DBF: A General Framework for Anomaly Detection in RFID Systems

Min Chen[†‡] Jia Liu[⊥] Shigang Chen[†] Yan Qiao[†‡] Yuanqing Zheng[§]
[†]Department of Computer & Information Science & Engineering, University of Florida, Gainesville, FL, USA
[‡] Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA, USA
[⊥]State Key Laboratory for Novel Software Technology, Nanjing University, China
[‡]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
Email:[†‡]{minchen, yqiao}@google.com [†]sgchen@cise.ufl.edu [⊥]jialiu.cs@gmail.com [§]csyqzheng@comp.polyu.edu.hk

***Abstract*—RFID technologies are making their way into numerous applications, including inventory management, supply chain, product tracking, transportation, logistics, etc. One important application is to automatically detect anomalies in RFID systems, such as missing tags, unknown tags, or cloned tags due to theft, management error, or targeted attacks. Existing solutions are all designed to detect a certain type of RFID anomalies, but lack a general functionality for detecting different types of anomalies. This paper attempts to propose a general framework for anomaly detection in RFID systems, thereby reducing the complexity for readers and tags to implement different anomaly-detection protocols. We introduce a new concept of differential Bloom filter (DBF), which turns physical-layer signal data into a segmented Bloom filter that encodes the IDs of abnormal tags. As a case study, we propose a protocol that builds DBF for identifying all missing tags in an efficient way. We implement a prototype for missing-tag identification using USRP and WISP tags to verify the effectiveness our protocol, and use large-scale simulations for performance evaluation. The results show that our solution can significantly improve time efficiency, when comparing with the best existing work.**

I. Introduction

Radio-frequency identification (RFID) technologies integrate simple communication, storage, and computation components into attachable tags to enable wireless communications over a distance [1]. An RFID system generally consists of three components: One or more readers, a large number of tags, and a backend server. The tags can be attached to different objects, varying from products in a warehouse, merchandizes in a retail store, animals in a zoo, or medical equipments in a hospital. A reader can read the unique IDs of the tags or collect aggregate information about the tagged objects by communicating with the tags via RF signals. The research community is working actively to expand the application scope of RFID technologies [2]–[6]. Practical RFID systems [7] have been widely applied to inventory and logistics management, object tracking, access control, automatic toll payment, theft prevention, localization, intelligent transportation systems, etc.

A large RFID system can contain thousands of tags. Therefore, RFID anomalies, e.g., missing tags, may occur from time to time, but are hard to be detected manually. It is of importance to have some tools that can automate the process of anomaly detection in RFID systems. Common anomalies in an RFID system, which may be caused by theft, management error or some targeted attacks, include missing tags, unknown tags, cloned tags, etc. A *missing tag* is one that should exist in the system but turns out to be not represent, an *unknown tag* is one that is not recorded by the system's inventory list, and a *cloned tag* is an illegal replica of an authentic tag in the system. A missing, unknown, or cloned tag is also called an *abnormal tag*. Timely anomaly detection is very important to RFID systems. Consider a large storage for retired military equipment or other long-term storages of sealed objects in civilian applications. When no one keeps a close eye on them for a long time, how do we know whether anything is missing? One way is to have someone periodically walk through the place, up and down a latter over shelve after shelve to count items. This will be laborious and error-prone, considering that things may be stacked together and objects on the back of shelves may be blocked from view. But if we attach an RFID tag to each item, the process of finding the absence of tags (and their associated objects) may be fully automated through the communications between tags and readers.

Because of its practical importance, tremendous efforts from the research community have been devoted to developing anomaly-detection solutions for RFID systems. Time efficiency is a key concern since RFID systems operate with low-speed communication channels. More importantly, anomaly events should be timely detected and properly handled, thereby minimizing their potential negative impact. For example, if the missing tags can be identified in time, actions such as blocking the exit may be taken to avoid the loss. The prior research on RFID anomaly detection can generally be categorized as follows:

Missing-tag detection: It further includes two subcategories of research problems: (1) missing-tag event detection [8], [9], which is to find out *whether any tag is missing*, and (2) missing-tag identification [10]–[14], which deals with a harder problem of identifying *which tag(s) is missing*.

Unknown-tag identification: This is to collect the IDs of all unknown tags which have not been recorded by the inventory list [15], [16].

Cloned-tag identification: This is to identify all the IDs of tags cloned by adversaries [17], [18]. As a cloned tag copies all data from an authentic tag (which may be compromised), it can pass any authentication. Hence, cloned-tag identification needs to verify whether an ID is carried by multiple tags.

One naive approach for anomaly detection is to collect the IDs of all tags currently in the system and compare them with an inventory list to see which ones are missing, unknown, or cloned. However, when the number of tags is very large, this approach is not efficient due to transmission collisions caused by

978-1-5090-5336-0/17/$31.00 ©2017 IEEE

channel contention. Prior work on anomaly detection generally relies on an important observation that an abnormal tag may cause state change of MAC-layer slots. A snapshot of an RFID system is taken using a slotted time frame, where each tag is mapped (by hashing its ID) to one of the slots and the tag is supposed to transmit in that slot. The idea is that if any abnormal tag exists, it may affect the slot it is mapped to. For example, if the only tag mapped to a certain slot is missing, that should-be busy slot will become an idle one as no tag will transmit during that slot.

However, although tone of solutions have been proposed for detecting a certain type of RFID anomalies, there is no prior work that provides a general functionality for detecting different types of anomalies. Since every existing solution is tailored for a specific application of anomaly detection, it is not trivial to adapt one to other applications. For example, in missing-tag identification, each tag is required to transmit one-bit information in the slot it is mapped to confirm its presence [12]–[14]. In contrast, for cloned-tag identification, each tag needs to transmit multiple bits in its slot such that the reader can recognize collision slots [17]–[19]. It will incur much complexity to implement all anomaly detection protocols, especially for low-cost RFID tags with limited resources.

Is it possible to design a general framework such that we can smoothly switch among different categories of anomaly detection? We observe that any abnormal tag, regardless its type, can have some impact on the aggregate physical-layer signals of the tags in the system. For example, the signals contributed by a missing tag will disappear from the aggregate signals. Therefore, instead of utilizing state changes of MAC-layer slots, which need to be detected in significantly different ways, we leverage the changes of aggregate physical-layer signals for anomaly detection.

To our best knowledge, this is the first work that proposes a general framework for anomaly detection in RFID systems. In this paper, we introduce a new concept called *differential Bloom filter* (DBF). It has the structure of a segmented Bloom filter and identifies the abnormal tags by taking physical-layer signal snapshots to derive differential signals such that the aggregate information of all normal tags is subtracted away and only the information about the small number of abnormal tags is digitized and encoded in the Bloom filter for the purpose of identification. As a case study, we use DBF for missing-tag identification. We implement a prototype of DBF using USRP and WISP tags to verify its effectiveness through small-scale experiments, while using simulations for large-scale evaluation. The results show that our solution can significantly improve time efficiency in most cases, when comparing with the best existing work.

