, t_2, t_3 and t_4 match this filter string, but t_5, t_6, t_7, t_8 do not)

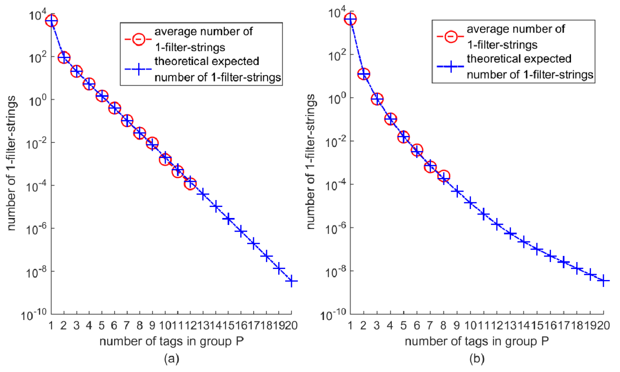


FIGURE 2. Expected numbers of 1-filter-strings for randomly generated S and P: (a) vary $|P|$ from 1 to 20 and set $|S| = 2|P|$; (b) vary $|P|$ from 1 to 20 and set $|S| = |P| + 20$.

is extremely small; secondly, we propose an efficient protocol that picks P from S by using multiple filter-strings.

B. ANALYSIS FOR SOLVING THE TAG-SELECTION PROBLEM BY USING A SINGLE SELECT COMMAND

A single "select" command can pick P from S, if there exists a substring $t[i, j]$ of tag t's EPC ($1 \le i \le j \le 96$), such that the following two statements are true:

(S1) $\forall u \in P$, $u[i, j] = t[i, j]$ ($t[i, j]$ is shared by every tag in P);

(S2) $\forall u \in S - P$, $u[i, j] \neq t[i, j]$ ($t[i, j]$ is not shared by the tags in $S - P$).

If such a substring exists, then we can use $t[i, j]$ to pick P from S by using a single "select" command: "`Select` MemBank = 1, Pointer = i, Length = $j - i + 1$, Mask = $t[i, j]$". We call the substring $t[i, j]$ a 1-filter-string.

We first conduct simulations to find the average numbers of 1-filter-string solutions for given n and N through 10^5 independent trails, where $n = |P|$ and $N = |S|$. The simulations are done with two settings. One is to set $N = 2n$, and let n go from 1 to 20. The other is set $N = n + 20$, and let n go from 1 to 20. Specifically, we takes the following steps for each setting:

(1) Pick the number n;

(2) Randomly choose N different EPCs from U to form set S and then randomly choose n EPCs from S to form set P;

(3) Do exhaustive enumeration to find all the 1-filter-strings for the sets: S and P which are determined in step (2), and record this number;

(4) Repeat step (2) and (3) for 10^5 times to compute the average number of 1-filter-strings for the number n.

Please note that the average number we get in the above procedure provides a simulated upper bound of the probability of existence of 1-filter-string solution for a given parameter n. Fig. 2 shows the average number of 1-filter-string solutions for the two settings. From the results, we can observe that the average number of 1-filter-string solutions decreases close to 0 rapidly, with the increase of n, which also indicates the probability of 1-filter-string solution decreases close to 0 quickly with the increase of n. In particular, when $n = 6$ and $N = 12$ (both P and $S - P$ contain 6 tags), the average number of 1-filter-string solutions in a trial is only 0.4, which means that in there are at least $10^5 \times (1 - 0.4) = 6 \times 10^4$ trials, in each of which there does not exist a 1-filter-string solution; when $n = 10$ and $N = 20$, the average number of 1-filter-string solutions is only 0.0016, which means that there are at least $10^5 \times (1 - 0.0016) = 99984$ trials, in each of which no 1-filter-string solutions can be found.

Secondly, we calculate the theoretical expected number of 1-filter-string solutions, denoted by $\mathbb{N}_1$, and derive an upper bound of the probability that there exists at least a 1-filter string for a trial, as stated in Theorem 3.

