Missing Tag Identification in Blocker-enabled RFID Systems

Xia Wang∗†, Jia Liu∗, Yanyan Wang∗, Feng Zhu∗, and Lijun Chen∗

∗State Key Laboratory for Novel Software Technology, Nanjing University, China
†College of Information Science and Engineer, Zaozhuang University, Shandong Province, China
Email: wangxia@smail.nju.edu.cn, jialiu@nju.edu.cn, {wangyanyan,zhfeng}@smail.nju.edu.cn, chenlj@nju.edu.cn

Abstract—In RFID systems, tags may blindly respond to any readers authorized or not, leading to privacy leakage. A feasible solution to this issue is to deploy blocker tags that emulate and behave like the genuine tags. With the blocker tags, there is always a virtual tag produced by the blocker tag that responds to the reader when the genuine tag to be protected replies, leading to irreconcilable collisions and thus preventing privacy-leakage. In this paper, we take the fresh attempt to study the problem of missing tag identification in blocker-enabled RFID systems. Unlike traditional missing tag identification, the reader in our problem will always get the responses from blocker tags even the genuine tags are lost, which makes the existing solutions unavailable. To address this problem, we propose a Group-based Missing Tag Identification (GMTI) protocol that first uses the grouping technology to divides the entire tag set into three subsets and later separately deals with each of them in different ways. The final result is that GMTI is able to identify any missing tags in a time-efficient, privacy-protected, and complete way.

I. INTRODUCTION

Radio Frequency IDentification (RFID) has been widely used in a variety of scenarios, such as supply chain management [1], tagged objects monitoring and tracking [2], [3], and warehouse inventory control [4], [5]. With a broader range of RFID-enabled applications, the privacy issues have attracted more and more attentions as a tag will blindly respond to any readers authorized or not, which causes privacy leakage in privacy-sensitive cases. For example, in a hospital, the patients may not would like to expose the tags associated with the medicine that reflects their illness to others. To protect the privacy, an effective solution is to block the tag’s response by using blocker tags [6], [7]. A blocker tag is actually a pre-configured super RFID tag or a specific RF device that can produce a set of virtual tags that behave like the genuine ones. More specifically, to protect a genuine tag, the blocker tag produces a virtual tag that has the same ID as the genuine one such that these two tags will reply concurrently, causing irreconcilable collisions and thus preventing the genuine tag being read. We refer to the virtual tags produced by the blocker tag as blocking tags.

In this paper, we investigate the new problem of missing tag identification in the blocker-enabled RFID system. Our objective is to identify any missing tags in a time-efficient way. This plays an important role in real-time warehouse management and monitoring as it helps detect the missing event and guard against theft. Although many advanced missing tag identification protocols [8]–[12] have been proposed, they cannot work properly in blocker-assisted RFID system as the virtual blocking tags will always reply to the reader no matter whether the genuine tags are lost or not. This is essentially due to the presence of blocker tags. Generally, a blocker-enabled RFID system consists of two tag sets: genuine tag set and blocking tag set. The former denoted by \( G \) includes all real tags whose IDs are known a priori by the legal reader, and the latter denoted by \( B \) is a virtual tag set that is produced by a blocker tag or multiple blocker tags. Compared with the previous work, this blocker-enabled RFID system introduces three following new features: (1) the blocking tags in \( B \) would never be lost; (2) the blocking tags in \( B \cap G \) will always reply to the reader even the genuine tags in \( B \cap G \) are lost; (3) all blocked tags (genuine tags protected by blocking tags) should not be exposed to others in the open wireless channel. Taking above features into account, the efficient missing tag identification protocol should follow the three design principles. First, the protocol does not need to check each blocking tag in \( B \) as they would never be missing. Second, to identify which genuine tags in \( B \cap G \) are lost, the protocol needs to distinguish more reply status rather than just check the presence or absence of replies. Third, the protocol should not transmit any blocked tags’ IDs.