II. System Model and Problem Statement

A. System Model and Assumptions

Consider a large long-term storage of objects, each of which is attached with a tag, carrying a unique identifier. There is one or multiple readers. The readers communicate with the tags using RF signals. The backend server has a database storing information about the system, and it is capable of carrying out high-performance computations on behalf of the readers. The readers are connected to the backend server via a high-speed wired or wireless link, so the communication latency between them is negligible.

There are different types of RFID tags on the market: *Active tags* have their own batteries, while *passive tags* harvest radio energy emitted from the readers for backscatter communication. Passive tags are more widely used nowadays because of their simplicity and low prices. In this paper, we focus on passive tags that operate at Ultra-High Frequency 860-960MHz [1]. As backscatter communication is generally within a narrow wireless band, the wireless channel between a tag t and a reader can be mathematically modeled with a complex number h_t, incorporating both signal attenuation and phase shift of the wireless channel [20].

The communications between a reader and tags operate in a request-and-response mode. The reader initiates the interrogation by broadcasting a command along with some parameters [1]. The tags will respond in the subsequent time frame. Following the prior work [13], [14], [21], the uplink communication from tags to a reader is assumed to be synchronized at symbol level by the reader, which has been proved to be achievable by experiments in [20].

We will first consider a system of a single reader, and later discuss the case of multiple readers. We assume that the reader has access to a database that stores the IDs of all tags. This assumption is necessary [8], [14]; we cannot determine if a tag is an abnormal one if we do not even know its existence. The assumption can be easily satisfied by a typical inventory management procedure: the tag IDs are read into a database when new objects are moved into the system, and they are removed from the database when the objects are moved out.

The scope of applicability for the proposed work, which is the same as those of [8]–[10], [12]–[19], is given as follows: We assume a long-term storage environment with stable conditions, where objects are statically placed. The function of anomaly detection does not work during occasions when people move objects in/out or rearrange them inside the storage facility. Namely, the function is designed to work at time when such activities are not present.

B. Problem Statement

Let N_i be the set of tags in the system at time T_i, $i \geq 0$. Suppose N_{i-1} contains no abnormal tags, and some anomaly event happens between time T_{i-1} and time T_i. We focus three common types of anomaly detection, which are listed as follows:

1) Missing-tag identification: Suppose a set M_i of tags are missing between T_{i-1} and T_i. We have $M_i \subseteq N_{i-1}$, $N_i \subseteq N_{i-1}$, $N_{i-1} = M_i \cup N_i$ and $M_i \cap N_i = \emptyset$. The problem of *missing-tag identification* is to identify all missing tags in M_i.
2) Unknown-tag identification: Suppose a set M_i of unknown tags are moved to the system between T_{i-1} and T_i. We have $M_i \subseteq N_i$, $N_{i-1} \subseteq N_i$, $N_i = M_i \cup N_{i-1}$ and $M_i \cap N_{i-1} = \emptyset$. The problem of *unknown-tag identification* is to identify all unknown tags in M_i.
3) Cloned-tag identification: Suppose a set M_i of tags are cloned by an adversary between T_{i-1} and T_i. Note that in this case, N_i and M_i can be multisets. We have $M_i \subseteq N_i$, $N_{i-1} \subseteq N_i$, $N_i = M_i \cup N_{i-1}$, and the distinct tags in

N_{i-1} and N_i are the same. The problem of *cloned-tag identification* is to identify all cloned tags in M_i.

We want to design a general framework that can be applied to any of the three types of anomaly detection. Our framework is expected to: (1) perform anomaly detection in a time-efficient way such that it is capable of reporting the real-time state of the system and scaling to large systems with thousands of tags; (2) generate no *false positive* or *false negative* in the identification result, where a false positive is defined as a tag in N_i being mistakenly included in M_i, and a false negative is defined as a tag in M_i not being identified. It would be interesting to perform anomaly detection when different types of abnormal tags coexist. However, due to space limitation, we will leave this more challenging problem as our future work.

III. Framework Design

We use physical-layer snapshots to derive a new construct called *differential symbol filter*, which is then digitized into a differential Bloom filter (DBF) that encodes the membership of the abnormal tags. When the context is clear, we also use DBF for referring to the protocol of building the filter (this section) and using it to identify the abnormal tags (the next section).

A. Motivation

The prior work generally encodes each tag to a separate slot, thereby identifying the abnormal ones based on the observed slot states. Hence, it takes at least $O(n)$ time slots, where n is the number of the tags in the system. In a large system, the tags of interest, i.e., the abnormal ones, can be small [10]–[14]. Can we identify all of them with $O(m)$ time slots instead, where m is the number of abnormal tags? Moreover, in order identify different types of abnormal tags, different types of slots are needed [8]–[19], rendering those protocols not universally applicable. This motivates us to design a new differential Bloom-filter approach below.

The idea is that we actually do not need a snapshot of $O(n)$ time slots with all tags being recorded separately. We only need a "snapshot" of the m abnormal tags, while the information of the normal tags is of no interest. This can be achieved through physical-layer signals: Suppose we encode all n tags in each snapshot of $O(m)$ time slots. It is likely that there is no singleton since there are too few slots. Instead of ternary collision/singleton/empty information, each snapshot now records the physical signals carried in the slots from all tags. By combining two consecutive snapshots, we can subtract away the unchanged information from the normal tags and produce differential symbols (signals), which were transmitted by the abnormal tags, either in the first snapshot or the second snapshot. But the real challenge is how to design the differential symbols and how to use them for anomaly detection.

In this paper, we present a design of differential symbols that can be digitized into binary states, which together form a Bloom filter [22], encoding the set of m abnormal tags only. The new approach only requires each tag transmits a few '1's, while staying silent for most of the time. As a result, only a small number of tags will transmit in each slot.

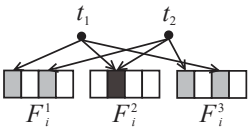

Fig. 1: In this example, the physical-layer snapshot consists of three segments, and each segment contains four symbols. The representative symbols of tag t_1 are $F_i^1[1]$, $F_i^2[2]$, and $F_i^3[3]$, which are shown in grey. The symbol $F_i^2[2]$ is a representative for both t_1 and t_2.