Theorem 3: Let S denote a tag population of N unique EPCs randomly chosen from U (the set of all possible EPCs), and P a subset of n EPCs randomly chosen from S. Then the expected number of 1-filter-strings can be calculated as:

$$\mathbb{N}_1 = n \sum_{i=1}^{96} (96 - i + 1) \left(1/2^i\right)^{n-1} \left(1 - 1/2^i\right)^{N-n}. \quad (6)$$

Note that formula (6) provides an upper bound of the probability that there exists at least one 1-filter-string for picking P from S.

Proof: The proof is shown in the Appendix. ■

We plot the curves of formula (6) in Fig. 2 (see crosses) which highly match the simulation results (see cycles).

C. SOLVING THE TAG-SELECTION PROBLEM BY USING MULTIPLE `SELECT` COMMANDS

Based on the previous analysis, we can see that 1-filter-string solution rarely exists for a tag-selection problem. Therefore, we deal with the tag-selection problem based on multiple filter-strings, or equivalently, multiple `Select` commands.

The basic idea of the designed protocol TagSP is to build a set $\mathbb{A}$ of candidate filter-strings such that the number of `Select` commands used for picking P can be reduced. More specifically, TagSP includes two stages: the first stage analyzes all the EPCs in subset P to build $\mathbb{A}$, and the second stage uses $\mathbb{A}$ to pick P from S in a greedy way, which always uses the substring in $\mathbb{A}$ that is shared by the largest number of tags in P as the filter-string in each issued `Select` command. The detailed steps of TagSP are shown in Algorithm 1.

In TagSP, the first stage is shown in Step 1-8, and the second stage is shown in Step 9-15. The variable $L_{\max}$ set in step 2 is used for reducing the number of substrings that we need to check when constructing $\mathbb{A}$. The reason for using $\lceil \log_2(N^2) \rceil$ as the maximal length is shown in Theorem 4. Step 4-7 check each substring $t[i, j]$, and then put $t[i, j]$ into $\mathbb{A}$ if there does not exist a tag $u \in S - P$ that shares $t[i, j]$ in its EPC. Step 11 finds the substring $t[i, j]$ in $\mathbb{A}$ that is shared by the largest number of tags in P; Step 12 issues a `Select` command by using the substring $t[i, j]$ found in Step 11 as the filter-string to pick some tags in P from S. From these two steps, we can see that TagSP tries to minimize the number of `Select` commands by adopting a greedy strategy. Please note that we do not try to find an optimal subset of filter-strings that can pick P from S by issuing a minimal number of `Select`s. The reason is that there are exponentially many different subsets of $\mathbb{A}$ needed to be considered when finding the optimal subset.

Theorem 4: Let S denote a tag population of N unique EPCs randomly chosen from U, and P a subset of n EPCs randomly chosen from S. Furthermore, let $\mathbb{A}$ be the set of candidate filter-strings generated by step 1-8 in Algorithm 1. Then we have the following two statements:
(A) $\forall t \in P$, the probability that there exists a filter-string $t^\star[i, j] \in \mathbb{A}$ such that $t[i, j] = t^\star[i, j]$ is nearly 1;
(B) The number of filter-strings in $\mathbb{A}$ is no more than $192 \times n \log_2(N)$.

Proof: Let $t[i, i+L_{\max}-1]$ represent the substring of t's EPC that starts at the i-th bit and ends at the $(i + L_{\max} - 1)$-th bit, where $1 \leq i \leq 97 - L_{\max}$. In other words, $t[i, j]$ is a substring of length $L_{\max}$. By Step 6 in Algorithm 1, we know that $t[i, j]$ will be put into $\mathbb{A}$ if and only if there does not exist a tag $u \in S - P$ such that $u[i, i+L_{\max}-1] = t[i, i+L_{\max}-1]$. Because the tags in S are randomly chosen from U (the set of all possible EPCs), we have the following:

$$\mathbf{Pr}\left(u[i, i + L_{\max} - 1] = t[i, i + L_{\max} - 1]\right) = \frac{1}{2^{L_{\max}}} \leq \frac{1}{N^2}.$$