In light of this, we in this paper propose the Group-based Missing Tag Identification (GMTI) to identify any missing tags in blocker-enabled RFID systems, satisfying the following three requirements: efficiency that means achieving the identification task in a time-efficient way; privacy that means never transmitting blocked tags’ IDs at any time, from the reader to tags or vice versa; completeness that means identifying every missing tag with certainty. To achieve this goal, GMTI first divides the entire blocker-enabled system into three subsets \( B - G, B \cap G, G - B \) by using the grouping technology, and later separately deals with each of them. For \( B - G \), GMTI silences all tags in this subset as they must not be lost. This will greatly improve the identification efficiency especially when \( B - G \) is large. For \( B \cap G \), GMTI utilizes the long (10-bit) tag replies to distinguish more reply status. By comparing the expected replies and the real ones, GMTI is able to infer the missing genuine tags in \( B \cap G \). For \( G - B \), GMTI uses the traditional protocols [12] to identify the missing tags as-is, since this subset has no blocking tags. The final result is that GMTI is able to achieve the missing tag identification in a time-efficient, privacy-protected, and complete way.
II. PROBLEM STATEMENT

A. System Model and Problem Description

In this paper, we consider a blocker-enabled RFID system that consists of a reader, a number of genuine tags under the reader’s coverage, and one or more blocker tags that can produce a set of virtual blocking tags. Seven tag sets in the system are formulated for ease of presentation of our protocols later. (1) \( G \): represents the on-site genuine tag set under the reader’s coverage. (2) \( B \): represents the blocking tag set produced by the blocker tags. (3) \((B \cap G)\): denotes the intersection of \( B \) and \( G \). Each tag in \((B \cap G)\) corresponds to an on-site genuine tag and a blocking tag that is used to protect this genuine tag. We refer to these two tags as a tag pair. (4) \((G - B)\): denotes the genuine tag subset that is not protected by \( B \). (5) \((B - G)\): denotes the blocking tag subset to which no genuine tags corresponds. (6) \((B+G)\): denotes all blocking tags and genuine tags in the reader’s interrogation zone. (7) \((B \cup G)\): denotes the union of tag sets \( B \) and \( G \). With the knowledge of these tag sets a priori, the problem of this paper is to identify all missing tags within \( G \) in a time-efficient, privacy-protected, and complete way.

B. Background

According to the C1G2 standard [13], the RFID communications are based on slotted ALOHA [14]. To achieve the communication, the reader first carries out a slotted frame by broadcasting a request \( \langle f, r \rangle \) to all tags in the interrogation zone, where \( f \) is the frame size and \( r \) is a random number. Upon receiving the request, each tag individually picks a slot in the frame by computing \( H({\text{ID}}, r) \mod f \), where \( \text{ID} \) is each tag’s ID, and then transmits the required information in the corresponding time slot. In other words, if two tags have the same ID, they will reply concurrently, making the reader fail to detect the missing event. To solve this problem, B-IIP uses the long tag response to distinguish three slot status: empty, singleton, and collision. When it finds a would-be 2-collision slot picking by a tag pair in \( B \cap G \) turns out to be a singleton slot, the reader is able to infer that the genuine tag is missing. For tags outside \( B \cap G \), B-IIP uses the previous ‘nonempty-to-empty’ scheme to infer missing tags. In this way, by comparing each expected slot and the real slot in the frame, B-IIP can successfully identify some missing tags. Generally, B-IIP contains multiple round identification, each of which consists of four steps below.

Building Real Frame. The real frame is built on all on-site tag’s responses. In the beginning of a round, the reader transmits an identification request with the parameters \( \langle f, r \rangle \) to issue a slotted frame, where \( r \) is the random seed and \( f \) is the frame size which is equal to \( |B \cup G| \). Upon receiving this request, each on-site tag will pick a slot whose index is \( H({\text{ID}}, r) \mod f \) to give a 10-bit long response to the reader, where \( \text{ID} \) is each tag’s ID. After that, the reader carries out this frame and listens to tag’s responses in each slot. According to the tags’ responses, the reader records each slot’s status as the real frame, denoted by \( F_R \).

Building Expected Frame. Unlike the construction of \( F_R \), the expected frame is built upon only the reader’s knowledge. Specifically, the reader assumes that no tag is missing. It takes the same parameters \( f, r \), and all tags in \((G + B)\) as the input to compute which tags pick which slot. According to all tags’ choices, the reader records each slot’s status, i.e., empty is ‘0’, singleton is ‘1’, and collision is ‘2’. We call the frame reflecting each real slot’s status as the real frame, denoted by \( F_E \).