B. Physical-layer Snapshot

At time T_i, the reader constructs a physical-layer snapshot F_i based on the tags' responses. The snapshot consists of k segments (one segment during each time frame), denoted as F_i^j, $1 \le j \le k$, each of which consists of l symbols (received by the reader from the tags), denoted as $F_i^j[s]$, $1 \le s \le l$.

By hashing its ID and j, each tag t is mapped to one symbol in each segment F_i^j, which is called a *representative* symbol of t. (Such a mapping function is also required by [8]–[19].) Each tag has exactly one representative symbol in every segment and k representatives in total, while each symbol $F_i^j[s]$ may be a representative for multiple tags, denoted as a set $R_i^j[s]$, that happen to be mapped to the same symbol. Note that when cloned tags exist, $R_i^j[s]$ can be a multiset. An illustrative example is shown in Fig. 1.

For each segment, F_i^j, $1 \le j \le k$, the reader broadcasts a command to initiate the construction. The duration for every symbol, referred to as a time slot, is fixed. Each tag t will wait until the time of its representative symbol and transmit a signal x_t. What the reader actually receives is

$$y_t = h_t x_t + e_t, \tag{1}$$

where h_t was introduced earlier in the system model, and e_t is a term of channel error. Note that the tags can be resynchronized by the reader's command before the construction of each segment, which significantly reduces the negative effect caused by clock drift of the tags. If the segment is too long, we may divide it into blocks and require the reader to synchronize the tags at the beginning of each block.

Due to random mapping of tags to symbols in the segment, there can be other tags that share the same representative symbol as t and thus transmit simultaneously. Hence, what the reader receives may be the combination of transmissions from multiple tags. More specifically, considering an arbitrary symbol $F_i^j[s]$ that the reader receives, we have

$$F_i^j[s] = \sum_{t \in R_i^j[s]} y_t. \tag{2}$$

Each tag transmits only once in each segment and stays silent for the rest of the time. Hence, each time slot records the transmissions from $\frac{n}{l}$ tags on average. By choosing a sufficient value for l, we can reduce $\frac{n}{l}$ to an arbitrarily small value.

C. Differential Symbol Filter

Suppose the reader constructs a new physical-layer snapshot after a certain time interval and uses it together with the previous snapshot for anomaly detection. To ensure timely detection,

one may want to set the time interval, $T_i - T_{i-1}$, between the constructions of the two consecutive snapshots small.

Over time, the reader constructs a series of snapshots, F_i, $i \geq 0$. For each new snapshot F_i constructed where $i > 0$, the reader derives a *differential symbol filter* D_i based on the new snapshot and its predecessor F_{i-1} as follows: Let D_i^j be the jth segment of the differential filter and $D_i^j[s]$ be the sth differential symbol in the segment, $1 \leq j \leq k$ and $1 \leq s \leq l$.

$$\begin{aligned} D_i^j[s] = F_{i-1}^j[s] - F_i^j[s] &= \sum_{t \in R_{i-1}^j[s]} y_t - \sum_{t \in R_i^j[s]} y_t \\ &= \sum_{t \in R_{i-1}^j[s] \Delta R_i^j[s]} y_t, \end{aligned} \tag{3}$$

where 'Δ' is the symmetric difference operator of sets, and $R_{i-1}^j[s] \Delta R_i^j[s] = (R_{i-1}^j[s] - R_i^j[s]) \cup (R_i^j[s] - R_{i-1}^j[s])$.

D. Convert Differential Symbol Filter to Bloom Filter

We convert each differential symbol filter D_i to a binary filter B_i, consisting of k segments, denoted as B_i^j, $1 \leq j \leq k$, each of which consists of l bits, denoted as $B_i^j[s]$, $1 \leq s \leq l$. The conversion is performed as follows: $\forall j \in [1,k], s \in [1,l]$,

$$B_i^j[s] = \begin{cases} 0 & \text{if } \|D_i^j[s]\| \leq \theta \\ 1 & \text{if } \|D_i^j[s]\| > \theta, \end{cases} \tag{4}$$

where the operator $\|.\|$ calculates the magnitude of $D_i^j[s]$, and θ is a threshold value that should be significantly larger than the magnitude of channel error. In practice, the reader may keep monitoring the channel, and set θ accordingly. Obviously, the signal strength transmitted by the tags should be much larger than θ. Suppose the aggregate channel error is smaller than θ. It is straightforward to see the following proposition.

Proposition 1. $\forall i > 0, j \in [1,k], s \in [1,l],\ B_i^j[s] = 0$ *if and only if* $R_i^j[s] = R_{i-1}^j[s]$.

Each bit in the binary filter B_i represents the differential state of a slot: Zero means that there is no state change and one means that there is a state change, indicating one or multiple abnormal tags. Hence, B_i can serve as a tool for anomaly detection. Furthermore, we show below that B_i can be used to identify the IDs of the abnormal tags as well because it is actually a Bloom filter that encodes those abnormal tags. (It is interesting to point out that even if the channel error is sometimes greater than θ and causes some bits to ones, it only increases false positives, which already exist in the Bloom filter and will be handled by our protocol shortly.)

Recall that N_i is the set of tags at the time when F_i is constructed, N_{i-1} the set at the time when F_{i-1} is constructed, and the set of abnormal tags is M_i. The following theorem shows that B_i is a segmented Bloom filter for M_i. Each member t in M_i is pseudo-randomly mapped to k bits, each from one segment, in the same way as t was mapped to F_i — these bits are called the *representative bits* of tag t. Tag t is encoded in B_i if all its representative bits are ones.

Theorem 1. *B_i is a segmented Bloom filter for M_i. That is, a bit in B_i is one if and only if it is a representative bit of a tag in M_i.*

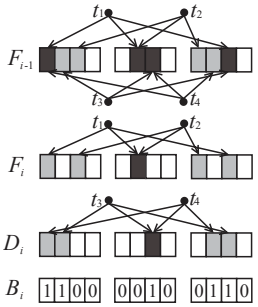

Fig. 2: An example of creating a Bloom filter of missing tags: F_{i-1} is the $(i-1)$th physical-layer snapshot with four tags, where a white rectangle represents no signal from tags, a light-grey rectangle represents a symbol from a single tag, and a dark-grey rectangle represents a combined symbol from signals of multiple tags. F_i is the ith physical-layer snapshot with two tags. D_i is the differential symbol filter generated by subtracting F_i from F_{i-1}, symbol by symbol. Its non-zero symbols (light or dark grey rectangles) are exactly those that are representatives of the missing tags. B_i is the Bloom filter derived from D_i.

Proof: Consider an arbitrary bit $B_i^j[s]$, $\forall j \in [1,k], s \in [1,l]$. We have two cases.