Thus, the probability that $t[i, j]$ is put into $\mathbb{A}$ is $(1 - 1/N^2)^{N-n} > 1 - 1/N$. Since there are at least $\lfloor 96/L_{max} \rfloor$ non-overlapped substrings of length L_{max} in t's EPC, we know the probability that at least one of these $\lfloor 96/L_{max} \rfloor$ substrings is put into $\mathbb{A}$ is larger than $1 - (1/N)^{\lfloor 96/L_{max} \rfloor} \approx 1$. Because N is usually large in RFID systems, we can have statement (A). Please note that this explains why we set $L_{\max} = \lceil \log_2(N^2) \rceil$, i.e., we do not consider the substrings of length larger than $L_{\max}$. The reason is that the substrings of length no more than $L_{\max}$ already can help us to pick each tag in P.

Statement (B) follows from the fact that there are $96 - l + 1 \leq 96$ different substrings of length l in a tag's EPCs, and the first stage of TagSP only considers the substrings of length less than or equal to L_{max}. ■

Theorem 5: Let S denote a tag population that contains N unique EPCs randomly chosen from U, and P denote a subset of n EPCs randomly chosen from S. Then the computational cost of TagSP is $O(nN \log_2(N))$.

Proof: Given an integer $l \in \{1, 2, .., 96\}$, there are at most 96 different substrings of length l in a tag t's EPC. Then we can see: at most $96 \times (N - n)$ substring comparisons are needed for a substring $t[i, j]$. This is because, a substring $t[i, j]$ is compared with the substring $u[i, j]$ of each tag $u \in S - P$, in the worst case (when $t[i, j]$ is put into $\mathbb{A}$). Furthermore, since we need to check the substrings of length no more than $L_{max} \leq \lceil \log_2(N^2) \rceil$, we know at most $n \times 96 \times (N - n) \times L_{max} = O(n(N - n) \log_2(N))$ substring comparisons are needed for the first stage.

Next, we look at the second stage of TagSP. By Theorem 4, we know that there are at most $192 \times n \log_2(N)$ filter-strings

TABLE 4. Number of bits used in the 6 parameters of a `Select` command. Note: This table is set according to 6-29 in page 76 of EPC C1G2 standard [31].

	Target	Action	MemBank	Pointer	Length	Mask
# of bits	3	3	2	8	8	Variable

in $\mathbb{A}$. Then, it is easy to see that at most $192 \times n \log_2(N)$ comparisons are needed for finding the filter-string that is shared by the largest number of tags of P in Step-11.[7] Obviously, at most n filter-strings are needed for picking P from S. Therefore, the second stage of TagSP needs at most $192 \times n \log_2(N) \times n = O(n^2 \log_2(N))$ comparisons.

Finally, since $O(n(N - n)\log_2(N)) + O(n^2 \log_2(N)) = O(nN \log_2(N))$, we have the conclusion. ■

IV. SIMULATION RESULTS

In this section, we compare the simulation performance of TagSP with the state-of-the-art protocol WB [20].

A. SIMULATION SETTING

We compare the performance of the two protocols: TagSP and WB. The timing scheme stated in the EPC C1G2 standard [31] is used as the basic unit to compute the communication time of the two protocols. The transmission rate from reader R to tags is set to 26.7Kbits/s. Any two consecutive transmissions are separated by a time slot of 302μs. Therefore, it takes reader R 3897.2μs to transmit a 96-bit string to tags (0.096Kbits/26.7Kbits/s+$302\mu s = 3897.2\mu s$). Please note that other transmission rates can only change the absolute communication time of protocols, and do not affect the trend. The communication time of a `Select` command is determined by the time for transmitting its 6 parameters Target, Action MemBank, Pointer, Length, and Mask, according to Table 4. Because WB needs a preprocess in which each tag of S is written with a $\log_2(2\,N)$-bit string, the communication time of WB equals the sum of the time for transmitting the n $(\log_2(2\,N))$-bit strings during the preprocess, and the time for transmitting `Select` commands. The computational cost of TagSP and WB is measured by implementing these two protocols in Matlab R2014b on a laptop PC with Intel I7-7700HQ CPU and 8GB RAM, running Windows 7 (64-bit).