Comparing Two Frames. In this step, B-IIP compares each slot between \( F_R \) and \( F_E \) and infers the missing tags. We use \( F_R[i] \) and \( F_E[i] \) (\( 1 \leq i \leq f \)) to denote the \( i \)th slot’s status in \( F_R \) and \( F_E \), respectively. We identify the missing tags according to the following rules.

(i) \( F_E[i] = 1 \) and \( F_R[i] = 0 \). In this case, all tags that would-be in the \( i \)th slot are lost.
(ii) \( F_E[i] = 2 \), \( F_R[i] = 1 \), and there is exactly one blocking protocol that is tailored to our problem. To identify missing tags, traditional protocols compare the expected slots and the real ones. If an expected nonempty slot turns out to be an empty slot in practice, the corresponding tags (would-be in this slot) are missing. In other words, these protocols infer the missing tags by checking the nonempty status or empty status of each slot. In our problem, however, the blocking tags in \( G \cap B \) will always reply to the reader even the genuine tags in \( G \cap B \) are lost, which makes the ‘nonempty-to-empty’ judgement unavailable.

In this section, we take the classic protocol IIP as the example to demonstrate how to make the existing work tailored to our problem. We refer to the improved protocol as B-IIP. Due to the presence of blocking tags, each tag pair in \( B \cap G \) has the same ID and will always reply to the reader concurrently when it is their turn to transmit. If the genuine tag in a tag pair is lost, the corresponding blocking tag will still respond as-is, making the reader fail to detect the missing event. To solve this problem, B-IIP uses the long tag response to distinguish three slot status: empty, singleton, and collision. When it finds a would-be 2-collision slot picking by a tag pair in \( B \cap G \) turns out to be a singleton slot, the reader is able to infer that the genuine tag is missing. For tags outside \( B \cap G \), B-IIP uses the previous ‘nonempty-to-empty’ scheme to infer missing tags. In this way, by comparing each expected slot and the real slot in the frame, B-IIP can successfully identify some missing tags. Generally, B-IIP contains multiple round identification, each of which consists of four steps below.

Building Real Frame. The real frame is built on all on-site tag’s responses. In the beginning of a round, the reader transmits an identification request with the parameters \( \langle f, r \rangle \) to issue a slotted frame, where \( r \) is the random seed and \( f \) is the frame size which is equal to \( |B \cup G| \). Upon receiving this request, each on-site tag will pick a slot whose index is \( H({\text{ID}}, r) \mod f \) to give a 10-bit long response to the reader, where \( \text{ID} \) is each tag’s ID. After that, the reader carries out this frame and listens to tag’s responses in each slot. According to the tags’ responses, the reader records each slot’s status as the real frame, denoted by \( F_R \).

Building Expected Frame. Unlike the construction of \( F_R \), the expected frame is built upon only the reader’s knowledge. Specifically, the reader assumes that no tag is missing. It takes the same parameters \( f, r \), and all tags in \((G + B)\) as the input to compute which tags pick which slot. According to all tags’ choices, the reader records each slot’s status, i.e., empty is ‘0’, singleton is ‘1’, and collision is ‘2’. We call the frame reflecting each real slot’s status as the real frame, denoted by \( F_E \).

Comparing Two Frames. In this step, B-IIP compares each slot between \( F_R \) and \( F_E \) and infers the missing tags. We use \( F_R[i] \) and \( F_E[i] \) (\( 1 \leq i \leq f \)) to denote the \( i \)th slot’s status in \( F_R \) and \( F_E \), respectively. We identify the missing tags according to the following rules.

(i) \( F_E[i] = 1 \) and \( F_R[i] = 0 \). In this case, all tags that would-be in the \( i \)th slot are lost.
(ii) \( F_E[i] = 2 \), \( F_R[i] = 1 \), and there is exactly one blocking
tag picking this slot. Since the blocking tag would never be lost, \( F_R[i] = 1 \) means only the blocking tag responds to the reader; the other tags that would-be in this slot are lost.

(iii) \( F_E[i] = 1, F_R[i] = 1 \). In this case, the reader knows that the corresponding tag is not missing.

**Silencing Tags**. In this step, the reader generates an \( f \)-bit vector to inform tags in each slot to keep silent. If a slot meets any above rule, the bit corresponding to this slot is set to ‘1’. Otherwise, it is set to ‘0’. The reader then broadcasts this vector to all tags. If a tag finds that its slot is ‘1’, it will keep silent in the following protocol execution. On the contrary, it will participate in the next round otherwise. By silencing tags, the cardinality of active tags will gradually decrease, improving the identification efficiency. After this step, this round terminates. This process repeats round by round until all tags are silent.