- *Case 1: $B_i^j[s]$ is a representative bit for some tag t in M_i.* We need to show that $B_i^j[s] = 1$. Because tag t is mapped to $B_i^j[s]$, it must also be mapped to the corresponding symbols, $F_i^j[s]$ and $F_{i-1}^j[s]$. Since t is an abnormal tag in M_i, it only belongs to either $R_{i-1}^j[s]$ or $R_i^j[s]$ by definition. Therefore, $R_i^j[s] \neq R_{i-1}^j[s]$, which means $B_i^j[s] = 1$ by Proposition 1.
- *Case 2: $B_i^j[s]$ is not a representative bit for any tag in M_i.* We need to show that $B_i^j[s] = 0$. Because no abnormal tag is mapped to the corresponding symbols $F_i^j[s]$ and $F_{i-1}^j[s]$, we must have $R_i^j[s] = R_{i-1}^j[s]$, which means $B_i^j[s] = 0$ by Proposition 1.

■

Fig. 2 gives an illustrative example of missing-tag event, where four tags in N_{i-1} are mapped to F_{i-1}, two tags in N_i are mapped to F_i (with two tags missing), only the signals from the missing tags will be recorded by the differential symbol filter D_i, and B_i is in fact a segmented Bloom filter for the two missing tags.

An interesting observation is that even when the number of tags is very large, the size of the filters can be small as long as there are not too many abnormal tags. In the above example, the same filters can be used when there are many more tags. Each slot will carry the signals from more tags, but as the signals from non-missing tags are subtracted away, in the end we will get the same B_i as long as only the two tags are missing.

It is well known that the size of an optimal Bloom filter is $-\frac{\ln p_f}{(\ln 2)^2}|M_i|$, where p_f is the false-positive ratio. Although we do not know $|M_i|$, we may set the filter size (thus the physical-layer snapshot size) based on an estimated upper bound of abnormal tags, which we will be further discussed later.

E. Using Differential Bloom Filter to Identify Abnormal Tags

We know that the differential Bloom filter encodes the IDs of all abnormal tags. Using the Bloom filter, we can identify a set of candidate missing tags for missing-tag identification or a set

of cloned tags for cloned-tag identification. Since Bloom filters do not lead to false negatives, but can cause false positives, all missing tags or cloned tags must be contained in the candidate set. The reader then can ping each tag in the candidate set (whose size is generally small), to verify if it is missing or cloned. The process of identifying unknown tags is a little different since the reader does not know the IDs of the unknown tags. To collect IDs from those unknown tags, the reader can broadcast the Bloom filter, and any tag passes the membership check should report its ID.

IV. Case Study: DBF for Missing-tag Identification

In this section, we apply the DBF framework to missing-tag identification, an anomaly-detection problem that has been well studied in the literature. The RFID reader knows the IDs of all tags in the original set N_0; note that if some tags are moved in or out from the system by normal activities, we need to take a new snapshot for the new tag set as N_0. After constructing each filter, the reader will identify the missing ones and thus know the remaining ones. As the inductive assumption, suppose the reader knows the correct set N_{i-1} of remaining tags. We show in this section that the reader is able to figure out the correct set of M_i (thus N_i) after obtaining B_i.

A. Prior Art

Li et al. [12] proposed a series of protocols for missing-tag identification based on a simple idea: If one and only one tag is mapped to a slot (which is called a singleton) and that slot turns out to be empty, then the tag must be missing. Their most efficient protocol (called THP) ensures that each tag will be mapped to a singleton and thus all missing ones can be identified. The number of time slots needed is $O(n)$ in order to encode the information of all tags in separate slots, where n is the number of tags in the system. Liu et al. proposed two protocols called MMTI [10] and SFMTI [11] to improve the time efficiency of THP based on the similar idea. Zhang et al. considered the case of multiple readers, and Protocol 3 is their most efficient protocol [13].

The previous binary-state solutions throw away a lot of useful information at the signal level. A slot only takes a binary value (busy or empty) even though the signal in the slot received by the reader carries more details. Zheng et al. proposed P-MTI [14] to utilize that information. Each of the n tags simultaneously transmits a pseudo-random number, consisting of physical-layer symbols (signals) representing '0's or '1's. Assuming bit-level synchronization, the reader receives a sequence of aggregate symbols. Each aggregate symbol is the combination of individual symbols from n tags, which may be in thousands. At a later time, the reader performs the same operation for a second sequence of aggregate symbols. If some tags are missing, based on the theory of compressive sensing, P-MTI tries to identify these tags by solving a convex optimization problem formulated from the difference between the two sequences of aggregate symbols. The difference is modeled as continuous signal waves. The convex optimization does not guarantee the identification of all missing tags, especially when the number of missing tags exceeds a threshold [14]. This is not acceptable for the applications that require the identification of every missing tag. Moreover, all tags in P-MTI transmit pseudo-random numbers simultaneously. With on-off keying where signals are transmitted only for '1's, there are $n/2$ tags transmitting at any bit time on average. P-MTI assumes that when multiple tags transmit simultaneously, the aggregate symbols received by the reader exhibit as the superposition of the individual symbols from tags. Experiments show that it is true for a small number of tags, e.g., 5 tags in [14], but the result may not be extrapolateable to a large number of tags in thousands. Each tag has small variation in its signals. The accumulated variation among a small number of tags may remain insignificant, but the accumulated variation among a very large number of tags that transmit together can be large.

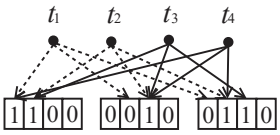

Fig. 3: An illustration of finding potentially missing tags using Bloom filter with three segments. t_3 and t_4 are two missing tags, while t_1 and t_2 are not missing tags, with t_2 causing a false positive.

B. DBF for Identifying the IDs of Potentially Missing tags

Recall from Section III-D that the segmented Bloom filter B_i encodes M_i. For each tag in M_i, its k representative bits in the filter are all ones. The reader performs a lookup for every tag t in N_{i-1}. Knowing the ID of t, the reader can generate the same pseudo-random bits that the tag uses to map itself to its representatives in the segments. If all k representative bits are ones, the reader inserts the tag's ID to a set M_i' for possible missing tags.

A Bloom filter does not have false negatives [22], which means that $M_i \subseteq M_i'$. All missing tags will be found in M_i'. However, a Bloom filter may have false positives, which means that tags in $N_{i-1} - M_i$ (not missing) may end up in M_i' because all their representative bits may happen to be ones. Fig. 3 gives an example. The Bloom filter encodes two missing tags, t_3 and t_4, whose representative bits are all ones, and thus they will be inserted into M_i'. Tag t_1 is not missing and thus not encoded in B_i. Because some of its representative bits are zeros, it will not be inserted into M_i'. Tag t_2 is not missing either, but all its representative bits are ones because they happen to be also the representative bits of t_3 or t_4. In this case, t_2 will end up in M_i'.