B. PROTOCOL PERFORMANCE

The performance of the two protocols is evaluated under various settings of N and n, which are the numbers of tags in S and P, respectively. We compare communication time and computational cost of TagSP and WB in 3 scenarios. In scenario 1, we set $N = 10^2$ and vary n from 10 to 50. In scenario 2, we set $n = 100$ and vary N from 200 to 10^3. In scenario 3, we vary n from 100 to 500 and set $N = 2 \times n$. For each scenario, the simulation results of these

[7]For two filter-string $t_1[i_1, j_1], t_2[i_2, j_2] \in \mathbb{A}$, we need to compare the number of the tags in P that share $t_1[i_1, j_1]$ with the number of the tags in P that share $t_2[i_2, j_2]$.

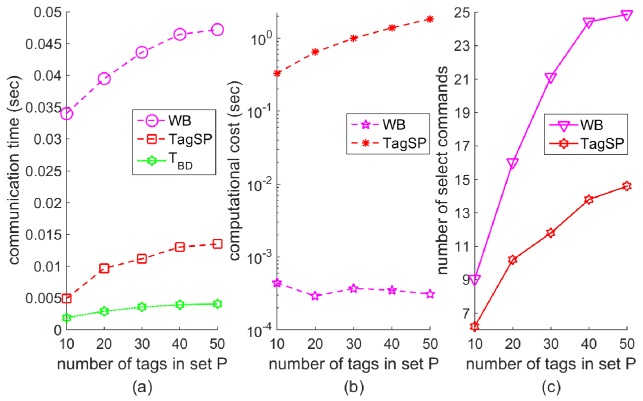


FIGURE 3. Scenario 1: vary |*P*| from 10 to 50 and set |*S*| = 100. (a) Communication time. (b) Computational cost. (c) Number of `Select` commands.

two protocol are the average of 100 independent trails. More specifically, for fixed values of n and N, we take the following steps:

(1) Generate a tag population S by randomly choosing N EPCs from U, and then a subset P by randomly choosing n EPCs from S;

(2) Use TagSP and WB to pick P from S, and record their simulation results (communication time, computational cost, and the number of used `Select` commands);

(3) Repeat (1) and (2) for 100 times, and use the averaged values of 100 independent trails as the performance of these two protocols. We also compute T_{BD} with various values of n and N according to (2) in Theorem 2.

In Fig. 3, we display the simulation results for scenario 1. We can see that both the communication time and the number of `Select` commands of TagSP are much less than that of WB. In particular, when $n = 50$ and $N = 100$, the communication time of TagSP and WB are 0.0135s and 0.047s, respectively; the number of `Select` commands used by TagSP and WB are 14 and 25, respectively. Hence, we can see that the communication time of TagSP is about 28% of that of WB. This is because TagSP uses much less `Select` commands as compared with WB, and WB needs a time-consuming preprocess while TagSP does not. In Fig. 3, we also see that the communication time of TagSP is at most 3.5 times of T_{BD}.