We take an example to illustrate the execution of B-IIP in a round. As shown in Fig. 1, the reader has the knowledge of the blocking tag set \( B = \{t_{1-5}\} \) and the genuine tag set \( G = \{t_{4-8}\} \). In Fig. 1(a), the reader first constructs the expected frame \( F_E \) based on \( B \) and \( G \). As we can see, \( F_E \) is ‘02120211’. However, since some genuine tags, i.e. \( t_4 \) and \( t_8 \), are missing, the real frame \( F_R \) differs from \( F_E \). As shown in Fig. 1(b), the reader obtains the real frame \( F_R \) which is ‘01120101’. By comparing \( F_E \) and \( F_R \), the reader is able to identify the present tags and missing tags. For example, since \( F_E[7] = 1 \) and \( F_R[7] = 0 \) which comply with rule (i), the reader knows that \( t_8 \) that is supposed to pick this slot is missing. With rule (ii), the reader knows \( t_6 \) and \( t_5 \) are missing by checking the 2nd slot and the 6th slot. With rule (iii), the reader knows that \( t_7 \) is not missing by comparing the 8th slot pair. After the inference, the reader broadcasts the vector ‘01100111’ to all tags to silence the corresponding tags. In this example, only \( t_4 \) is active and will participate in following rounds; others are silent.

Though B-IIP protocol can identify the missing tags in the blocker-enabled RFID systems, the time efficiency is low. There are two reasons. First, B-IIP needs to check each on-site tag, including \( B - G \) which would never be lost. Second, the reader in B-IIP needs all tags to transmit 10-bit long response for distinguishing three slot status, lowing the protocol performance. These two drawbacks give us huge room for improving the identification performance, which forms the efficient Group-based Missing Tag Identification (GMTI) protocol.

**IV. GROUP-BASED MISSING TAG IDENTIFICATION**

A. Basic Idea

As aforementioned, in an RFID system with the presence of blocker tag, the blocking tags \( B \) would never be lost, and the blocking tags in \( B \cap G \) will always respond to the reader even the genuine tags in \( B \cap G \) are lost. Based on these observations, GMTI improves the time-efficiency of B-IIP by following the two design principles below. (1) *Not checking tags in \( B - G \)*: since the blocking tags in \( B - G \) would never be lost and have no corresponding genuine tags, GMTI does not need to individually examine whether they are lost. (2) *Dealing with the tags in \( B \cap G \) and \( G - B \) separately*: GMTI utilizes 10-bit long reply to verify the presence of each tag for \( B \cap G \) and uses the 1-bit short responses to identify the missing tags for \( G - B \). To achieve above goals, GMTI divides the entire blocker-enabled system into three subsets: \( B \), \( B \cap G \), \( G - B \) by the grouping technique [15], and then deals with each subset in different ways. It generally consists of two phases: grouping phase and identification phase.

B. Grouping Phase

In this phase, the reader divides all tags into three groups \( P = \{P_i\}_{i=1}^3 \), where \( P_1 \) denotes \( B - G \), \( P_2 \) denotes \( G \cap B \), and \( P_3 \) denotes \( G - B \). We assign each tag a 2-bit group ID \( Gid \) to tell the tag which group it belongs to. ‘01’ represents \( P_1 \), ‘10’ represents \( P_2 \), and ‘11’ represents \( P_3 \). The grouping approach used here is Concurrent Grouping (CCG) protocol [15]. CCG contains multiple rounds and each round consists of two stages: ordering stage and labeling stage. We here give the skeleton of the two stages and the detailed protocol execution can be seen in [15].

In the ordering stage, the reader broadcasts a request with \( (f_1, r_1) \), where \( f_1 \) is the frame size, and \( r_1 \) is a random seed. Each ungrouped tag in \( B + G \) then maps its ID into selected time slot by hash function \( H(ID, r_1) \mod f_1 \). The singleton or collision slots that are selected by the tags from the same group are called homogeneous slots. The reader builds an ordering vector \( V \) whose size is \( f_1 \) according to the status of each time slot, i.e., homogeneous slots are denoted by ‘1’ and other slots are denoted by ‘0’. The reader broadcasts the ordering vector \( V \) for informing each tag which slot it picks. The tags that pick homogeneous slots are selected. In the labeling stage, the reader carries out an actual labeling frame of \( h \) slots (where \( h \) is the number of homogeneous slots) and transmits a 2-bit \( Gid \) in each slot to tell the selected tags in homogeneous slots their group IDs.