For an arbitrary tag in $N_{i-1} - M_i$, the probability for one of its representative bits to be one is $1-(1-\frac{1}{l})^{|M_i|}$. The probability for all k representative bits to be ones, also called *false-positive ratio* and denoted as p_f, is

$$p_f = (1 - (1 - \frac{1}{l})^{|M_i|})^k \approx (1 - e^{-\frac{|M_i|}{l}})^k, \tag{5}$$

which can be made arbitrarily small by increasing the value of l. For example, suppose $|N_{i-1}| = 10000$, $k = 10$, $l = 720$, and $|M_i| = 500$. The false-positive ratio will be just 0.001.

Therefore, the expected size of M_i' is given as follows:

$$E(|M_i'|) = |M_i| + p_f|N_{i-1} - M_i|. \tag{6}$$

With the parameters of the previous example, $E(|M_i'|)$ will be 510.

C. Confirming the Missing Tags

The reader confirms whether the tags in M_i' are actually missing through a simple polling phase: The reader transmits the tag IDs in M_i' one after another and requires any tag receiving its own ID to acknowledge. A tag must be missing if the expected acknowledgement is not received. After the polling phase, all missing tags will be identified.

D. Performance Improvement through Singleton Bits in B_i

A performance improvement step may be introduced before the polling phase based on a simple observation: If a tag t in M_i' is mapped a bit b in B_i that no other tag in M_i' is mapped to, then t must belong to M_i, i.e., it is a missing tag. This observation follows directly from Theorem 1: The fact that b is a representative bit of a tag in M_i' means that b must be one because otherwise the tag would not be inserted into M_i' in the first place. By the theorem, there must be a missing tag mapped to b. All missing tags are in M_i', and t is the only tag in M_i' mapped to b, which can only mean that t is missing.

We call a bit with only one tag mapped to it as a *singleton bit*. Hence, after mapping the tags in M_i' to the bits in B_i, the reader finds out which tags are mapped to singleton bits and those tags must be missing. We have the following proposition, the proof of which is omitted due to space limitation.

Proposition 2. *The probability p_s of a missing tag t to be mapped to a singleton bit is*

$$p_s = 1 - (1 - (1 - \frac{1}{l})^{|M_i|-1} \times \sum_{1 \le x \le \min\{l, |M_i|\}} \frac{\binom{l}{x} \times x! \times S(|M_i|, x)}{l^{|M_i|}} \times (1 - \frac{1}{x})^{|M_i' - M_i|})^k, \quad (7)$$

where $S(|M_i|, x) = \frac{1}{x!} \sum_{i=0}^{x} (-1)^i \binom{x}{i} (x-i)^{|M_i|}$.

Note that p_s is also the probability for any missing tag in M_i to be confirmed from M_i' based on singleton bits. Continuing the previous numerical example, with $|N_{i-1}| = 10000$, $k = 10$, $l = 720$, $|M_i| = 500$, and $|M_i'| = 510$, the value of p_s is 0.9987.

We remove the confirmed missing tags (based on singleton bits) from M_i' and pass the remaining tags to the polling phase described in the previous subsection. Following the above numerical example, with $|N_{i-1}| = 10000$, $k = 10$, $l = 720$, and $|M_i| = 500$, we find that $|M_i'|$ = 510 before this performance improvement step and $|M_i'|$ = 10 after the step, which will greatly reduce the polling overhead.

E. Setting the System Parameters

The values of the system parameters, l and k, are under our control. Below we show how to set their optimal values.

1) Optimal Value of l: Under an arbitrary value of k, below we find an optimal value of l that minimizes the false-positive ratio p_f. Taking the logarithm of both sides of (5) and then the first derivative of $\ln p_f$ with respect to l, we have

$$\frac{d(\ln p_f)}{dl} = \frac{dp_f}{p_f dl} = \frac{-k}{l} \ln(1 - e^{-\frac{|M_i|}{l}}) - \frac{\frac{|M_i|k}{l^2}}{1 - e^{-\frac{|M_i|}{l}}} e^{-\frac{|M_i|}{l}}. \quad (8)$$

By setting $\frac{d(\ln p_f)}{dl}$ to zero and solving the equation, we obtain the optimal value of l as follows:

$$l^* = \frac{|M_i|}{\ln 2}, \quad (9)$$

which is a function of $|M_i|$ (see further discussion shortly).

2) Optimal Value of k: The execution of the DBF protocol mainly consists of two phases: (1) Construct a differential Bloom filter to identify a set M_i' of potentially missing tags; (2) Poll the tags in M_i' to confirm their presence/absence. There exists a tradeoff between the time costs of these two phases: If k is larger, longer physical-layer snapshots are built in phase one such that p_f will be smaller and p_s will be larger. In this case, the execution time of phase one will rise, but the time expenditure for the polling phase will decrease since fewer unidentified tags will be left in M_i'. The opposite is true if k is smaller. Hence, our objective is to find the optimal value of k that minimizes the overall execution time T of the DBF protocol, which can be expressed as follows:

$$T = k(t_c + l \times t_s) + |M_i'|(t_{id} + t_s), \quad (10)$$

where t_c is the duration for the reader broadcasting a command to initialize the construction of a segment of the physical-layer snapshot. The expected value of T is

$$\begin{aligned} E(T) &= k(t_c + l \times t_s) + E(|M_i'|)(t_{id} + t_s) \\ &= k(t_c + l \times t_s) + (|M_i|(1 - p_s) + |N_i|p_f)(t_{id} + t_s). \end{aligned} \quad (11)$$

The value of p_s approaches to 1 quickly with the increase of k, and $|M_i|$ is generally much smaller compared to $|N_i|$. Therefore, the term $|M_i|(1-p_s)$ in (11) can be made negligibly small and is omitted for simplicity. Meanwhile, we let $l = \frac{|M_i|}{\ln 2}$, Eq. (11) is simplified as

$$E(T) \approx k(t_c + \frac{|M_i|}{\ln 2} \times t_s) + |N_i| \times (\frac{1}{2})^k \times (t_{id} + t_s). \quad (12)$$

Take the first derivative of (12) with respect to k and set it to zero, we have

$$\frac{dE(T)}{dk} \approx t_c + \frac{|M_i|}{\ln 2} t_s - \ln 2 \times |N_i| \times (t_{id} + t_s) \times (\frac{1}{2})^k = 0. \quad (13)$$

As a result, the optimal value for k is

$$k^* = \frac{\ln(\ln 2 |N_i|(t_{id} + t_s)) - \ln(t_c + \frac{|M_i|}{\ln 2} t_s)}{\ln 2}. \quad (14)$$

Note that k is a non-negative integer. We should round k^* either to the ceiling or to the floor, depending on which one results in a smaller value of $E(T)$.