The simulation results for scenario 2 and 3 are shown in Fig. 4 and Fig. 5, respectively, where we can observe similar results as in Fig. 3. In particular, in Fig. 4, when $n = 100$ and $N = 1000$, the communication time of TagSP and WB are 0.059s and 0.48s, respectively; the number of `Select` commands used by TagSP and WB are 62 and 90, respectively. Hence, we can see that the communication time of TagSP is about 13% of that of WB. In Fig, 5, when $n = 500$ and $N = 800$, the communication time of TagSP and WB are 0.12s and 0.475s, respectively; the number of `Select` commands used by TagSP and WB are 116 and 200, respectively. Hence, we can see that the communication time of TagSP is about 25% of that of WB. In Fig. 4 and 5, the communication time of TagSP is at most 3.8 times of the lower bound. The communication time of TagSP can get close

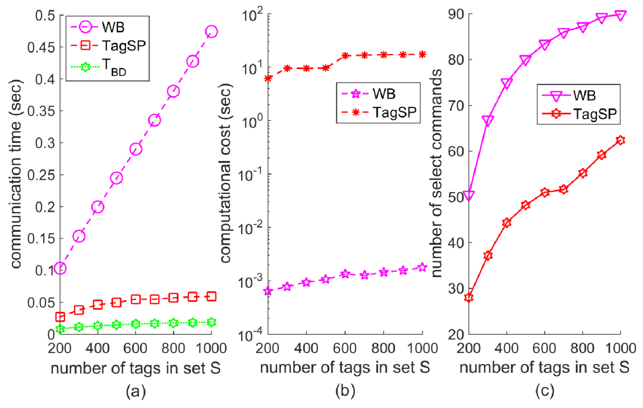


FIGURE 4. Scenario 2: vary $|S|$ from 200 to 1000 and set $|P| = 100$. (a) Communication time. (b) Computational cost. (c) Number of `Select` commands.

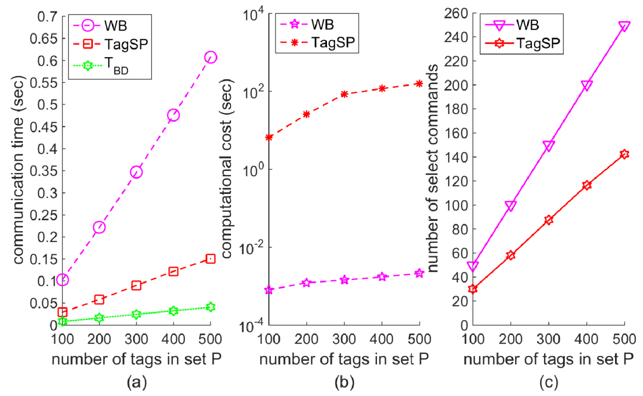


FIGURE 5. Scenario 3: vary $|P|$ from 100 to 500 and set $|S| = 2 \times |P|$. (a) Communication time. (b) Computational cost. (c) Number of `Select` commands.

to the lower bound, because $\mathbb{A}$ (the set of candidate filter-strings) contains the best substrings that can be used to pick tags in P in `Select` commands. However, due to redundant information transmitted in `Select` commands, the ratio between the communication time of TagSP and the lower bound can not equal to 1 approximately. Please note that the transmitted parameters: Target, Action and MemBank remain unchanged during the execution in TagSP (see Algorithm 1), and there are some redundancy in the 3 parameters: Pointer, Length and MASK (MASK already contains the information about the length of the filter-string).

The computation burden of both schemes is not very heavy despite of using the normal desktop computer and can be satisfied when a powerful computing server is available in RFID systems, although the computational cost of TagSP is higher than that of WB in Fig. 3-5.

V. CONCLUSION

In this paper, we study the tag-selection problem, which is a basic and under-investigated problem in RFID systems. We prove a nontrivial lower bound of communication overhead for a protocol which is capable of solving the studied problem. Then we design an efficient protocol TagSP for this problem only based on the EPC and the `Select` command from EPC C1G2 standard. Compared with the existing protocols, the proposed TagSP has a much less communication time and can be directly applied into off-the-shelf RFID systems. Extensive simulation are conducted and the results verify the superiority of TagSP.