C. Identification Phase

In this phase, GMTI copes with each group \( P_1, P_2 \) and \( P_3 \) separately. For \( P_1 \), since the tags in \( P_1 \) would never be lost, and there are no genuine tags, GMTI silences all tags in \( P_1 \), which greatly improves the time-efficiency especially when \( B - G \) is large. For \( P_2 \), due to the presence of blocking tags, GMTI is built on top of slotted ALOHA [14] and utilizes the long tag reply to distinguish three types of slots: empty slots, singleton slots, and collision slots. GMTI infers the missing genuine tags in \( P_2 \) by comparing the expected slots and the
real ones. For $P_3$, since there are no blocking tags, GMTI just uses 1-bit tag response to identify the missing tags.

According to above three groups, the identification phase consists of three stages: silencing tags in $P_1$, identifying missing tags in $P_2$, and identifying missing tags in $P_3$.

1) Silencing Tags in $P_1$: The reader broadcasts a specific request to all tags in $P_1$ to let them keep silent, which saves the identification overhead in the following protocol execution.

2) Identifying Missing Tags in $P_2$: For $P_2$, GMTI uses 10-bit response to distinguish three slot statuses: empty, singleton, and collision. When it finds a would-be 2-collision slot picking by a tag pair in $P_2$ turns out to be a singleton slot, the reader is able to infer that the genuine tag is missing. Identifying missing tags in $P_2$ contains multiple rounds, each of which consists of four steps: building expected frame, building real frame, comparing two frames, and silencing tags.

Building Expected Frame. Based on the reader’s knowledge, the reader constructs a slotted frame with the length of $f_2$ by taking the random seed $r_2$ and $P_2$ as the input. The frame size $f_2$ is equal to the number of active tags in $P_2$ at the beginning of the current round. Similar to B-IIP, the reader records each slot’s status and builds the expected frame, denoted by $F_e$. In $F_e$, ‘0’ indicates empty, ‘1’ indicates singleton, and ‘2’ indicates j-collision ($j \geq 2$). Note that, there are no singleton slots in $F_e$ since each tag ID corresponds to a genuine tag and a blocking tag in $P_2$.

Building Real Frame. The real frame is built on on-site tags’ responses. In the beginning of a round, the reader transmits an identification request with the parameters $(f_2, r_2)$ to issue a slotted frame. Upon receiving this request, each tag in $P_2$ randomly picks a slot through the same hash computation. Then, the reader carries out an actual slotted frame and the on-site tags picking 2-collision slots give a 10-bit long response to the reader in the corresponding slots. According to the tags’ responses, the reader records each slot’s status in practice as the real frame, denoted by $F_r$. In $F_r$, ‘0’ indicates empty, ‘1’ indicates singleton, and ‘2’ indicates collision. Note that, unlike $F_e$, the reader can distinguish only the three types of slots according to tags’ responses.

Comparing Two Frames. In this step, GMTI compares each slot between $F_e$ and $F_r$, and infers the missing tags in $P_2$. We use $F_e[i]$ and $F_r[i]$ $(1 \leq i \leq f_2)$ to denote the ith slot’s status in $F_e$ and $F_r$, respectively. We identify the missing tags according to the following rules.

(iv) $F_e[i] = 2$ and $F_r[i] = 1$. In this case, the genuine tag would-be in the $i$th slot is lost.

(v) $F_e[i] = 2$ and $F_r[i] = 2$. In this case, the reader knows that the corresponding tag is not missing.

Silencing Tags. In this step, the reader generates an $f$-bit vector to inform tags in each slot to keep silent. If a slot meets any above two rules, the bit corresponding to this slot is set to ‘1’. Otherwise, it is set to ‘0’. The reader then broadcasts this vector to all tags. If a tag finds that its slot is ‘1’, it will keep silent in the following protocol execution. Otherwise, it will participate in the next round. By silencing tags, the cardinality of active tags will gradually decrease, improving the identification efficiency. After this step, this round terminates. The above four steps repeat round by round until all tags in $P_2$ are silent.