F. Unknown Value of $|M_i|$

From (9) and (14), the optimal values of l and k are functions of $|M_i|$, which is however unknown. In practical applications, we may substitute it with an estimated upper bound C based on historical data, as what's been done in [14]. We stress that all missing tags will be identified in our design, even if $|M_i|$ turns out to be larger than C. When that happens, the only negative effect is that the DBF protocol may take more time than what the optimal setting could have achieved with the knowledge of $|M_i|$. We will investigate the impact of C on the performance of DBF in Section V.

G. Overhead

In the DBF protocol, each tag only needs to generate $k \log l = k \log \frac{C}{\ln 2}$ pseudo-random bits to index its k representative symbols, where k is typically a small number. Each tag needs to transmit those k symbols (bits). The reader needs to generate $|N_{i-1}|k \log \frac{C}{\ln 2}$ bits to index all tags' representatives, and the computation complexity of producing the differential symbol filter and the segmented Bloom filter is $O(kl) = O(kC)$. Because C must be set smaller than the number of existing tags, $|N_{i-1}|$, the overall computation complexity is $O(|N_{i-1}|k \log C)$ at the reader. The communication complexity for the reader to take a snapshot is $O(kl) = O(kC)$ since the total number of symbols in the snapshot is kl.

H. Multiple Readers

The communication range of a single RFID reader is limited (e.g., 10 meters) due to power constraints. Hence, multiple readers should be deployed to ensure all tags in the system can be covered. If multiple readers share the same channel, the reader-to-reader collision problem may arise. Namely, if two readers broadcast their commands simultaneously, the tags located in their overlapping area cannot resolve the collided messages. A basic solution to avoid reader-to-reader collision is to divide the readers into non-interfering groups and schedule each group at different times for missing-tag identification, with the readers in the same group executing the protocol in parallel. Each non-interfering group can be determined through experiments at the time when readers are installed. Reader scheduling can be implemented by the back-end server. The union of missing tags identified by all readers gives the set of missing tags in the system. Therefore, the problem of missing-tag identification in the multi-reader scenario can be reduced to reader scheduling plus missing-tag identification in each non-interfering group. Since the readers in the same group will run in parallel, the time efficiency for each group is determined by the bottleneck reader in the group that has the maximum number of tags in its coverage area. Therefore, the comparison of different missing-tag identification protocols under a multi-reader scenario is equivalent to the sum of the comparison under single bottleneck readers.

V. Performance Evaluation of DBF for Missing-tag Identification

We use simulations to evaluate the performance of DBF for missing-tag identification in large RFID systems, which will be complemented with a small-scale implementation for feasibility demonstration. We compare the proposed DBF with the state-of-art protocols P-MTI [14], THP [12], SFMTI[1] [10] and Protocol 3 [13]. Since it takes much more time to identify all tags in a large RFID system, we do not include tag identification protocols in the comparison.

A. Simulation Setup

Following the parameter setting in [14], we assume both the reader-to-tag transmission rate and the tag-to-reader transmission rate are 40kbps. We measure the execution time in terms of the number of time slots, each of which can carry one symbol from tags. The time slot is a generic concept which fits with different interpretations in the literature. For example, the experiments in [14], [20] show that the bit-level synchronization of off-shelf passive RFID tags is achievable after adopting drift correction. In this case, a time slot is the time of transmitting a bit. However, those experiments are performed for only a small number of consecutive symbols (bits), while the protocols under comparison in this section all require transmissions of long sequences of consecutive symbols in thousands. A more conservative interpretation of slots can be found in [12], where each slot includes one symbol and a waiting time (0.30ms used in [12] according to the Philips I-code specification [23] and 0.27ms used in [9] according to the EPC C1G2 standard [1]). Even more conservatively, each slot includes a reader-tag exchange, with the reader issuing a command and tags responding with a symbol [24]. The reader's command also serves for the purpose of synchronization. Our protocols also require the reader to transmit tag IDs (96 bits each) to resolve false positives. We conservatively count each bit from the reader also as a slot.

[1]SFMTI and MMTI [11] are proposed by the same authors. SFMTI outperforms MMTI, so MMTI is not included for comparison.

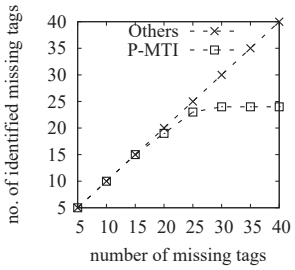

Fig. 4: Number of identified missing tags by P-MTI, and other protocols including DBF, THP, SFMTI and Protocol 3.

P-MTI has to set the value of C large in order to guard against false negatives. In the original paper, it is set as a fraction of $|N_{i-1}|$. We set it as $0.1|N_{i-1}|$ by default unless it is otherwise explicitly specified. Our protocols do not require such a large value of C because they do not have false negatives. We nevertheless set $C = 0.1|N_{i-1}|$ in order to compare our protocols with P-MTI on the same footing even though that can cause longer execution time for our protocols.

Under each parameter setting, we run the simulation 500 times and average the results.

B. Number of Identified Missing Tags

One important performance metric for a missing-tag identification protocol is whether it can exactly identify all missing tags. We set $|N_{i-1}| = 1000$, vary $|M_i|$ from 5 to 40 at steps of 5, and set C to 20 for DBF and P-MTI. Fig. 4 illustrates the numbers of missing tags identified by these protocols. All protocols except P-MTI can always exactly identify all missing tags. Although P-MTI has no false positive, it fails to identify some missing tags when $|M_i|$ becomes close to or larger than C, resulting in false negatives. In missing-tag identification, false negatives tend to be worse than false positives. For example, if a stolen product is not detected in time, we may miss the best time to trace it and get it back.

C. Execution Time

The second performance metric to study is time efficiency. In the first set of simulations, we vary $|N_{i-1}|$ from 1000 to 10000

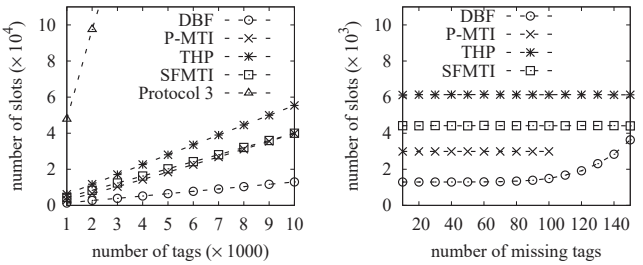

Fig. 5: *Left plot:* Execution time of DBF, P-MTI, THP, SFMTI and Protocol 3 with different numbers of tags. *Right plot:* Execution time of DBF, P-MTI, THP, and SFMTI with different numbers of missing tags.