APPENDIX

PROOF OF THEOREM 3.1

Firstly, we can calculate the expected number of filter-strings in a trial, where P and S are randomly generated. Without loss of generality, we assume that $P = \{t_1, t_2, .., t_n\}$ and $S = P \cup \{t_{n+1}, t_{n+2}, \ldots, t_N\}$. We use t to represent tag t as well as t's EPC, and $t[i, j]$ to represent the substring of t's EPC that starts at the i-th bit and ends at the j-th bit, $1 \le i \le j \le 96$. For each EPC t_k in the tag population S, and each integers $i, j \in \{1, 2, \ldots, 96\}$ $(i \le j)$, let $X^k_{i,j}$ be a random variable such that

$$X^k_{i,j} = \begin{cases} 1, & \text{If } \forall t_{k'} \in P,\ t_{k'}[i,j] = t_k[i,j] \\ & \quad \text{and If } \forall t_{k'} \in S - P,\ t_{k'}[i,j] \neq t_k[i,j], \\ 0, & \text{Otherwise.} \end{cases}$$

Then, $X^k_{i,j} = 1$ represents the event that the substring $t_k[i, j]$ is a 1-filter-string for selecting tags in P from S, and

$$\sum\nolimits_{k=1}^{n} \sum\nolimits_{i=1}^{96} \sum\nolimits_{j=i}^{96} X^k_{i,j} \tag{7}$$

represents the total number of 1-filter-strings in a trail. Note that, in a trial, we generate a tag population S by randomly choosing N different EPCs from U the set of all 2^{96} possible EPCs. This is equivalently to the process of N random selections, in each of which an EPC is chosen independently and uniformly from U.[8] Then we get:

$$\mathbf{Pr}\left(t_{k'}[i,j] = t_k[i,j]\right) = 1/2^{j-i+1}, \quad \forall k' \neq k \in \{1, 2, .., N\}.$$

Since P and $S - P$ contain n and $N - n$ EPCs, respectively, the probability that $X^k_{i,j} = 1$ can be obtained as follows:

$$\mathbf{Pr}\left(X^k_{i,j} = 1\right) = \left(1/2^{j-i+1}\right)^{n-1} \times \left(1 - 1/2^{j-i+1}\right)^{N-n}.$$

Based on the above probabilities, now we can compute the mathematical expectation of $\sum_{k=1}^{n} \sum_{i=1}^{96} \sum_{j=i}^{96} X^k_{i,j}$ shown in (7) as follows.

$$\begin{aligned} E&\left[\sum\nolimits_{k=1}^{n} \sum\nolimits_{i=1}^{96} \sum\nolimits_{j=i}^{96} X^k_{i,j}\right] \\ &= \sum\nolimits_{k=1}^{n} \sum\nolimits_{i=1}^{96} \sum\nolimits_{j=i}^{96} E\left[X^k_{i,j}\right] \\ &= \sum\nolimits_{k=1}^{n} \sum\nolimits_{i=1}^{96} \sum\nolimits_{j=i}^{96} \left(1/2^{j-i+1}\right)^{n-1} \\ &\quad \times \left(1 - 1/2^{j-i+1}\right)^{N-n} \\ &= n \sum\nolimits_{i=1}^{96} (96 - i + 1) \\ &\quad \times \left(1/2^{i}\right)^{n-1} \times \left(1 - 1/2^{i}\right)^{N-n}. \end{aligned} \tag{8}$$

[8] S can be generated by N selections, in each of which an EPC is chosen independently and uniformly from U. Because $2^{96} \approx 7.9 \times 10^{28}$ and $N \ll 2^{96}$, the probability that any two of the N EPCs are identical can be omitted.

The first equality above is based on the linearity of mathematical expectation.

Secondly, the event that there exists at least one filter-string in a trial is equivalent to $\sum_{k=1}^{n}\sum_{i=1}^{96}\sum_{j=i}^{96} X_{i,j}^{k} \geq 1$. Then, by Markov inequality and (8), we have the conclusion.