3) Identifying Missing Tags in $P_3$: Since $P_3$ only contains the genuine tags, GMTI utilizes traditional methods to infer the missing tags by checking the nonempty status or empty status of each slot. Missing tags identification in $P_3$ also contains multiple rounds, each of which consists of four steps.

Building Expected Frame. This step is to construct an expected frame $F_e$ which is same as that in previous phase.

Building Real Frame. This step is to build a real frame $F_r$ according to the tags’ responses. The frame size $f_3$ is equal to the number of active tags in $P_3$ at the beginning of the current round. When replying, the tags in $P_3$ send 1-bit short response rather than 10-bit long response to the reader. There are two types of slots, empty slots denoted by ‘0’ and non-empty slots denoted by ‘1’.

Comparing Two Frames. In this step, GMTI compares each slot between $F_e$ and $F_r$ to infer the missing tags in $P_3$. We identify the missing tags according to the following rules.

(vi) $F_e[i] \geq 1$ and $F_r[i] = 0$. In this case, the genuine tags would-be in the $i$th slot are lost.

(vii) $F_e[i] = 1$ and $F_r[i] = 1$. In this case, the genuine tag in the $i$th slot is not missing.

Silencing Tags. Similar to dealing with $P_2$, this step silences the tags picking the slots which meet the above rule by constructing and broadcasting a vector.

We take an example to illustrate the one-round execution of GMTI. As shown in Fig. 2, the reader has the knowledge of the blocking tag set $B = \{t_1-6\}$ and the genuine tag set $G = \{t_{3-10}\}$. GMTI groups all tags into three subsets: $P_1 = \{t_1, t_2\}$, $P_2 = \{t_{3-6}\}$, and $P_3 = \{t_{7-10}\}$. In Fig. 2(a), for $P_1$, the reader lets $\{t_1, t_2\}$ keep silent. For $P_2$, the reader constructs the expected frame $F_e$ ‘0242’. The reader then broadcasts $(f_2, r_2)$ to the tags in $P_2$. These tags respond 10-bit information to the reader. For $P_3$, the reader constructs the expected frame $F_e$ ‘1010’. The reader then broadcasts $(f_3, r_3)$ to the tags in $P_3$. These tags respond 1-bit information to the reader. Since some genuine tags, i.e., $t_3$ and $t_{10}$, are missing, the real frame $F_r$ differs from $F_e$. As shown in the Fig. 2(b), for $P_2$, the reader obtains the real frame $F_r$ which is ‘0122’. By comparing $F_e$ and $F_r$, the reader is able to identify the present tags and missing tags. For example, since $F_e[2] = 2$ and $F_r[2] = 1$ which comply with rule (iv), the reader knows that tag $t_3$ that is supposed to pick this slot is missing. With rule (v), the reader knows that $t_{10}$ is not missing by comparing the 4th slot pair. For $P_3$, the reader obtains the real frame $F_r$ which is ‘0010’. By comparing $F_e$ and $F_r$, the reader is able...
to identify the present tags and missing tags. For example, since \( F_e[1] = 1 \) and \( F_r[1] = 0 \) which comply with rule (vi), the reader knows that \( t_{10} \) is missing. With rule (vii), the reader knows that \( t_8 \) is not missing by comparing the 3rd slot pair.

D. Performance Analysis

We now discuss the execution time of GMTI. As aforementioned, GMTI consists of two phases: grouping phase and identification phase. Due to the space limitation, we omit the analysis of the execution time \( T_0 \) of the grouping phase and the details can be seen in [15]. On the contrary, we focus on the execution time of identification phase, which actually contains three stages: time \( T_{P_3} \) of coping with \( P_1 \), time \( T_{P_2} \) of coping with \( P_2 \), and time \( T_{P_1} \) of coping with \( P_3 \).

1) Execution time \( T_{P_1} \): To silence all tags in \( P_1 \), the reader just needs to broadcast a specific request by taking the group ID of \( P_1 \) as the destination address. The length of the request can be set as required. For example, if we set it to 8 bits and take the 2-bit group ID into account, the reader in this stage can be set as required. For example, if we set it to 8 bits and take the 2-bit group ID into account, the reader in this stage just needs to send 10-bit data in total.