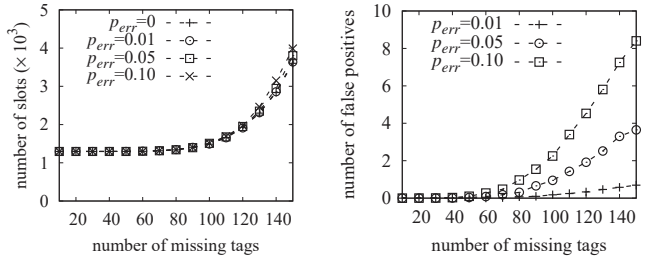

Fig. 6: *Left plot:* Execution time of DBF with respect to channel error rate. *Right plot:* False positives with respect to channel error rate.

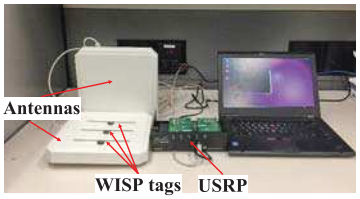

Fig. 7: A testbed for the DBF protocol. Two circular antennas are mounted to USRP1 that is connected to a laptop to work as an RFID reader.

at steps of 1000. For each N_{i-1}, we let $|M_i| = 0.05|N_{i-1}|$, and $C = 0.1|N_{i-1}|$ for DBF and P-MTI. The left plot of Fig. 5 shows the results, from which we can see that DBF is much more efficient than P-MTI, THP, SFMTI and Protocol 3, especially when the number of tags is getting larger. For example, when the number of tags in the system is 10000, DBF only takes 32.6% of the execution time of P-MTI, 32.5% of the execution time of SFMTI, 23.4% of the execution time of THP, and 2.7% of the execution time of Protocol 3, respectively, to identify the 500 missing tags.

In the second set of simulations, we set $|N_{i-1}| = 1000$, $C = 0.1|N_{i-1}|$ for DBF and P-MTI, and vary $|M_i|$ from 10 to 150 at steps of 10. The right plot of Fig. 5 presents the execution time of DBF, P-MTI, THP, and SFMTI with respect to the number of missing tags. Protocol 3 takes approximately 50,000 slots to identify all missing tags, much more time-consuming than the other four protocols. To make the trend of protocol execution time with respect to $|M_i|$ more legible, we do not plot Protocol 3 so that we can reduce the vertical scale. Recall that when $|M_i| > C$, P-MTI cannot identify all missing tags. Therefore, we only consider the execution time of P-MTI when $|M_i| \leq C = 100$ for fair comparison. In general, DBF outperforms the other four protocols in terms of time efficiency. We observe that the execution time of THP, SFMTI and P-MTI is not sensitive to the change of $|M_i|$. In contrast, the performance of DBF increases when $|M_i| > C$.

D. Impact of Channel Error

Finally, we investigate the impact of channel error on the performance of DBF in identifying missing tags. We adopt the *random channel error model*. The model is characterized by a parameter p_{err} called *error rate*, which means any symbol in the differential filter has a probability p_{err} to be corrupted by the channel error (whose magnitude exceeds the threshold θ.). As a result, the corresponding bit in the Bloom filter will be set to 1 according to (4), resulting in a higher false-positive ratio.

In the simulations, we set $|N_{i-1}| = 1000$, $C = 0.1|N_{i-1}|$, and vary $|M_i|$ from 10 to 150 at steps of 10. The value of p_{err} is set to 0.01, 0.05 and 0.1, respectively. The left plot of Fig. 6 shows the execution time of DBF under different channel error rates. It is clear that the execution time of DBF is not significantly affected. Even if $p_{err} = 0.1$, the execution time increases less than 10% when comparing with the execution time without channel error. The right plot of Fig. 6 shows the increased number of false positives as the channel error rate increases, which is expected. When $|M_i|$ is small, the number of false positives is negligible. When $|M_i| > C$, the number of false positives is still in an acceptable range. For example, when $p_{err} = 0.1$ and $|M_i| = 150$, there are 8.4 false positives ($\approx 1\%$) on average. We emphasize that DBF does not have any false negative, with or without channel error.

VI. Prototype Implementation

We implemented a prototype of DBF using USRP1 and WISP programmable tags [25] for missing-tag identification. As shown in Fig. 7, two circular antennas are mounted to USRP1 that is connected to a laptop to serve as an RFID reader. The software defined reader works in the 915MHz UHF band and has full control over the RFID physical layer. We set the sampling rate of the reader to 4MHz and the backscatter link frequency of the WISP tags to 24KHz. Besides the mandatory commands (e.g., Query, Select, Read), C1G2 allows the users to defined a set of custom commands. To enable the reader and tags to run DBF, they should support the DBF command, which is added as a custom command.

The parameters of DBF were set as follows: Each physical-layer snapshot consisted of two segments, i.e., $k = 2$, each segment allowed transmissions of 20 symbols, i.e., $l = 20$, and there are 16 tags. The experiments are performed in our lab. Fig. 8 shows two executions of DBF. After the reader started (0.35ms), it first powered up the tags by transmitting continuous carrier waves (0.35–1.48ms). During the first execution, the reader broadcasts a DBF command (1.48–2.19ms) to initiate the DBF protocol. Upon receiving the DBF command, each of the 16 tags randomly selected its representative in each segment and sent a response to the reader (2.33–3.47ms), which formed the 1st physical-layer snapshot. In the second execution, two tags were programmed to keep silent to emulate missing tags, and the remaining 14 tags responded as usual. The reader broadcast another DBF command (4.14–4.86ms), and the 14 non-missing tags chose their representatives (the same ones as in the first

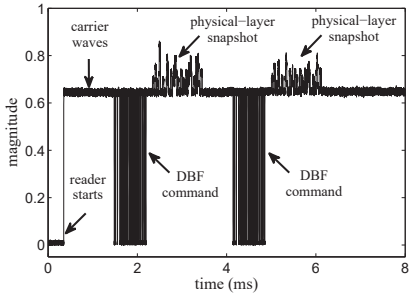

Fig. 8: Two executions of DBF with 16 WISP tags.