REFERENCES

[1] L. Zhang, W. Xiang, X. Tang, Q. Li, and Q. Yan, "A time- and energy-aware collision tree protocol for efficient large-scale RFID tag identification," *IEEE Trans. Ind. Informat.*, vol. 14, no. 6, pp. 2406–2417, Jun. 2018.

[2] 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.

[3] 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.

[4] 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.

[5] L. Zhang, W. Xiang, I. Atkinson, and X. Tang, "A time-efficient pair-wise collision-resolving protocol for missing tag identification," *IEEE Trans. Commun.*, vol. 65, no. 12, pp. 5348–5361, Dec. 2017.

[6] 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.

[7] X. Wang, Z. Liu, Y. Yang, X. Shao, Y. Gu, and S. Ishihara, "Approximate range emptiness in constant time for IoT data streams over sliding windows," in *Proc. 28th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2019, pp. 1–10.

[8] 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.*, to be published.

[9] J. Luo and K. G. Shin, "Detecting misplaced RFID tags on static shelved items," in *Proc. 17th Annu. Int. Conf. Mobile Syst., Appl., Services*, 2019, pp. 378–390.

[10] J. Liu, F. Zhu, Y. Wang, X. Wang, Q. Pan, and L. Chen, "RF-scanner: Shelf scanning with robot-assisted RFID systems," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[11] J. Liu, S. Chen, Q. Xiao, M. Chen, B. Xiao, and L. Chen, "Efficient information sampling in multi-category RFID systems," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 159–172, Feb. 2019.

[12] J. Han, C. Qian, X. Wang, D. Ma, J. Zhao, W. Xi, Z. Jiang, and Z. Wang, "Twins: Device-free object tracking using passive tags," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1605–1617, Jun. 2016.

[13] G. Wang, C. Qian, L. Shangguan, H. Ding, J. Han, N. Yang, W. Xi, and J. Zhao, "HMRL: Relative localization of RFID tags with static devices," in *Proc. 14th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2017, pp. 1–9.

[14] L. Yang, Y. Chen, X.-Y. Li, C. Xiao, M. Li, and Y. Liu, "Tagoram: Real-time tracking of mobile RFID tags to high precision using COTS devices," in *Proc. 20th Annu. Int. Conf. Mobile Comput. Netw.*, 2014, pp. 237–248.

[15] X. Liu, J. Yin, S. Zhang, B. Ding, S. Guo, and K. Wang, "Range-based localization for sparse 3-D sensor networks," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 753–764, Feb. 2019.

[16] Y. Bu, L. Xie, Y. Gong, C. Wang, L. Yang, J. Liu, and S. Lu, "RF-Dial: An RFID-based 2D human-computer interaction via tag array," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 837–845.

[17] H. Hu, Z. Liu, and J. An, "Mining mobile intelligence for wireless systems: A deep neural network approach," *IEEE Comput. Intell. Mag.*, vol. 15, no. 1, pp. 24–31, Feb. 2020.

[18] Y. Luo, H. Hu, Y. Wen, and D. Tao, "Transforming device fingerprinting for wireless security via online multitask metric learning," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 208–219, Jan. 2020.

[19] D. K. Klair, K.-W. Chin, and R. Raad, "A survey and tutorial of RFID anti-collision protocols," *IEEE Commun. Surveys Tutr.*, vol. 12, no. 3, pp. 400–421, 3rd Quart., 2010.

[20] J. Liu, X. Chen, X. Liu, X. Zhang, X. Wang, and L. Chen, "On improving write throughput in commodity RFID systems," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1522–1530.

[21] Q. Xiao, S. Chen, J. Liu, G. Cheng, and J. Luo, "A protocol for simultaneously estimating moments and popular groups in a multigroup RFID system," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 143–158, Feb. 2019.