2) Execution time \( T_{P_2} \): The stage of dealing with \( P_2 \) consists of four steps. The main execution time comes from the communication delay caused by building real frame and silencing tags. Consider the \( i \)th round. The execution time of building the real frame is \( f_{2,i} \times t_l \), where \( f_{2,i} \) is the frame size; the execution time of silencing tags is \( f_{2,i} \times \frac{t_{tag}}{96} \). Thus, we have the execution time \( T_{P_2} \) in this round:

\[
T_{P_2} = \frac{f_{2,i}}{96} \times t_{tag} + f_{2,i} \times t_l \tag{1}
\]

The execution time \( T_{P_2} \) is the sum of \( k \) rounds, i.e.,

\[
T_{P_2} = \sum_{i=1}^{k} T_{P_2} = \frac{1}{96} \sum_{i=1}^{k} f_{2,i} \times t_{tag} + \sum_{i=1}^{k} f_{2,i} \times t_l \tag{2}
\]

Now we derive the value of \( f_{2,i} \). Since \( f_{2,i} \) is equal to the number \( N_{2,i} \) of genuine tags in \( P_2 \) at the beginning of the \( i \)th round, the item \( \sum_{i=1}^{k} f_{2,i} \) equals \( \sum_{i=1}^{k} N_{2,i} \). Clearly, in the first round, \( N_{2,1} = N_2 \), where \( N_2 \) is the number of genuine tags that would be in \( P_2 \). In the \( i \)th round, the tags in the slots satisfying rules (iv) and (v) will keep silent. In other words, if a slot picked by exactly one blocking tag, the corresponding genuine tag will keep silent (if not missing) or not be taken into account (if missing). Therefore, we have the probability \( p_i \) that a genuine tag keeps silent after the \( i \)th round:

\[
p_i = \left( N_{2,i} \right) \times \frac{1}{f_{2,i}} \times (1 - \frac{1}{f_{2,i}})^{N_{2,i} - 1} \approx e^{-1} \tag{3}
\]

where \( e \) is the natural constant. Thus, we have:

\[
\sum_{i=1}^{k} N_{2,i} = N_2 \times \sum_{i=1}^{k} (1 - e^{-1})^{i-1} \approx 2.72 N_2 \tag{4}
\]

where the expected value of \( k \) is \( \left\lfloor \frac{1}{e^{-1}} \right\rfloor = 3 \). Substituting (4) for \( \sum_{i=1}^{k} f_{2,i} \) in (2), we have:

\[
T_{P_2} = 2.72 N_2 \times \left( \frac{1}{96} t_{tag} + t_l \right) \tag{5}
\]

3) Execution time \( T_{P_3} \): The analysis of \( T_{P_3} \) is similar to \( T_{P_2} \), except for two minor differences. First, when dealing with \( P_3 \), the tag just gives a short response \( t_s \) instead of \( t_l \) in \( P_2 \). Second, the number of genuine tags in \( P_3 \) may not differ from that in \( P_2 \). Thus, we have:

\[
T_{P_3} = 2.72 N_3 \times \left( \frac{1}{96} t_{tag} + t_s \right) \tag{6}
\]

where \( N_3 \) is the number of genuine tags in \( P_3 \). By summing up \( T_0 \), \( T_{P_1} \), \( T_{P_2} \), and \( T_{P_3} \) together, we can derive the total execution time of GMTI.

V. EVALUATION

A. Simulation Setting

Our simulation settings follow Philips I-Code specification [16]. Any two consecutive communications, from the reader to tags or vice versa, are separated by 302\( \mu \)s interval. The tag-to-reader transmission rate and the reader-to-tag data rate are not symmetric, which depends on the physical implementation and the practical environment. The transmission rate from the reader to the tag is set to 26.7\( \text{kb/s} \). We let the request for silencing \( P_1 \) be 10 bits long and we have \( T_{P_1} = 0.7\text{ms} \). The transmission rate from the tag to the reader is set to 47.5\( \text{kb/s} \). Thus, we have \( t_{tag} = 3.8\text{ms} \), \( t_{Gid} = 0.08\text{ms} \), \( t_l = 0.8\text{ms} \), and \( t_s = 0.3\text{ms} \). All simulation results are the average of 100 simulation runs in Matlab.