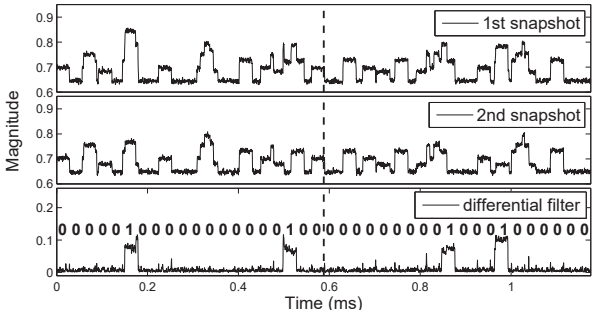

Fig. 9: Two physical-layer snapshots received by the reader and the corresponding differential Bloom filter.

execution) to send responses to the reader (5.00–6.12ms), which formed the 2nd physical-layer snapshot.

The construction of a differential Bloom filter is shown by Fig. 9, where the magnitude of each signal is depicted. The first plot from the top shows the 1st physical-layer snapshot for 16 initial tags. The dashed line in the middle shows the boundary of the two segments in the snapshot. The second plot shows the 2nd snapshot for 14 tags after while two tags are missing. By subtracting the aggregate signals in the 2nd physical-layer snapshot from the 1st physical-layer snapshot, we obtained the differential symbol filter, which is presented in the third plot. Despite the slight jitters in tags' responses, the aggregate signals can be roughly aligned [20]. From the differential symbol filter we can clearly recognize the representatives of the missing tags, which are the sixth symbol and the eighteenth symbol in the first segment, and the tenth symbol and the fourteenth symbol in the second segment, exactly matching the true representatives of the missing tags. After that, the differential symbol filter is converted to a segmented Bloom filter for missing-tag identification as discussed in Section III. The Bloom filter is shown in the third plot.

VII. Conclusion

In this paper, we introduce a new concept called differential Bloom filter, which is constructed from physical-layer differential symbols. It is carefully designed such that the filter only encodes the abnormal tags. Based on this concept, we propose a general framework for anomaly detection that guarantees the identification of all abnormal tags and in the meantime significantly reduces the execution time. We apply the proposed framework to missing-tag identification as a case study. The simulation results demonstrate that our solution drastically outperforms the best existing protocols for missing-tag identification. In addition, We implement a prototype using USRP and WISP tags to verify the effectiveness of the proposed solution. In our future work, we will study how to apply the framework to anomaly detection where different types of abnormal tags coexist. This is a more practical and complicated problem that has not yet be investigated.

VIII. Acknowledgments

This work is supported in part by National Science Foundation of United States under grants CNS-1115548.

References

[1] "EPC Radio-Frequency Identity Protocols Class-1 Gen-2 UHF RFID Protocol for Communications at 860MHz-960MHz, EPCglobal," *http://www.epcglobalinc.org/uhfclg2*.

[2] J. Liu, B. Xiao, X. Liu, and L. Chen, "Fast rfid polling protocols," *Proc. of ICPP*, pp. 304–313, August 2016.

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

[4] M. Chen and S. Chen, "Identifying state-free networked tags," *Proc. of IEEE ICNP*, pp. 267–278, November 2015.

[5] L. Xie, H. Han, Q. Li, J. Wu, and S. Lu, "Efficiently Collecting Histograms Over RFID Tags," *Proc. of IEEE INFOCOM*, April-May 2014.

[6] L. Shangguan, M. Li, Z. Yang, M. Li, and Y. Liu, "OTrack: Order Tracking for Luggage in Mobile RFID Systems," *IEEE INFOCOM*, 2013.

[7] E. C. Jones and C. A. Chung, *RFID in Logistics: A Practical Introduction*, CRC Press, 2007.

[8] C. Tan, B. Sheng, and Q. Li, "How to Monitor for Missing RFID tags," *Proc. of IEEE ICDCS*, June 2008.

[9] W. Luo, Y. Qiao, S. Chen, and T. Li, "Missing-Tag Detection and Energy-Time Tradeoff in Large-Scale RFID Systems with Unreliable Channels," *IEEE/ACM Transactions on Networking*, vol. 22, no. 4, August 2014.

[10] Xiulong Liu, Keqiu Li, Geyong Min, Yanming Shen, A.X. Liu, and Wenyu Qu, "Completely pinpointing the missing rfid tags in a time-efficient way," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 87–96, Jan 2015.

[11] Xiulong Liu, Keqiu Li, Geyong Min, Yanming Shen, A.X. Liu, and Wenyu Qu, "A multiple hashing approach to complete identification of missing rfid tags," *IEEE Transactions on Communications*, vol. 62, no. 3, pp. 1046–1057, March 2014.

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

[13] R. Zhang, Y. Liu, Y. Zhang, and J. Sun, "Fast Identification of the Missing Tags in a Large RFID System," *Proc. of IEEE SECON*, June 2011.

[14] Y. Zheng and M. Li, "P-MTI: Physical-layer Missing Tag Identification via compressive sensing," *Proc. of IEEE INFOCOM*, pp. 917 – 925, 2013.

[15] X. Liu, H. Qi, K. Li, I. Stojmenovic, A. X. Liu, Y. Shen, W. Qu, and W. Xue, "Sampling bloom filter-based detection of unknown rfid tags," *IEEE Transactions on Communications*, vol. 63, no. 4, pp. 1432–1442, April 2015.

[16] X. Liu, B. Xiao, S. Zhang, and K. Bu, "Unknown tag identification in large rfid systems: An efficient and complete solution," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, June 2015.

[17] K. Bu, X. Liu, J. Luo, B. Xiao, and G. Wei, "Unreconciled collisions uncover cloning attacks in anonymous rfid systems," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 3, pp. 429–439, March 2013.

[18] K. Bu, M. Xu, X. Liu, J. Luo, and S. Zhang, "Toward fast and deterministic clone detection for large anonymous rfid systems," *Proc. of IEEE MASS*, pp. 416–424, October 2014.

[19] M. Lehtonen, F. Michahelles, and E. Fleisch, "How to detect cloned tags in a reliable way from incomplete rfid traces," *Proc. of IEEE RFID*, pp. 257–264, April 2009.

[20] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk, "Efficient and Reliable Low-power Backscatter Networks," *Proc. of ACM SIGCOMM*, 2012.

[21] T. Li, S. Chen, and Y. Ling, "Identifying the Missing Tags in a Large RFID System," *Proc. of ACM Mobihoc*, pp. 1–10, September 2010.

[22] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Math*, vol. 1, no. 4, pp. 485–509, 2003.

[23] Philips Semiconductors, "Your Supplier Guide to ICODE Smart Label Solutions," *http://www.nxp.com/acrobat_download2/other/icode_supplier_list_2008_10.pdf*, October 2008.

[24] B. Chen, Z. Zhou, and H. Yu, "Understanding RFID Counting Protocols," *Proc. of ACM Mobicom*, 2013.

[25] "WISP," *http://wisp.wikispaces.com/*.