[22] Q. Xiao, Y. Zhang, S. Chen, M. Chen, J. Liu, G. Cheng, and J. Luo, "Estimating cardinality of arbitrary expression of multiple tag sets in a distributed RFID system," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 748–762, Apr. 2019.

[23] J. Huang, Z. Wen, L. Kong, L. Ge, M.-Y. Wu, and G. Chen, "Accelerate the classification statistics in RFID systems," *Theor. Comput. Sci.*, vol. 788, pp. 39–52, Oct. 2019.

[24] 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.

[25] Z. Zhou, Y. Guo, Y. He, X. Zhao, and W. M. Bazzi, "Access control and resource allocation for M2M communications in industrial automation," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 3093–3103, May 2019.

[26] 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.

[27] J. Yu, J. Liu, L. Chen, and Y. Zhu, "Efficient group labeling for multi-group RFID systems," in *Proc. IEEE/ACM 25th Int. Symp. Quality Service (IWQoS)*, Jun. 2017, pp. 1–2.

[28] F. Zhu, B. Xiao, J. Liu, Y. Wang, and L.-J. Chen, "Dynamic grouping in RFID systems," in *Proc. 14th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2017, pp. 1–9.

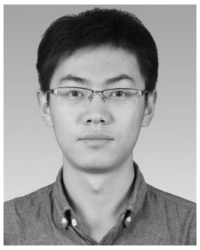
[29] J. Liu, B. Xiao, S. Chen, F. Zhu, and L. Chen, "Fast RFID grouping protocols," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 1948–1956.

[30] J. Liu, M. Chen, B. Xiao, F. Zhu, S. Chen, and L. Chen, "Efficient RFID grouping protocols," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 3177–3190, Oct. 2016.

[31] EPCglobal. (2018). *EPC Radio-Frequency Identity Protocols Generation-2 UHF RFID Standard, Specification for RFID Air Interface Protocol for Communication 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

LIYUE ZHU received the Ph.D. degree from the University of Science and Technology of China, in 2010. He is currently a Senior Engineer and the Deputy Director of the Academy of Broadcasting Science, NRTA, China. His research interests include coaxial transmission, network security, and collaborative optimization.

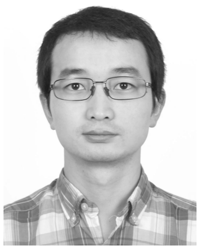
XIUJUN WANG received the Ph.D. degree in computer software and theory from the University of Science and Technology of China, in 2011. He is currently a Lecturer with the School of Computer Science and Technology, Anhui University of Technology. His research interests include data stream processing,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



YANGZHAO YANG received the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2014. He is currently a Senior Researcher with the Research Institute of Cyberspace Security of CETC. His current research interests include artificial intelligence and social networks.

SHUBIN XU received the Ph.D. degree from the University of Science and Technology of China, in 2009. He is currently a Research Professor with the Research Institute of Cyberspace Security of CETC, China. His research interests include network security, deep-learning-enabled network flow control, and the Internet of Things (IoT) security.

XUANGOU WU received the Ph.D. degree from the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China, in 2013. He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University of Technology, Ma'anshan, China. His research interests include computer networks, the Internet of Things, and privacy and security.



WEI ZHAO (Member, IEEE) received the Ph.D. degree in applied information science from Tohoku University, in 2015. He was an overseas Researcher under the Postdoctoral Fellowship of the Japan Society for the Promotion of Science" (JSPS) with Prof. T. Hara at Osaka University. He is currently an Associate Professor with the Anhui University of Technology, China. His major research interests include wireless mesh networks and mobile ad hoc networks. His articles received best paper awards at GLOBECOM 2014 and WCSP 2014.

HUIBIN FENG received the Ph.D. degree in information and communication engineering from the Nanjing University of Posts and Telecommunications. He is currently an Associate Professor with Minjiang University. His research interests include mobile edge computing and deep reinforcement learning for wireless networks. He is also a member of CCF.

• • •