B. Execution Time

We compare the execution time of B-IIP and GMTI under three different scenarios. In the first scenario, we set \( N_1 = 10000 \), \( N_2 = 500 \), and \( N_3 = 500 \), where \( N_1 \) is the number of tags in \( P_1 \) \((B \cap G)\), \( N_2 \) is the number of genuine (blocking) tags in \( P_2 \) \((B \cap G)\), and \( N_3 \) is the number of tags in \( P_3 \) \((G \cap B)\). In the second scenario, we set \( N_1 = 8000 \), \( N_2 = 500 \), and \( N_3 = 500 \). In the third scenario, we set \( N_1 = 10000 \), \( N_2 = 500 \), \( N_3 = 1000 \). Fig. 3(a) shows the simulation results. We observe that GMTI takes much less execution time than B-IIP under these three scenarios. Take the 2nd scenario as example. The execution time of B-IIP is 12.05s, GMTI drops it to 2.3s, producing a 5.2\times performance gain, which well indicates the high efficiency of GMTI.

C. Impact of Parameter Settings

Now we study in greater details the impact of different parameter settings, including \( N_1, N_2, \) and \( N_3 \). The number of missing tags is set to 20. Fig. 3(b) shows how \( N_1 \) influences the execution time of GMTI and B-IIP. In this simulation, we fix \( N_2 = 500 \), \( N_3 = 1000 \), and vary \( N_1 \) from 5000 to 10000. As shown in this figure, the execution time of GMTI almost remains stable since it does not need to check the tags in \( P_1 \) (which keep mute via previous broadcast by the reader). In contrast, the execution time of B-IIP increases as the number \( N_1 \) increases. This is intuitive as B-IIP needs to check more tags when \( N_1 \) is bigger. Next, we study the impact of the \( N_2 \) and \( N_3 \) on the execution time, respectively. In Fig. 3(c), we fix \( N_1 = 10000 \), \( N_3 = 1000 \), and vary \( N_2 \) from 1 to 10000. In Fig. 3(d), we set \( N_1 = 10000 \), \( N_2 = 500 \), and vary \( N_3 \) from
1 to 1000. From these two figures, we can see that execution time of the two protocols increases with $N_2$ and $N_3$. This is in consistent with what we expected since the bigger $N_2$ and $N_3$ are, the more tags these two protocols need to check. Even so, we assert that although both B-IIP and GMTI experience the rise trend, GMTI is superior to B-IIP under all various parameter settings.

VI. RELATED WORK

Missing tag identification is one of the most important research branches in RFID as it is able to detect the missing event and guard against theft [8]–[12]. Existing missing tag identification protocols fall into two categories: the probabilistic [8], [11], [17] and the deterministic [9], [10], [12]. The former reports only the missing-tag event, without pinpointing which ones are missing. TRP protocol was early work that was proposed to detect the missing-tag event when the number of missing tags exceeds a tolerance threshold [8]. To improve the time-efficiency and energy efficiency, EMD protocol introduced birthday paradox to detect the missing tag event [11]. To further improve the performance, a multi-hashing approach was proposed to increase the utilization of the time frame [17]. The deterministic protocols can identify missing tags exactly. IIP protocol [12] utilized the singleton slots to verify the presence of the tags for identifying the missing tags completely. SFMTI protocol used reconciling and filtering methods to increase the efficiency of IIP [10]. P-MTI investigated physical-layer missing tag identification via compressive sensing approach [9]. Although these works make a great performance boost to the missing tag detection, they cannot work properly in Blocker-enabled RFID systems.

VII. CONCLUSION

This paper investigates a new problem of missing tags identification in the blocker-enabled RFID systems. Unlike traditional missing tag identification, the reader in this system will always get the responses from blocker tags even the genuine tags are lost, which makes the existing solutions unavailable. In light of this, we propose Group-based Missing Tag Identification (GMTI) protocol to identify every missing tag in blocker-assisted RFID systems. By using grouping technique, GMTI divides the entire RFID system into three subsets, and later deals with each of them in different ways. Simulation results demonstrate that GMTI greatly improve the identification efficiency, compared with the baseline protocol.

ACKNOWLEDGMENT

This research is financially supported by the National Natural Science Foundation of China (No.61272418), the National Science and Technology Support Program of China (No.2012BAK26B02), the Future Network Prospective Research Program of Jiangsu Province (No.BY2013095-5-02), the Lianyungang City Science and Technology Project (No.CG1420, JC1508), the Fundamental Research Funds for the Central Universities. This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

REFERENCES


